UNIVERSITÀ DEGLI STUDI DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

A TRUSTLESS ARCHITECTURE FOR BLOCKCHAIN-BASED IoT APPLICATIONS USING CONSTRAINED DEVICES

Miguel Rodrigo Pincheira Caro

Advisor

Fabio Antonelli

Fondazione Bruno Kessler

Co-Advisor

Prof. Massimo Vecchio

Fondazione Bruno Kessler

May 2021

Abstract

Despite the increasing interest in blockchain as a possible solution to replace centralized IoT architectures, previous work failed to provide a direct role for the sensing devices, i.e., direct interaction with the blockchain without additional components. Moreover, few studies focus on permissionless blockchains, even if it is the most secure platform for developing blockchain-based applications.

This thesis presents an architecture that considers constrained sensing devices as direct actors on a public blockchain network. A public blockchain network allows the seamless inclusion of several unknown actors, and smart contracts provide a platform to develop complex IoT applications. The research followed an iterative DSR approach; designing, building, and evaluating new IT artifacts using two case studies in the agricultural IoT domain. These cases fostered two exploratory studies that diverged from the main IoT domain; however, they also provide novel contributions to blockchain-based applications.

Thus, the novel architecture tackles three problems of current blockchain-based IoT systems i) constrained sensing devices as direct actors on a blockchain system, ii) permissionless blockchain networks and iii) smart contracts as an IoT application platform. Furthermore, the exploratory analyses examine two challenges of blockchain-based applications i) user experience and monetary costs and ii) data sharing and decentralized storage.

Keywords

[Internet of Things; Blockchain; Smart Contracts; Distributed Ledger]

Acknowledgements

This work is the conclusion of a journey full of challenges and difficulties but always supported by family, friends, and colleagues to whom I am deeply thankful. First, I would like to thank my supervisor, Mr. Fabio Antonelli, for the opportunity to take this motivating journey under his tutelage, support, and experience. Next, I would like to thank my co-supervisor, Dr. Massimo Vecchio, who has been a constant source of motivation and support throughout my learning experience, always encouraging my academic and professional growth. I also thank Mr. Raffaele Giaffreda for sharing his valuable experience and knowledge. A special thanks to the OpenIoT unit that welcomed me to a warm and enjoyable working environment full of coffee, laughs, and food tips.

I want to thank my friends and family in Chile. Our shared memories overcame the distance and always remind me of the happiness of life. And to new friends, that provided a home away from home, with laughs, food, and wine. I am particularly grateful to Elena, who not only offered her friendship but selflessly shared her time and knowledge through this Ph.D. journey.

My most important thank is to my family, Tamy. She is the source of strength and encouragement. Without her unconditional love and support, this journey will not have been possible. I thank my father for his continuous support. I thankful to my Socia and those who are not with me. Finally, I thank my mother for all that she taught me. Her memory is and always will be a continuous light of guidance.

The works in this thesis was partially supported by (1) the AGILE project (G.A. No 688088) and (2) the DECENTER project (G.A No 815141) within the Horizon 2020 Programme of the EU, and (3) the SAPIENCE project within the Climate KIC programme (P.ID No 200081) of the EIT.

Contents

1 Introduction **1**

1.1 Towards decentralization of the Internet of Things 2

1.2 Sensing devices in blockchain-based IoT systems 3

1.3 A novel trustless architecture for blockchain-based IoT applications 4

1.3.1 Design Science Research approach 5

1.3.2 Case studies 5

1.3.3 Exploratory analyses 6

1.3.4 Innovative aspects 7

1.4 Structure of the thesis 8

Part I: Background on blockchain technology and IoT **13**

2 A primer of blockchain technology **13**

2.1 Blockchain working principles 14

2.1.1 Transactions and digital signatures 16

2.2 Taxonomy of blockchain networks 18

2.2.1 Public/permissionless blockchains 19

2.2.2 Private/permissioned blockchains 19

2.3 Decentralized consensus algorithms 20

2.3.1 Consensus on permissionless blockchains 21

2.3.2 Consensus on permissioned blockchains 24

2.3.3 Comparison of permissionless and permissioned consensus algorithms 27

2.4 Smart contracts 28

2.5 Salient features of blockchain technology 30

3 State of the Art in integrating blockchain and IoT 33

3.1 IoT layered architectures 35

3.1.1 Requirements of IoT systems 38

3.1.2 Challenges in centralized IoT architectures 41

3.1.3 Decentralization of IoT through blockchain technology 43

3.2 Integration schemes for blockchain and IoT 44

3.2.1 Cloud-level integration 47

3.2.2 Edge-level integration 47

3.2.3 Device-level integration 49

3.3 Interactions through smart contracts 49

3.4 Summary and gap analysis 54

Part II: Blockchain-based decentralized IoT applications 59

4 A novel trustless architecture for blockchain-based IoT applications 59

4.1 High-level architecture 60

4.2 Novel contributions 61

4.2.1 Constrained sensing devices as direct actors on the blockchain system 61

4.2.2 Permissionless blockchain as the underlying network . 62

4.2.3 Smart contracts as a software platform 62

4.3 Blockchain-based decentralized IoT framework 63

4.3.1 M1: Device module 65
4.3.2 M2: Gateway module 67
4.3.3 M3: Blockchain module 68

5 A blockchain-based IoT system for traceability in Agri-Food 71

5.1 Introduction 72
5.2 Related work 73
5.3 Proposed system architecture 75
5.4 Architecture evaluation 80
5.5 Conclusions 81

6 Cost-effective IoT devices in a blockchain-based water management system 83

6.1 Introduction 84
6.2 State of the art and rationale 87
6.3 Proposed system architecture 89
6.4 Implementation 93
6.4.1 The Device module 94
6.4.2 The Gateway module 96
6.4.3 The Blockchain module 97
6.5 Architecture evaluation 98
6.5.1 Device module footprint 100
6.5.2 Device module performance 100
6.5.3 Transaction compression 101
6.5.4 Transaction cost and processing time 103
6.5.5 Device power consumption and energy budget 104
6.6 Conclusions 107

Part III: Exploratory analyses of blockchain-based applications challenges 111

7 Cost and user experience in blockchain-based applications 111

7.1 Rationale and practical assessment of a fully distributed blockchain-based marketplace of Fog/Edge computing resources 112

7.1.1 Introduction 112

7.1.2 Related work 114

7.1.3 Proposed system architecture 116

7.1.4 Experimental setup 119

7.1.5 Architecture evaluation 121

7.1.6 Conclusions and future works 128

7.2 Cost Model for blockchain-based applications 130

7.2.1 Introduction 130

7.2.2 Application description and requirements 131

7.2.3 Proposed transaction taxonomy 132

7.2.4 Cost model for permissionless networks 133

7.2.5 Evaluation of the marketplace application 138

7.2.6 Conclusions 141

8 Data-sharing and decentralised storage in blockchain-based applications 143

8.1 A blockchain-based approach to enable remote sensing trusted data. 143

8.1.1 Introduction 143

8.1.2 Proposed architecture 147

8.1.3 Use case: precision agriculture 153

8.1.4 Conclusions and future works 156

8.2 Decentralized storage for trusted data sharing 157

8.2.1 Introduction 157

8.2.2 Proposed Architecture 159
8.2.3 Experimental setup 162
8.2.4 Evaluation results 164
8.2.5 Conclusions 170

9 Conclusions **173**
9.1 Novel contributions 174
9.2 Future works 176

Bibliography **177**

Appendix A: List of Abbreviations **200**

List of Tables

2.1 Comparison of public and private blockchains. 29

3.1 Node types in blockchain networks 45

5.1 Performance of AgriBlockIoT in terms of latency, network traffic, and CPU load. 81

6.1 Components of a transaction in the Ethereum blockchain. . . 94
6.2 Main characteristics of the hardware platforms used in the evaluation campaign. 99
6.3 Device module program size footprint (in bytes). 100
6.4 Device module memory footprint (in bytes). 101
6.5 Average processing time overhead of the device module (in milliseconds). 102
6.6 Transaction costs for different values of T_p (slow, avg, and fast correspond to 2, 5, and 10 gwei, respectively, while 1 $ETH =$ 205 USD is the exchange rate). 104
6.7 Average energy consumption of the device module at 5V (in Joules). 105

7.1 Average cost for creating reservations ($T_p = 10\,gwei$ and $C_e = 205\,USD$). 121
7.2 Average cost for creating and updating advertisements ($T_p = 10\,gwei$ and $C_e = 205\,USD$). 123

7.3 Average transaction cost for creating the marketplace ($T_p = 10$ gwei and $C_e = 205$ USD). 124
7.4 Average time response (in seconds) for querying the list of advertisements from the MSC. 125
7.5 Average time response (in seconds) for querying details of all the advertisement. 126
7.6 Average time response (in seconds) for querying the detailed list of advertisements using local buffer. 127
7.7 Average processing time (in seconds) and cost of the functionalities in a live network ($T_p = 10\, gwei$ and $C_e = 205\, USD$). . 128
7.8 Basic requirements of blockchain-based application on permissionless networks. 132
7.9 Parameters for the cost model of infrastructure using a permissionless blockchain. 134
7.10 Basic transactions for a marketplace classified using the CRIV taxonomy. 139
7.11 Parameters for the cost model of the marketplace using in a public blockchain. 140
7.12 Cost scenario for a decentralized marketplace on a public blockchain.141

8.1 Transaction size and cost. 165
8.2 Transaction processing times in terms of minimum, maximum, average, and variation on a Ropsten and a Goerli network. . . 166
8.3 Upload times expressed in seconds. 168
8.4 Download times expressed in seconds. 170

List of Figures

2.1 Blockchain data structure. 15

2.2 Transactions in a Merkle Tree. 16

2.3 Public key signature process. 17

2.4 Performance and scalablity of different consensus (adapted from [1]). 28

2.5 Main features of blockchain technology. 31

3.1 Component-Oriented Architecture for modern IoT systems. . 36

3.2 Service-Oriented Architecture for IoT systems (extracted from [2]) 37

3.3 Blockchain integration for IoT systems 46

3.4 Two examples of IoT architectures integrating blockchain at Edge-level. 53

4.1 High-level overview of the novel trustless architecture for blockchain-based IoT applications 60

4.2 Blockchain-based decentralized IoT framework. 63

4.3 Trustless architecture: Constrained Sensing devices (D) provide trustworthy information to a permissionless blockchain (B) where unknown users (U) securely interact with decentralized IoT applications. 66

4.4 Class diagram of a sample application using Smart Twins, and Smart Twin Apps. 69

4.5 Example interactions over the proposed architecture. 70

5.1 Simplified version of the Agri-Food supply chain management process. 76
5.2 Layered architecture of AgriBlockIoT. 77

6.1 Typical high-level 3-layer architecture of modern IoT systems. 87
6.2 High-level architecture of the proposed blockchain-based sustainable water management system. 90
6.3 Software modules of the proposed architecture. 92
6.4 Device module footprint in terms of program size (in percentage). 101
6.5 Device module footprint in terms of memory (in percentage). 102
6.6 Average current consumptions at 5V. 105
6.7 Estimated daily energy budget distribution for (a) reactive and (b) continuous monitoring. 106

7.1 The proposed fully-distributed e-marketplace software architecture. 117
7.2 Sequence diagram of the interactions within the proposed e-marketplace. 119
7.3 Average time response (in seconds) for querying the list of advertisements 125
7.4 Average time response (in seconds) for obtaining the details of all advertisements. 126
7.5 Average time for all advertisements using local buffer. 127
7.6 Life-cycle of a blockchain-based application 133
7.7 Cost and benefits per transaction in the marketplace reflecting the volatility of the cryptocurrency in 2020. 141

8.1 Proposed architecture – Untrusted actors interact using the smart contracts to share, search, retrieve, and score metadata and receive rewards. 147
8.2 Geographic extension of the datasets shared in the blockchain system– in blue, the IoT raw and processed data, and in yellow, the processed Sentinel-2A data. 152
8.3 Simplification of decentralized storage protocol (based on IPFS and Swarm). 158
8.4 Proposed architecture based on blockchain, smart contracts, and DFS to provide an infrastructure for sharing datasets among untrusted actors. 161
8.5 Transactions processing times histograms 167
8.6 Comparison of upload and download times for different file size.169

Chapter 1

Introduction

In recent years, the Internet has expanded its limits from business to social interactions and from people to everyday things. Today, not only computers connect to the Internet, but also objects such as televisions, cars, and bikes, interact over the network under the field of the "Internet of Things" (IoT). Everyday objects now incorporate technical capabilities to connect the digital and physical world. IoT is growing at such a speed that in December 2017, the number of connected objects was over 8 billion [1], and the number of Internet users was about 4 billion [2]. Moreover, recent studies estimate that by 2025, the number of connected devices will be over 30 billion [3]. The IoT aims to make the Internet more immersive and pervasive by enabling easy interaction with a wide variety of devices [3], fostering the development of applications to provide new services to individuals [3] and transforming existing industrial operations [4].

[1]https://www.gartner.com/

[2]https://www.internetworldstats.com/stats.htm

[3]https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

1.1 Towards decentralization of the Internet of Things

Billions of sensing devices build the core of IoT, collecting information from the physical world. These devices interact with each other and with other technological components, creating complex IoT systems for several applications [5]. The ubiquity of Internet connections fosters a steady growth of IoT systems worldwide, which is also encouraged by the decreasing price and size of embedded computers [6]. Every day, smaller and cost-effective devices empower new kinds of IoT applications including infrastructure monitoring, smart homes, precision agriculture, personal healthcare, and industrial manufacturing [7]. As billions of devices are connected, it is necessary to define appropriate architectures that allow easy communications, control, and functional applications [8].

Centralized cloud-based architectures are the current choice for IoT systems, where a validating third party provides services such as authentication, authorization, and data handling for both sensing devices and end-users [9]. These architectures simplify the design and the deployment of IoT systems and applications. However, they introduce additional concerns regarding data management and privacy, as well as an over-exposition to several cyber-security threats [10]. Moreover, the presence of the intermediary decreases the efficiency of interactions between devices, reducing the potential growth of the IoT applications [7]. These challenges require radically new approaches for the architectures behind the next-generation IoT systems and applications.

In recent years, there has been a growing interest in integrating blockchain technologies into IoT [11] for enabling trustless architectures. Blockchain uses a unique combination of cryptography, data structures, and incentive mechanisms to maintain a particular type of distributed database (i.e., a *ledger*) in a peer-to-peer network. The distributed ledger is immutable by design and offers an auditable and transparent source of information. Integrating blockchain into IoT could enable decentralized applications without a validating central authority [12]. Blockchain provides a trusted repository of information for IoT systems, where data is secure and traceable, and the data source can be precisely identified [13]. Moreover, blockchain directly benefits several business processes (e.g., accounting, billing [14]) by enabling the seamless inclusion of several actors [15] and making their interactions leaner, faster, and more transparent [11].

1.2 Sensing devices in blockchain-based IoT systems

Despite the growing interest in blockchains, one aspect that previous works have failed to consider is a direct role for the sensing device, i.e., the capability of devices to interact with the blockchain without the intervention of an additional system component. The sensing devices are typically the most constrained hardware component of the technology stack. Thus, devices that autonomously interact with the blockchain might offer limited onboard computing resources with restricted energy budgets [16, 17]. Current studies consider scenarios without communications or energy restrictions [18, 19, 20]. This approach contrasts with the average requirements of IoT applications favoring power efficiency and low-cost to ensure cost-effective and long-term operations (e.g., agriculture, smart cities) [21, 22]. For restricted scenarios, the current research focuses on sensing devices that rely on another component to interact with the blockchain (e.g., gateway nodes [23, 24, 25]). However, the intermediary still introduces security concerns [6], reducing the trustworthiness of the sensed data [11]. Another aspect missing in the current literature is the benefits of using permissionless blockchain infrastructures as most of the studies focus on permissioned infrastructures [18, 26, 27]. However, a permissioned blockchain does not provide the same level of openness, decentralization, and neutrality as a permissionless network. Moreover, the type of consensus and the size of existing permissionless blockchains networks offer a more secure platform for developing decentralized applications [28, 29]. Thus, a need emerges for an architecture that acknowledges constrained devices on permissionless blockchain networks and provides the tools to develop new types of decentralized IoT applications.

1.3 A novel trustless architecture for blockchain-based IoT applications

This thesis presents an architecture that integrates blockchain technology into the Internet of Things, enabling the development of next-generation IoT applications. The architecture relies on blockchain inherent features to mitigate some of the main challenges in centralized IoT architectures. The architecture assumes that each device securely manages its cryptographic keys. Although this is a strong assumption, it is aligned with the state of the art [30, 31]. Furthermore, blockchain holds the potential to more complex identity schemes as key-building blocks for realizing completely decentralized public key infrastructures [11]. However, this research falls outside the focus of this thesis.

The proposed architecture considers the sensing devices as direct actors on a permissionless blockchain network to guarantee a *root of trust* for the sensed data [32]. Moreover, the integrity, auditability, and traceability of the sensed data are maintained and enforced by the blockchain network [12].

In the proposed architecture, smart contracts provide a platform to define complex business logic by autonomously enforcing agreements between untrusted actors [33], based on trusted values coming from ubiquitous IoT devices. We argue that this tamper-proof, immutable, and decentralized repository of sensed real-world information is the key to build novel decentralized IoT applications.

1.3.1 Design Science Research approach

"Design science research (DSR) is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artifacts, thereby contributing new knowledge to the body of scientific evidence. The designed artifacts are both useful and fundamental in understanding that problem" [34].

The DSR process starts with a problem with practical significance and involves two essential stages, building and evaluating. The building stage is required to produce a new Information Technology (IT) artifact. The evaluation involves the assessment of the newly created IT artifact. The evaluation process provides feedback and generates new knowledge, improving both the design and artifact [35]. Thus, a rigorous DSR approach guides towards more general and more valuable contributions. In this thesis, the DSR approach enables the generation of theoretical insights rather than starting with a core theory, similarly to [35, 36]. We believe that a DSR approach is favorable for this research, considering the early stages of research and development of blockchain technology.

1.3.2 Case studies

The proposed architecture is the result of a DSR process using two case studies in the agricultural IoT domain. The first iteration addressed the problem of reconstructing a centralized IoT application with a decentralized blockchain-based architecture. We targeted food traceability applications (Agri-Food), considering the growing adoption of IoT in this domain. Current IoT-based traceability systems for Agri-Food supply chains are built on top of centralized infrastructures, leaving unsolved issues and concerns such as data integrity, tampering, and single points of failure. We propose a blockchain-based traceability architecture for Agri-Food supply systems that seamlessly integrates IoT devices producing and consuming digital data along the process. Building and evaluating the IT artifact provided valuable knowledge about blockchain-based IoT applications. These steps clarified the requirements of large-scale IoT deployments and highlighted a crucial problem with current blockchain-based IoT applications– the role of constrained sensing devices in the architecture.

The second iteration explored the use of constrained IoT devices on public blockchains networks for incentivizing and rewarding sustainable water management practices in agriculture. Current IoT precision agriculture applications focus on energy efficiency, which generally translates into power-and-resource-constrained sensing devices. We proposed an architecture that enables a trustless water management system where constrained IoT devices can directly transact sensed data on a public blockchain network. We implement the proposed solution on off-the-shelf hardware devices and quantitatively assess the impact in terms of memory, program size, communication overheads, and power consumption. The encouraging results and the benefits of public blockchain networks as open and trustless application platforms further defined the proposed architecture.

1.3.3 Exploratory analyses

The iterative process over these two cases generated new knowledge about the strengths and challenges of blockchain-based IoT applications and fostered two exploratory studies. These case studies diverged from the main IoT domain of the thesis; however, they provide novel contributions to blockchain-based systems.

The first exploratory analysis focuses on the performance and cost evaluation of a blockchain-based application. Current blockchain-applications tend to use additional components for compensating the current limitations of public blockchains (e.g., data storage). We describe a case study where smart contracts provide all the functionalities needed by the blockchain-based application. We develop a full-fledged e-marketplace and evaluated several design decisions in terms of cost and performance. Furthermore, we propose a financial cost model for blockchain-based applications on permissionless networks.

The second exploratory analysis provides a performance evaluation of decentralized storage to complement blockchain-based applications. Currently, sharing and retrieving data is possible through tools managed by intermediaries. We present an architecture based on blockchain to build a system to share and retrieve data acquired by untrusted sources in a decentralized way. We analyze a case study on precision agriculture, and we evaluate decentralized storage as a possible solution to the current limitations of public blockchain in terms of data storage.

1.3.4 Innovative aspects

The proposed architecture tackles three main problems of current blockchain-based IoT systems:

- **Constrained sensing devices as direct actors on a blockchain system.** Current blockchain-based IoT systems favor powerful sensing devices or require additional elements to integrate the sensors. However, the majority of IoT applications promote less-constrained sensing devices for cost-effective and large-scale IoT systems. In our architecture, constrained sensing devices are direct actors on the blockchain without relying on another system component.

- **Permissionless blockchain networks.** Current blockchain-based IoT systems favor permissioned blockchain implementations. However, the most secure approach to develop blockchain applications is on top of a permissionless network. The proposed architecture utilizes a permissionless network. On the one hand, a majority of honest peers ensures the integrity of the sensed data. On the other hand, a majority of honest peers avoid dishonest peers to compromise the integrity of the entire system [28].

- **Smart contracts as an IoT application platform.** Currently, smart contracts are mainly used for financial applications or providing basic functionalities such as asset registers. In the proposed architecture, smart contracts define complex business logic that autonomously enforces agreements between untrusted actors based on the trusted values coming from the ubiquitous IoT devices.

1.4 Structure of the thesis

This chapter presented the context of the research, the proposal, and its novel contributions. The rest of the thesis is organized into three parts as follows.

Part I provides a detailed context of blockchain technology and the Internet of Things. Chapter 2 presents the main concepts of blockchain, working principles, taxonomy, decentralized consensus, and smart contracts. Chapter 3 reviews the state of the art on blockchain and IoT, including current IoT architectures, integration schemes for blockchain and IoT, and smart contracts as an interaction platform. The chapter concludes with a gap analysis on blockchain-based architectures for the IoT.

Part II describes novel blockchain-based decentralized IoT applications. Chapter 4 presents a novel blockchain-based architecture that considers constrained IoT devices. The architecture is conceptualized as a layered-framework to develop new types of IoT applications and evaluated with two case studies in the agricultural domain. Chapter 5 introduces a blockchain-based IoT system for traceability in agri-food applications. Chapter 6 describes a blockchain-based IoT system to foster sustainable agriculture practices by incentivizing virtuous behaviors in water management.

Part III presents exploratory analyses on two challenges identified on novel blockchain-based applications. Chapter 7 analyses the monetary costs associated with blockchain infrastructure to support the application. Chapter 8 evaluates decentralized storage to compensate for the current limitations of blockchain implementations to store large amounts of data.

Chapter 9 summarize the thesis and discusses possible future works. Finally, Appendix A lists all abbreviations used in the thesis.

Part I: Background on blockchain technology and IoT

Chapter 2

A primer of blockchain technology

This chapter contains text taken from the published work:

- *Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., & Rehmani, M. H. (2018).Applications of blockchains in the Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 21(2), 1676-1717.*

Blockchain technology combines data structures, incentive mechanisms, and cryptography techniques to maintain a distributed database on a peer-to-peer network [37, 38, 39]. This database stores the information by using logical blocks linked to each other, creating a chained data structure. All the peers (i.e., computers in the network) maintain identical copies of the database (i.e., the blockchain). New blocks are added to the blockchain using a decentralized consensus algorithm. The consensus algorithm defines how to validate new blocks and reach a unique global state that is agreed upon by all the network peers. Blockchain is also called a *distributed ledger* as the information stored represents transactions, i.e., interactions occurring within users of the system [15].

Using blockchain technology, two mutually unknown actors (i.e., users of the system) can interact without the need for a third-party intermediary. This trustless environment is enabled by a unique combination of elements (e.g., incentives, cryptography, peer-to-peer network) inherent to a system using blockchain technology. Cryptocurrencies are the first use case of a blockchain-based system; however, the technology is increasingly attracting interest from several other domains and applications [12, 15].

This chapter introduces the working principles and essential concepts required to understand a blockchain system. First, we describe the different components (i.e., transactions, blocks, and signatures) and their role in enabling decentralized trust. Section 2.2 presents a taxonomy for the different types of blockchains using two categories: public and private. Section 2.3 describes the most used decentralized consensus algorithms used for each type of blockchains to coordinate the peer-to-peer network. In Section 2.4, we introduce the concept of smart contracts as a tool for programming blockchain-systems. We conclude the chapter by summarizing the inherent salient features of blockchain technology.

2.1 Blockchain working principles

A blockchain, as described on Bitcoin [40], is a distributed timestamped data-structure built on blocks containing details of transactions that have occurred within the network. Transactions represent the interactions among actors and hold information regarding value transfer or data exchange. Blocks are data structures that aggregate transactions and are divided into two parts, .i.e., the header and the body [40]. The body stores the transactions. The header contains several fields, including a timestamp, an identifier, and the identifier of the previous block. The blocks are linked, creating a chain of blocks (i.e., the blockchain), as shown in Figure 2.1. The blockchain is an append-only data structure. Blocks can not be updated or deleted after adding them to the blockchain, and the first block in the chain is called genesis block.

The block identifier is the cryptographic hash of its content, creating an integrity check for the entire block [12]. Linking the blocks helps to preserve

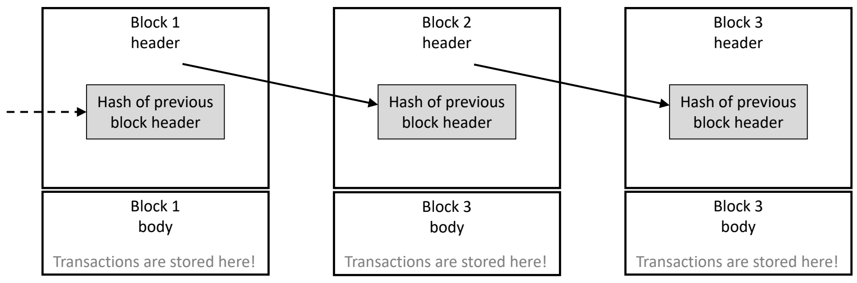


Figure 2.1: Blockchain data structure.

the integrity of all the information in the blockchain. An attack that alters the content of a block changes its identifier, making the block invalid. This change triggers a domino effect, invalidating the following blocks. Therefore, to successfully modify a single block, the attack must alter all the subsequent blocks as well. Moreover, the attack must compromise the majority of nodes in the network for the peers to reach a consensus on this altered blockchain. If the blockchain network comprises a considerable number of nodes, this attack is complex to perform and requires a substantial amount of computational power, making the blockchain practically immutable [28].

The header of the block also contains the Merkle tree root for all the transactions stored [41]. All transactions have a unique transaction ID that is the cryptographic hash of the data in the transaction. The transaction IDs are hashed together in pairs, and a hash tree is built within the block as seen in Figure 2.2. As part of the consensus algorithm, transactions are verified each time that a new block is broadcast on the network. The Merkle tree significantly reduces the time to verify a transaction, as the verification can be done using the Merkle tree branch containing the transaction. A tampered transaction would produce altered hashes within its branch and would be detected with a low computational effort.

Verified transactions are grouped in a new block that is broadcast from one node to the rest of the network, and each peer appends it to the local copy of the blockchain after validation. Appending a new block implies the

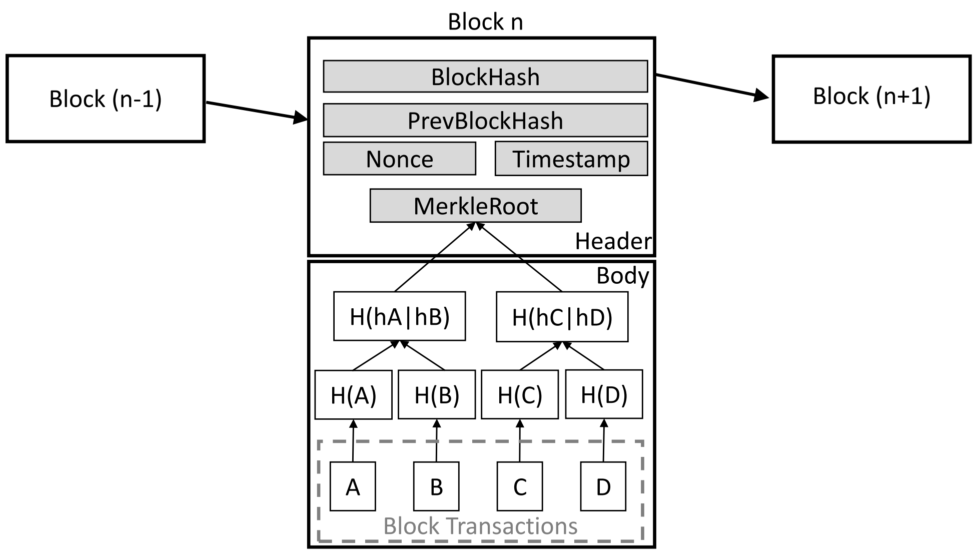


Figure 2.2: Transactions in a Merkle Tree.

agreement of its contents, updating the agreed state of the blockchain system [15]. Since the blockchain replicates in each peer, the entire system has a fault-tolerant architecture, i.e., if one peer fails, any other peers can replace it [37]. Moreover, since all the information is always accessible by any peer, blockchain is considered fully auditable [12].

Having the blockchain replicated in all the peers provides several benefits; however, it also requires handling some particular scenarios [15]. For instance, when multiple nodes in the blockchain network produce valid blocks simultaneously, the blockchain can fork. In this case, maintaining a single accepted state of the blockchain becomes an issue for the system. Typically, blockchains resolve this issue by considering the longest fork as canon and discarding the blocks in the other forks [40, 42].

2.1.1 Transactions and digital signatures

Transactions represent the interactions among the actors of a blockchain-based system. Each actor has a unique address associated with a pair of cryptographic keys. Actors use the keys (i) to sign their transactions to certify their origin and (ii) to validate the integrity and the origin of the transaction they receive. Thus, all the interactions in blockchain are made through digitally signed transactions.

In Bitcoin, transactions are signed using the Elliptic Curve Digital Signature Algorithm ECDSA[43], an asymmetric cryptographic algorithm that uses a public/private key pair. The public key is shared to all the actor of the system while the private keys must be kept secret. Signing a transaction with the private key produces a unique signature composed pair of numbers (r, s). Anyone in possession of the public key can easily have an unequivocal confirmation that the signature was generated by the matching private key, as shown in Figure 2.3.

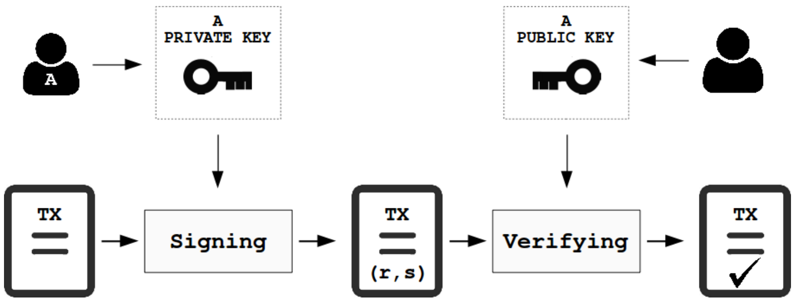


Figure 2.3: Public key signature process.

Transactions can be considered as an arbitrary data array Tx. The fields of the array vary with each implementation; however, most of them share common fields. Based on [40, 42, 44, 45, 46], a general transaction structure for Tx can be considered as

$$Tx = \{T_{header}, T_{destination}, T_{data}, T_{value}, T_{signature}, T_{extra}\} \tag{2.1}$$

Where T_{header} is the header of the transaction typically including a unique identifier. $T_{destination}$ is the identifier of the transaction recipient normally derived from the public key. T_{data} is the information that is exchange between the interacting parties. T_{value} represent value transfer among the parties (i.e., cryptocurrency exchange). T_{extra} represents all the additional information pertinent to each blockchain implementation. Finally, $T_{signature}$ is the digital signature of T, represented by (r, s) and is normally appended to the end. From the signature is possible to obtain the public key of the transaction sender and thus, the sender identity.

2.2 Taxonomy of blockchain networks

Bitcoin introduced blockchain technology for storing the financial transactions of a decentralized payment scheme using a public peer-to-peer network [40]. After this first use case, several new implementations appeared, introducing modifications to the original protocol to provide additional features. One example are blockchain systems such as Hyperledger [47], Stellar [48], and Multichain[49] that propose permissioned peer-to-peer networks suitable for use cases where a certain level of trust among the peers exists. This assumption softens some security concerns and increases the overall performance of the system (e.g., latency, transaction throughput)[50].

Another example is IOTA [13] that proposes a directed acyclic graph approach (DAG) instead of a single sequence of blocks to achieve the higher scalability and data throughput required by some use cases. This proposal is ledger built and maintained by peers, but without using a blockchain (*i.e.*, a sequence of blocks forming a chain) as the underlying data structure. In the literature, these approaches (i.e., without using a chain of blocks) belong to the Distributed Ledger Technology DLT class, of which blockchain-based approaches only represent a subset. For a more formal taxonomy and description, the interested reader is referred to [11, 18, 51] and the references therein.

Currently, there is no standard taxonomy for the blockchain-based approaches of the DLT. For instance, authors in [52], [13] and [33] classify blockchain by generations, being Bitcoin the first generation. However, there is not a clear separation between generations or how many generations exist. In this thesis, we consider the two-type taxonomy agreed by several researchers and practitioners, i.e., public and private blockchain [53, 54, 55, 56].

2.2.1 Public/permissionless blockchains

Public blockchains rely on unknown users willing to operate the network peers in exchange for incentives paid in cryptocurrencies [56]. The peers in the public blockchain can choose to validate new blocks or merely issue transactions. Each transaction has a processing fee that serves as an incentive to the peers for creating and publishing new blocks onto the blockchain[15]. Since the users operating the nodes are unknown, malicious actions can occur. To mitigate dishonest behaviors, publishing new blocks typically requires spending computational resources by solving a cryptographic puzzle [40] or staking a predefined cryptocurrency amount [57]. These requirements aim to discourage dishonest users from wrongly validating blocks as this malicious behavior impacts the user resources [29]. Examples of public blockchain are Bitcoin [40], Ethereum [42], and Litecoin [29], to name a few.

2.2.2 Private/permissioned blockchains

Private blockchain are *permissioned*, and the infrastructure supporting this network is managed by a stakeholder (or a group of stakeholders). The stakeholders can define different roles and permissions for the users, restricting their access to the system. Examples of these types of blockchain include Hyperledger [47], or Multichain [49]. Private blockchains do not require cryptocurrency to operate, and there are no processing fees included in the transactions. Blocks are published by authorized nodes, reducing the security constraints and thus, increasing the performance of the system [50]. However, a private blockchain is not as tamper-resistant as a public blockchain, and the organization may choose to roll back the blockchain to any point in the past. Even if a private blockchain provide auditability and offers better performance (e.g., lower latency, higher transaction throughput) is not entirely decentralized or censorship-resistant as a public blockchain [58].

2.3 Decentralized consensus algorithms

Both types of blockchain systems use consensus algorithms to coordinate the peers of the network. In any blockchain, a system based on "state machine replication", consensus protocols ensure all replicas of the shared state are synchronized and in agreement at any given point in time. Earlier works on consensus protocols [59] involved cryptography and partial concurrency [60], and precursor designs and proposals of digital currency [61, 62] were the building blocks that went into developing decentralized consensus algorithms used in blockchain networks.

Core principles applied in designing consensus algorithms are safety, liveness, and fault tolerance. Safety is the extent to which a system can tolerate failures, say in an (n, f) fault-tolerant system, where n is the total number of processes, the system should be able to tolerate at most f faults. Safety is the ability to mitigate corrupted or out-of-order messages so that all non-faulty nodes reach consensus. The liveness of a fault-tolerant system means that despite the presence of f faults, all correctly participating nodes should be able to move forward with their distributed processes.

Maintaining fault tolerance in a consensus protocol becomes difficult when nodes can fail or start acting dishonestly. This fault is termed the "Byzantine Generals Problem" [63]. In a distributed system running a consensus protocol, a node can fall under a Byzantine fault as a result of software bugs or by being compromised. Byzantine faults occur when a node sends false messages and misleads the other nodes participating in the consensus protocol. Several algorithms are proposed in the literature, as surveyed by authors in [64]. These algorithms approach Byzantine faults on many types of distributed systems by making assumptions on different parameters such as the specific use-cases, the network performance, or the type of node failure.

In this chapter, we discuss a few decentralized consensus algorithms relevant to both permissioned and permissionless blockchains. The main goal of these consensus algorithms is the agreement on which new information is added to the blockchain. Authors on [65], [58], and [50] contain exhaustive details on consensus algorithms for both types of blockchain networks.

2.3.1 Consensus on permissionless blockchains

Reaching consensus using votes in a permissionless blockchain is problematic. Participants can use multiple accounts on the blockchain and launch a Sybil attack [66] to drive decisions in their favor. Therefore, consensus algorithms in permissionless blockchain are based on a lottery-based selection of a single node (i.e., peer in the network) that will add a new block. On a public blockchain, adding a new block needs to be expensive, so the resources of a single peer are insufficient to bias the consensus of the entire network.

Proof of Work (PoW)

The first public blockchain consensus protocol was the Proof of Work as described by Bitcoin [40]. Any node can add a block to the blockchain by showing that it has performed a computationally expensive amount of work. Adding new blocks with PoW is called mining, and miners (i.e., nodes creating blocks) engage in a race to solve a cryptographic puzzle. The puzzle consists of finding a nonce that, when hashed with the hash of a block, produces a resultant smaller than a predefined threshold. The proportional inverse of this threshold is called the *difficulty level*, which is stored in the block header, and gets adjusted with the increasing number of participants, to maintain an average block processing time [15, 29]. The calculated nonce is the proof of the spent resources a miner does. The miner adds the nonce to the block header and broadcasts the block to the network. All participating nodes need to verify the block published by the miner. Subsequently, the miner claims the processing fees associated with the transactions stored within the block as a reward for mining [40]. In PoW consensus, the computationally expensive block creation and transaction fees secure the network against DDos attacks and false block creation [40, 29].

PoW also defines how to handle the case of two nodes publishing a block almost concurrently [62]. Consider the case where after a block n, a node in Australia mines a valid block $(n + 1)$, and at the same time, a node in Sweden mines another valid block $(n + 1)'$. This event creates a temporary fork, i.e., two groups of peers having a different version of the blockchain. After the block n, one fork has the block $(n+1)$ and the other has the block $(n + 1)'$. Moreover, both forks continue adding blocks to their version of the blockchain. In this case, the consensus algorithm defines that the fork with the most work committed to it (i.e., more blocks) is hence canon, and the other fork is orphaned [40, 46].

PoW consensus is vulnerable when an attacker takes control of 51% of the processing power of the network [67][68]. Therefore, PoW provides fault tolerance as long as the total computational power is $n \geq 2f+1$ where f is the computational power occupied by a single malicious user [29]. To overcome this type of attack, and considering the append-only nature of blockchain, Pow blockchain such as Bitcoin and Ethereum introduced the concept of delayed finality for a block. Before the finality, the blockchain can be rolled back to a previous block in the event of a 51% attack. After reaching finality, a block is irreversible. Ethereum and Bitcoin consider a block finalized after six confirmations, i.e., after six new blocks are added [29].

Proof of Stake (PoS)

The Proof of Stake algorithm aims at reducing the ever-increasing electricity consumption of mining in PoW blockchain networks [69]. PoS aims to stake peers' economic share in the network [70]. Here, the term miner is replaced with validators, and similar to the PoW, one of the validators is chosen to publish (i.e., add) a new block onto the blockchain. The difference lies in how the validator is chosen. In PoS, a validator is selected in a pseudo-random fashion, with the probability of being selected proportional to the validator's share in the network [71] [72]. NaivePoSconsensus mechanisms are prone to attacks like the "nothing at stake" attack and require further considerations to be consensus-safe [73]. Block finality is faster compared to PoW blockchains since there is no computational puzzle solving involved in choosing the validator. Ethereum's Casper [57] is currently one of the most advanced implementations of the PoS and punishes malicious nodes by subtracting their funds in case of dishonest behavior.

Proof of X

Other alternative consensus algorithms for public blockchain are classified as "Proof of X". In [50], the author presents an exhaustive study of these algorithms. However, most of these algorithms are suited for private blockchains, and those proposed for public scenarios, lack the maturity of PoW and PoS. For instance, *Proof of activity* (PoA) [74] was proposed as an alternative to Bitcoin mining and combines aspects of the PoW and PoS. Computational puzzle-solving in PoA only involves finding a nonce against the block header without considering the transactions in the block. A random group of validators is chosen to vote the validity of the mined block header. Similar to PoS, the probability of the validators being chosen is proportional to their share in the network. Transaction fees are split between the miner and validators. Concerns towards PoA include the high computational power requirements and the fact that a naive implementation can be prone to nothing at stake attacks [1]

2.3.2 Consensus on permissioned blockchains

In permissioned blockchain deployments, only a limited number of known participants have a copy of the entire blockchain [75]. Maintaining consensus, therefore, is straightforward and does not require costly proofs for publishing a new block. Since participants are known, there is no risk of a Sybil attack, therefore voting mechanisms are used to achieve consensus. Hence permissioned blockchains have a much higher performance than permissionless blockchains in terms of latency, transaction throughput, and power consumption [1, 50].

Practical Byzantine Fault Tolerance (PBFT)

The Practical Byzantine Fault Tolerance (PBFT) algorithm [76] involves multiple rounds of voting by all nodes of the network to commit a state change (i.e., add a new block). The PBFTalgorithm includes an optimized encrypted message exchange for making global voting more practical. This algorithm requires $n \geq 3f+1$ nodes to tolerate f failing nodes. In PBFTconsensus, one node is chosen to be the leader, who assembles a set of ordered transactions into a block. The validating peers in the network calculate a hash of the block and broadcast it. Validating peers observe the hashes they receive from the rest of the network, which can be seen as votes over multiple rounds. If 2/3 votes are in favor of the candidate block, the peers add it to their copy of the blockchain. PBFTconsensus provides high throughput and low latency in validating transactions. However, the overhead incurred by broadcasting blocks and votes in PBFTconsensus set a maximun of tens of validators.

Hyperledger Fabric uses a variation of PBFTcalled Sieve [77], designed to perform consensus while executing a non-deterministic chaincode.

Tendermint

Tendermint [45] is a Byzantine Fault Tolerant consensus algorithm, that, similarly to PBFT, provides an $n \geq 3f+1$ fault tolerance. Tendermint usesPoSin combination with principles of PBFTto provide security, high throughput, and low block processing times of about 1-3 seconds. Tendermint uses the lottery-based properties ofPoSto select the leader node with probability proportional to the node share in the network. Tendermint performs multiple rounds of voting to reach a consensus on a new block and requires a supermajority or 2/3 of its validators to maintain 100% uptime. If more than 1/3 go offline, the network may stop progressing and lose liveness. Assuming that less than a third of all validators are faulty, Tendermint provides a safety guarantee that no conflicting blocks are created and no forks appear in the blockchain. Transaction finality in Tendermint is approximately 1 second, and the protocol is compatible with public and private chains. However, it has a lower level of scalability than PoW orPoS[13].

Federated BFT (FBFT)

Blockchain implementations in Ripple [44] and Stellar [48] extended the traditional Byzantine Fault Tolerance to scenarios involving a consortium of nodes (i.e., subnetworks). Ripple consensus begins with a unique node list (UNL) of active validator in the network. Each node has a UNL with 100+ nodes in it. Each nodes UNL has to overlap by at least 40% with the UNLs of other nodes. Ripple carries out multiple rounds of voting, where nodes assemble transactions into candidate blocks and broadcast them to the nodes in their UNL. Nodes then broadcast votes on each candidate block. Each round of voting helps nodes refine their candidate block, and a new block is added to the ledger once it receives a supermajority vote of 80%. Even though Ripple carries out multiple rounds of votes, it provides a fault tolerance of $n \geq 5f + 1$ [44]. The consensus in the entire network is based on consensus within subnetworks, so Ripple allows open-ended participation of users, market entities, and gateways to other subnetworks [44].

Stellar introduces the idea of quorums in blockchain networks, where a quorum is a set of nodes used to reach consensus. A node in such a network can be part of multiple quorum slices, where each quorum slice securely reaches consensus through voting. Since the quorums and quorum slices are allowed to intersect, Stellar allows open participation of nodes in different subnetworks within the main Stellar network. Stellar opts for safety over liveness, with malicious behavior, the blockchain does not progress till the malicious behavior is resolved. Stellar provides flexible trust and low latency since it is computationally inexpensive, and quorums contain a limited number of nodes that share vote messages [48].

Proof of Elapsed Time (PoET)

Among the Proof of X protocols for permissionless networks, one example is Proof of Ellapsed Time by Hyperledger Sawtooth [78]. PoET runs in a Trusted Execution Environment (TEE), such as Intel Software Guard Extensions (SGX) [79]. A trusted voting model built on SGX helps electing a validator for publishing a new block. PoET is lottery-based without expensive computational puzzle solving or monetary stake. Nodes in the PoET network request a wait time from a trusted function within the SGX. The validator with the shortest wait time is selected as the leader. Another trusted function attests if the validator waited an allotted amount of time before publishing a new block. Even if the algorithm meets the prerequisites of a viable lottery-based consensus algorithm, its limitation is in the use of specialized hardware.

2.3.3 Comparison of permissionless and permissioned consensus algorithms

Table 2.1 summarizes the differences and similarities between public and private blockchain, including the consensus algorithms and examples for each blockchain type. The decision of the blockchain type depends on the requirements (e.g., permissions and performance) of each use-case.

Permissionless blockchains are open systems designed to accommodate several unknown and widely disperse actors[28]. However, permissionless blockchain must have slow block creation speeds, taking into account the propagation speeds of nodes within the network. Permissionless blockchain work with unknown users, which requires additional security measurements to prevent abuses and guarantee the integrity of the information[15]. These measurements increase the system latency, providing a limited throughput in terms of transactions per second. At the time of writing, Bitcoin process around five transactions per second, while Ethereum process close to 20 per second [29].

Permissioned blockchains have much lower latency but suffer from a severe scalability issue. The networking overhead incurred from voting mechanisms limits permissioned blockchains to scale to only hundreds of nodes[50], as shown in Figure 2.4. Despite the decentralized architecture, permissioned blockchains are not entirely decentralized in terms of administration as peers have different levels of permissions [13].

For early adopters of this new technology, private blockchains present a harmless transition from traditional centralized systems [13]. Access control and higher throughput enable cross-organization business processes [52], which has proven beneficial on domains such as Manufacturing [23], Smart Homes[80], and Communications[20], to name a few.

However, an increasing number of blockchain-based systems are thriving

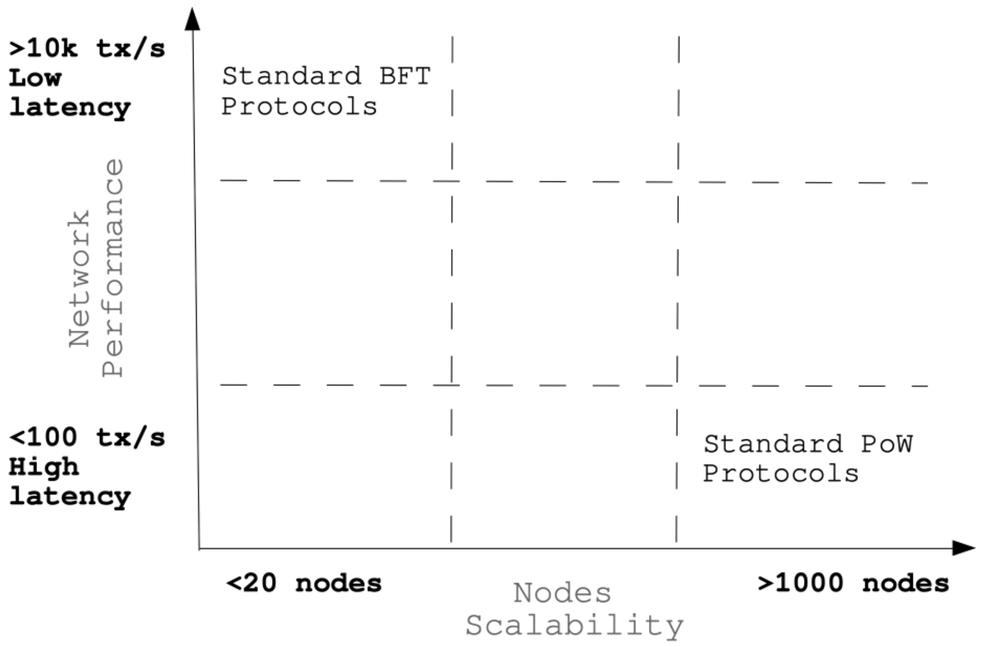


Figure 2.4: Performance and scalablity of different consensus (adapted from [1]).

from permissionless networks. Applications in Energy Trading [81], Utilities management [82, 83], Land ownership [84] [85], Remote Sensing [84] are a few examples embracing the unique benefits offered by public blockchain networks and the possibility to securely interact with unknown actors.

2.4 Smart contracts

Smart contracts are scripts stored in the blockchain, roughly similar to stored procedures in a database [12]. Bitcoin offers limited scripting capabilities through a restricted set of instructions that allow, for example, multi-signature transactions and escrow agreements [15]. To overcome the scripting limitation, Ethereum [42] extended Bitcoin to include a Turing-Complete programming language, allowing the development of more complex software applications called smart contracts [46]. The term "smart contract" was coined by N. Szabo with the objective of *"securing relationships on public network"* [86]. In blockchain networks, smart contracts perform the function of carrying out

Table 2.1: Comparison of public and private blockchains.

	Public Blockchain	**Private Blockchain**
Participation in Consensus	All nodes	Selected Nodes
Access	Public read/write	Can be restricted
Identity	Pseudo-anonymous	Approved participants
Immutability	Yes	Partial
Transaction Processing Speed	Slow	Fast
Permissionless	Yes	No
Consensus Algorithms	PoW [29], PoS [57]	PBFT [77], FBFT [48] [77] Tendermint [45] PoET [78]
Implementations	Bitcoin[40], Ethereum[46] Litecoin [29]	Hyperledger Fabric[77], Hyperledger Sawtooth [78] Stellar [48] Multichain [49]

transactions in a predetermined fashion agreed by interacting actors.

A smart contract is a distributed software stored in the blockchain that implements functions callable by any actor using transactions. Once on the blockchain, the code in the smart-contract is immutable [13]. Every time a smart contract is called, the software runs deterministically in all the peers at the same time. The exact execution output enforces the agreement among actors without the need for any third-party validator. The trust in the valid execution of the code arises from the trust in the underlying blockchain system[52].

Smart contracts have an internal state composed of storage and a balance. The storage is a private data repository, and the balance allows the smart contract to generate transactions [12]. Smart contracts can process information from the blockchain, create new transactions, and make decisions on behalf of their creators [87]. Moreover, smart contracts can instantiate and invoke other smart contracts [52]. Thus, smart contracts can be seen as "autonomous agents" as they have their own identity and are considered an actor on the blockchain system [88].

While Bitcoin is considered the reference implementation for the blockchain protocol, the reference for smart contracts is Ethereum [89]. Permissionless blockchains have taken Ethereum as the model for implementing smart contract functionalities or provide direct compatibility with Ethereum smart contracts [47]. Thus, smart contracts provide a general-purpose programmable infrastructure [33] to implement different types of processes [52] and create new types of decentralized applications.

2.5 Salient features of blockchain technology

Blockchain provides a decentralized, trustless environment where mutually untrusted actors can interact without a validating intermediary [13]. Applications that previously required a trusted intermediary can now work in a decentralized manner [12]. Blockchain enables this secure environment by a unique combination of inherent properties and working principles. According to current literature, the following are the most important features of blockchain technology enabling decentralized trust.

Transparency On a public blockchain, there are no restrictions to access the system, and all transactions are visible to any system actors, creating a real transparent environment. [25, 55, 90, 28, 12].

Auditability Any actor can download a copy of the blockchain, making the information available for everyone to query, verify, and audit. The use of digital signatures adds non-reputability to this public verification of the blockchain. [37, 32, 12, 25, 55].

Immutability Once a transaction reaches finality, it becomes immutable. Transactions can not be modified or deleted without a substantial amount of computing. [13, 91, 25, 90, 37].

Integrity The cryptographic hashes of blocks and transactions combined with asymmetric cryptography protect from unauthorized changes on the information. [13, 91, 25, 90, 33, 37].

Openness On the one hand, blockchain implementations typically open-source. On the other hand, public blockchains are open to any user to join the system. [90, 13, 12, 87].

Scalability Distributed systems can dynamically adjust their resources. Moreover, they remove the bottleneck imposed by the presence of centralized services. [23, 90, 87, 13, 12].

Neutrality New information added to the blockchain is agreed upon by the peers via decentralized consensus making the system censorship-resistant. On a public blockchain, all peers have the same rights, creating a system based on distributed power. [87, 12, 37, 33].

Fault-tolerance The decentralized nature of blockchain eliminates the existence of a single point of failure in the architecture. All blockchain peers contain identical replicas of the ledger records, adding inherent redundancy. [12, 37, 55, 13].

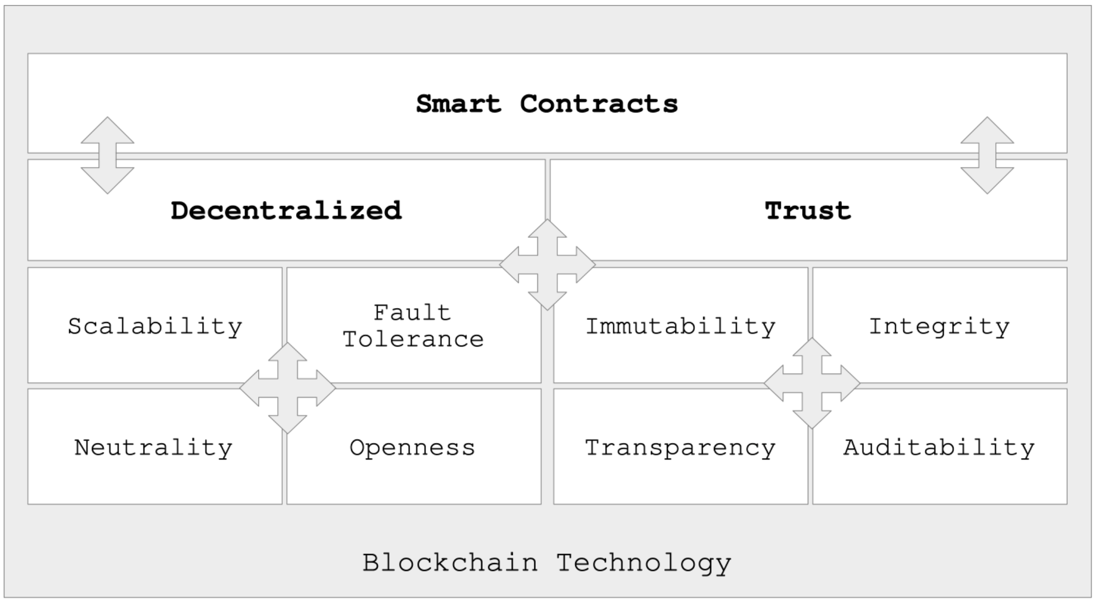


Figure 2.5: Main features of blockchain technology.

These features interconnect, as shown in Figure 2.5, to enable decentralized trust among unknown actors. Moreover, smart contracts provide a platform to develop new types of decentralized applications. Cryptocurrencies are the first application of the record-keeping and trustless environment of blockchains. Despite the current limitations (e.g., latency, transaction throughput), the technology holds the potential to support other types of applications that required secure interactions between unknown actors.

Chapter 3

State of the Art in integrating blockchain and IoT

This chapter contains text taken from the published work:

- *Pincheira, M., Vecchio, M., Giaffreda, R., & Kanhere, S. S. (2021).Cost-effective IoT devices as trustworthy data sources for a blockchain-based water management system in precision agriculture. Computers and Electronics in Agriculture, 180, 105889.*
- *Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., & Rehmani, M. H. (2018).Applications of blockchains in the Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 21(2), 1676-1717.*

The Internet of Things (IoT) is a concept that describes an interconnected system where billions of everyday objects act as sensing devices creating ubiquitous applications [7]. The main objective of IoT is to connect a heterogeneous group of devices. These devices sense, collect, analyze, and store large amounts of data to improve the awareness of the physical environment [6]. IoT is a disruptive technology solving an increasing number of present-day issues in domains such as Smart Cities, Agriculture, Health, and Industry, to name a few [7].

IoT systems are based on computationally-constrained devices capable of sensing the surrounding environment and interact with several other IoT devices creating complex systems [4]. Currently, IoT systems are based on centralized architectures, where an intermediary enables the trust for the device interactions. Centralized architectures simplify the design and the deployment of IoT systems; however, they also introduce challenges and risks regarding security and privacy [92]. Furthermore, an architecture that requires an intermediary for trusted interactions between devices will not scale at the same exponential growth of IoT applications [13].

In recent years there has been growing interest in blockchain as a possible solution to address these concerns. As described in Section 2.5, blockchain provides a trustless environment enabling interactions between mutually untrusted actors without a validating intermediary. Current literature proposes different integration schemes for blockchain and IoT. These schemes define distinct roles for the various types of IoT devices according to their capabilities. However, far too little attention has been paid to how blockchain is integrated with the core of IoT systems, i.e., the constrained sensing devices.

In this chapter, we present the state-of-the-art regarding the integration of blockchain technology in the Internet of Things domain. First, we describe modern IoT architectures using a component-oriented approach and a service-oriented approach. We identify the requirements and current challenges of the IoT systems and highlight the benefits that blockchain offers to tackle these challenges. Section 3.2 reviews the most relevant blockchain and IoT integration schemes found in current literature. The schemes are classified into three categories, considering the architecture of IoT systems. The chapter concludes with a gap analysis for the integration of blockchain and IoT, focusing on the constrained sensing devices as direct actors in a blockchain system. This gap presents the motivation and requirements for the integration framework proposed in this thesis.

3.1 IoT layered architectures

Modern IoT systems consist of several components, each of them with unique constraints and capabilities [4]. These components can be organized using a layered-architecture working as a reference for IoT systems. Since the coinage of the term *“Internet of Things”* in 1999, the reference architecture has gone through several designs that include up to 7 different layers, as proposed by Cisco Systems Inc[1].

The architecture defines each layer and the components within and there are several ways to approach an architectural design [9]. Typically, IoT architectures follow a component-oriented approach, a service-oriented approach, or a mixture of both. A component-oriented approach focuses on the devices, their requirements, and constraints from a functional perspective [6]. A service-oriented approach focus on the expected functionalities and requirements from an application perspective [2].

Following a component-oriented approach, the trend of the literature is a 3-layer architecture [6, 19, 93, 9] , as shown in Figure 3.1. The pyramidal shape represents the scale (i.e., number of devices) of each of the layers, and the typical data flow starts at the bottom layer and moves through the top.

On the bottom, the **Device Layer** is the core of IoT systems, comprising the sensing devices that acquire data from the physical world. These devices are small and cost-effective [7, 3], favoring low-power energy consumption modes against capabilities such as processing, storage, or connectivity [5, 3]. The Device Layer is also called the perception layer or physically layer [6, 5]. In the middle, the **Edge Layer** provides connectivity and additional computational power to the sensing devices. This layer is sometimes also referred to as gateway layer or network layer [6]) and includes less-constrained devices in terms of computing capabilities and communications.

[1]https://www.cisco.com/

At the top of the architecture, the **Cloud Layer** represents powerful servers responsible for processing and storing the data [6]. IoT applications are typically hosted at this layer, giving the users an entry point to the entire IoT system.

Figure 3.1 shows the estimated scale of each layer (i.e., number of devices). Following this architecture, the typical data flow starts at the Device Layer sensing environmental parameters. The sensed-data is sent Edge Layer that provides the connectivity required to reach the Cloud Layer. Once the sensed-data reaches the top layer, it is processed and stored as required by the IoT application.

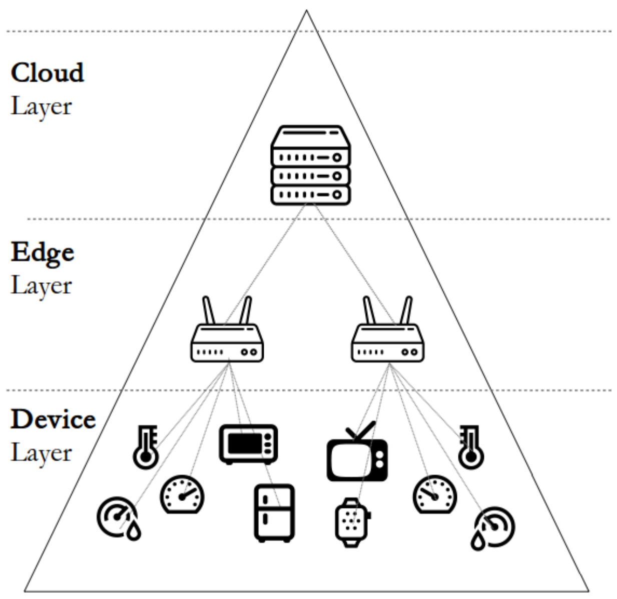


Figure 3.1: Component-Oriented Architecture for modern IoT systems.

In recent years, the Service-Oriented Architecture (SoA) [94] approach is attracting considerable interest as an alternative to describe IoT systems [6, 4]. The different components of IoT systems work together as networked devices, presenting the IoT applications as services on service-oriented architecture [2]. As surveyed on [2], the principal focus for SoA in IoT is security [95, 2] and the architecture is usually arranged into three layers as shown in Figure 3.2.

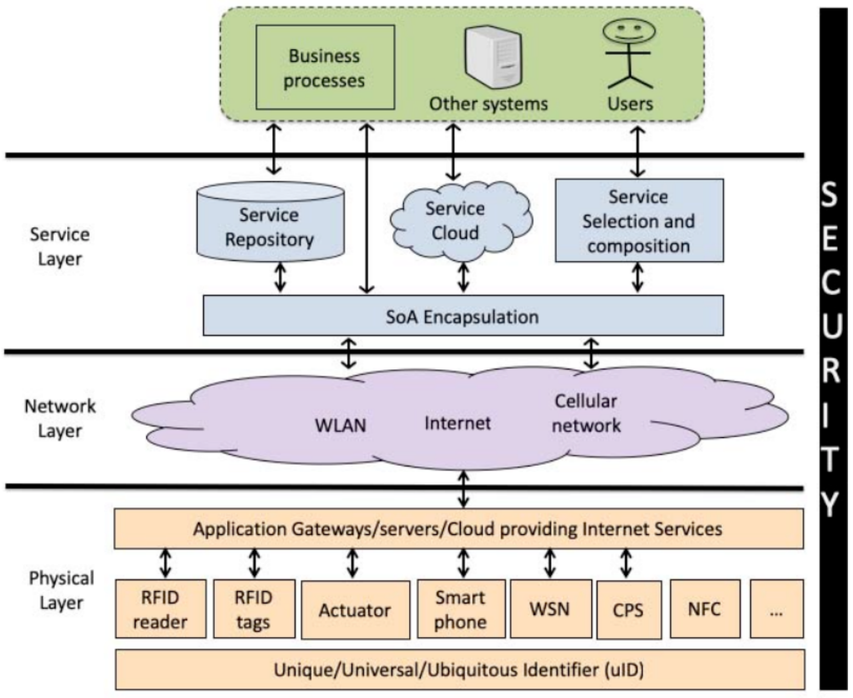


Figure 3.2: Service-Oriented Architecture for IoT systems (extracted from [2])

On the SoA architecture, the **Physical Layer** groups both *Device Layer* and *Edge Layer* of the previous component-oriented architecture. From a SoA perspective, both components are responsible for interconnecting physical sensing entities to internet services. One of the key elements that support the entire architecture is a unique device identifier [2], as shown in Figure 3.2. The **Network Layer** supports the Internet as the communication channel for connecting the sensing devices, following a network-oriented approach. The objective of this layer is to enable information sharing between IoT devices and traditional IT systems. At the top, the **Service Layer** drives towards the seamless interoperation between IoT and Internet services [2], following a semantic-oriented approach. At this layer, the identification of traditional application requirements semantically defines an IoT service. Among the requirements, one key element is that the IoT service needs to be accessible by standard interfaces to support the demands from users and applications [2].

To summarize, both architectures aim to describe the diverse elements composing modern IoT systems, their capabilities, and their functionalities with different levels of abstractions. The first approach, as shown in Figure 3.1, is component-oriented. The approach focuses on describing the capabilities and constraints of the devices. The second approach is service-oriented, as shown in Figure 3.2. This approach focuses on describing the functionalities and expected behaviors of each IoT component. Thus, both architectures provide different details to describe the complexities behind modern IoT systems and should be considered on an blockchain and IoT integration framework.

3.1.1 Requirements of IoT systems

The requirements of IoT systems vary considerably within domains and use cases (i.e., IoT applications). For instance, applications in Industrial environments [4] favor high connectivity with near real-time response. These requirements demand IoT devices (at the device layer) with a high computational capacity and fast network communications, increasing the energy requirements. Additionally, these environments impose higher restrictions in terms of access control and permissions at the cloud layer.

Another example is smart-cities[3], where one of the main requirements is mobility. Sensing devices should be smaller and portable and demand constant connectivity even while changing their location. Applications will favor low-power consumption devices to achieve more lasting energy autonomy, lowering communications and processing speeds. Additionally, to compensate for periods without network access, sensing devices favor storage and memory versus processing power. Similarly, Health Care [96] also values mobility and pervasiveness of the IoT devices but introduces additional security and privacy requirements. These requirements demand strict security and privacy policies that need to be defined and guaranteed in all architecture layers.

Finally, Agriculture is one of the most restricting IoT domains. Precision agriculture applications require power efficiency and low-cost devices to guarantee large-scale, cost-effective, and long-term installations [21]. Moreover, agriculture applications in this domain require energy-efficient communication networks [97], favoring coverage over speed.

A complete analysis of all possible application requirements is beyond the scope of this thesis. We focused our research on a subset of system requirements with a higher impact on the sensing devices. Based on the surveyed literature, we define the following as a list of traditional constraints that need to be addressed in IoT systems.

- ***Connectivity***: Communications are the key to the complex interconnections inside of IoT systems. Connectivity requirements (i.e., latency, bandwidth, coverage, and mobility) directly define each component of the entire IoT system. For instance, the cloud layer is typically based on IP Internet access, while the device layer might use RF communications. This scenario forces the edge layer to implement different protocols to provide the expected connectivity. Moreover, communications are typically one of the most power-consuming tasks of the IoT device work-cycle. [5, 6, 3, 98, 7, 99, 100]

- ***Computing***: The computing power (i.e., processing, memory, and available storage) is a defining requirement for the entire architecture. Sensing devices are typically resource-constrained to reduce size and cost, and microcontrollers provide the most suitable platform. If sensing devices are resource-constrained, the edge layer needs to compensate for the missing computing, storage, or communication capabilities, favoring system-on-a-chip hardware (SoC). The computing requirements also determine the services required from the provider at the cloud layer.

[5, 4, 7, 99, 101]

- ***Energy***: This constraint reflects energy requirements such as the energy source (e.g., battery, solar-powered, energy-harvesting) and the autonomy (e.g., days, months, years), particularly at the device and edge layers. Different communication protocols will have varying energy requirements where typically faster speeds require additional energy. The same applies to computing requirements as processing power increases energy consumption. However, energy is also a restriction directly imposed by the IoT application. For instance, Smart Cities applications will require mobile battery-powered devices, while Industrial applications might provide continuous energy sources for static devices. [5, 98, 4, 7, 3, 99]

- ***Scale***: This requirement reflects the size of the IoT system in terms of users and devices. IoT systems usually employ a massive number of sensing devices to achieve the pervasiveness required by IoT applications. On the one hand, reductions in hardware costs foster the number of devices at the lower layers. On the other hand, the broad adoption of IoT translates into use cases (e.g., Smart Cities) involving thousands of users at the application layer. [101, 4, 6, 7, 5]

- ***Costs***: Cost becomes a constraint when selecting the sensing devices which are the core of the IoT system. A sensing device with high computing power, fast connectivity, and extended energy autonomy is achievable but with a high monetary cost. Large-scale deployments (i.e., with a large number of devices), as applications in smart cities or agriculture, demand cost-effective devices for achieving economic feasibility. [101, 3, 7, 6, 5]

This list of requirements, while not exhaustive, provides a general overview of the typical constraints that IoT systems need to take into consideration.

Therefore, a framework that integrates blockchain into the IoT must address these requirements in all architectural layers.

3.1.2 Challenges in centralized IoT architectures

Centralized cloud-architectures are the predominant scheme for building IoT platforms [13]. In these architectures, a central server manages several tasks such as data handling, device coordination, and actors authorization [9]. Centralized services have contributed to the exponential growth of IoT; however, they have introduced several concerns regarding trust and data transparency [28]. Centralized cloud services act as a black box for IoT services and IoT users do not have control and total confidence in how the data they share will be used [13]. Moreover, these centralized servers are intermediaries present in all interactions of the actors in the system [12]. Based on the surveyed literature, the following are some of the main challenges currently affecting centralized IoT architectures.

- ***Security*** The diverse types of interconnected IoT devices provide several attack surfaces that are difficult to immunize against security threats. The entire IoT system exposed many entry-points to traditional IT attacks such as phishing, password security, message spoofing/alteration, traffic analysis, distributed denial of Service, Sybil attack, and eavesdropping, among many others[6]. The edge layer is another entry point to the system with inherent threats and risks. An attack on one gateway compromises the entire device layer, and thus, the trustworthiness of the sensed data of several IoT devices [13]. Finally, IoT devices are commonly isolated hardware solutions that, depending on their deployment conditions, are subject to tampering in ways that may be unpredictable by manufacturers. These attacks can alter how devices measure the physical world or how they share the data. Moreover, very few systems securely handle the device identity, enabling attacks such as node capture, malicious code injection, false data injection, and message replay, to name a few[6]. [6, 5, 4, 7, 101, 13, 100, 98]

- ***Availability*** The IoT represents a proliferation of always-available smart devices that continuously collect data from the environment [7]. This high availability may not always be the case in architectures involving centralized servers [100]. An attack on the centralized provider can compromise the entire application, affecting hundreds of users and thousands of devices [99]. Moreover, since cloud services are multitenant by nature, an attack on a single provider can directly reach several IoT applications, increasing the impact of such an attack. [101, 6, 4, 7, 99]

- ***Data Managemenet*** The data generated by IoT devices may offer detailed information about the context where the device exists [102]. In many domains (e.g., Health Care, Smart Home), the data collected by the IoT devices describe sensitive personal information about individuals [4]. In the current IoT service provision, users surrender their data to centralized service providers acting as isolated silos. On the one hand, the provider might collect this information without any explicit user consent [100]. On the other hand, this information can be later disclosed to third parties, withholding users of control on which data and to whom their data is given access [102]. Additionally, users lack the tools for verifying the integrity of the data in the IoT system, starting at the device and finishing at the application. [7, 102, 6, 4, 100]

- ***Potential Growth*** IoT systems are typically composed of numerous sensing devices to create ubiquitous systems. The volume of data generated by these sensing devices can be enormous and difficult to manage for elaboration, transmission, and storage[4]. IoT systems must scale to efficiently handle this massive growing volume of data[103]. However, the presence of a central service-provider can create a bottleneck limiting the performance of the entire systems[13]. [98, 5, 103, 4, 7]

- ***Interoperability*** On the one hand, IoT systems interconnect several types of devices with a plethora of hardware architectures. This heterogeneous system requires the connections between various types of networks using multiple communications technologies[4]. On the other hand, the IoT is full of standards supported by multi-national governance bodies, alliances, or organizations [104]. These standards cover different aspects of IoT, from communication technologies to architectures [101]. The uncontrolled proliferation of standards only leads to fragmentation and becomes a real barrier for the IoT adoption in multiple application domains [104]. IoT systems should avoid isolated systems based on proprietary solutions and enable data sharing and interoperability among these closes subsystems[7]. [5, 101, 4, 104, 3]

These challenges provide a general overview of the problems that IoT systems are currently facing. Centralized cloud-architectures fostered the development of the IoT domain. However, the centralized model is also responsible for several of the reviewed challenges. Therefore, a decentralized paradigm might be the key to develop new types of decentralized IoT systems.

3.1.3 Decentralization of IoT through blockchain technology

The salient features of blockchain, described in Section 2.5, can help to overcome several concerns of centralized IoT architectures. Thus, blockchain technology is an attractive solution for developing a secure decentralized architecture for IoT systems [12]. The “trustless environment" enabled by blockchain removes the need for trusted centralized entities to handle device interactions [87]. Moreover, a decentralized system can scale better to fit the exponential growth of IoT systems [13].

Security on decentralized IoT systems benefits directly from several of the intrinsic features of blockchain [23]. For instance, the cost of creating transactions (either monetary or computational) protect the network against flooding and DDoS attacks. The use of cryptography can prevent and mitigate false data injection or node capture attacks [13].

Blockchain-based IoT systems can soften concerns regarding data management by the inherent properties of auditability and transparency [90]. Regarding data corruption, data integrity is one of the core properties of the blockchain [25]. Moreover, blockchain networks store redundant replicas of records over blockchain peers preventing data loss.

IoT systems require high availability, which may not always be the case in architectures involving centralized servers. Blockchains are Byzantine fault-tolerant record-keeping mechanisms that can identify failures through distributed consensus protocols [55].

Finally, blockchain is a big step towards interoperability [23]. On the one hand, blockchain technology work over heterogeneous hardware platforms, eliminating the dependency on a particular provider. On the other hand, smart contracts provide a semantics-independent platform to develop publicly available system interfaces [12].

3.2 Integration schemes for blockchain and IoT

Blockchain is in the early stages of research and development, and there are still multiple research challenges towards seamlessly integrating IoT and blockchains [12]. Among these challenges, one critical problem is to define the roles for each device in a blockchain-based IoT system [13].

The possible role of an actor in a blockchain system depends on the blockchain implementation. Private blockchains can define different permission levels that translate into several roles[47, 78]. On the other hand, public blockchains typically provide a reduced number of roles [42]. However, for both types of blockchains three major roles are generally accepted, i.e., full node, light node, and transaction issuer [40, 42, 13].

Full nodes hold the complete copy of the blockchain, issue transactions, validate transactions, and may add new blocks by participating in the consensus algorithms [40, 42]. Light nodes hold a reduced copy of the blockchain, issue and validate transactions, but cannot add new blocks. Light nodes are used as an entry point to the blockchain, using limited computational resources [46]. A Transaction Issuer, also called a blockchain client, does not have a copy of the blockchain and neither has the capabilities to create a new block. However, it holds the cryptographic capabilities to create and send transactions to the blockchain through light or full nodes [13]. These nodes can be in the same local network as the transaction-issuer, or in the case of the Ethereum platform, a third-party service like Infura [2] and Metamask[3]. The former is a more suitable choice since using third-party services nullifies the point of decentralization. Table 3.1 summarizes the general roles in a blockchain system.

Table 3.1: Node types in blockchain networks

Node Type	**Storage**	**Validator**
Full Node	Full Blockchain	Yes
Light Node	Block headers	No
Transaction Issuer	None	No

Based on their capabilities, devices at the cloud layer can assume any role in the blockchain system and even participate in the consensus algorithms [6,

[2]www.infura.io
[3]www.metamask.io

13]. Devices at the edge can usually take any role, but they rarely participate in the consensus. In this case, other full nodes in the blockchain network can carry out decentralized consensus and block validation [13]. However, sensing devices typically have limited capabilities, restricting their possible roles in the blockchain-based system. Thus, integration architectures are essential to describe blockchain-based IoT systems [13].

While an increasing number of works are integrating blockchain into IoT systems, few of them provide a direct role for most constrained devices. If sensing devices are not direct actors on the blockchain, they do not have a blockchain identity [32]. From the surveyed literature, we have categorized the proposed architectures into three integration levels: at the cloud, at the edge, and the device. The layer at which blockchain is integrated reflects where the blockchain identity of the IoT device is managed.

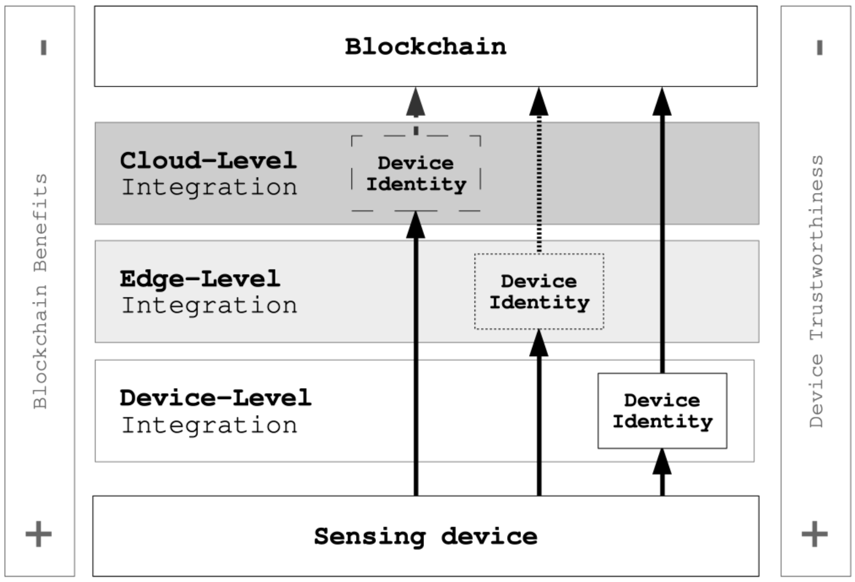


Figure 3.3: Blockchain integration for IoT systems

3.2.1 Cloud-level integration

This approach preserves the traditional IoT architecture and the blockchain act as an additional element in the system. Cloud-level integration is promoted by an increasing number of cloud service-providers offering Blockchain-as-a-Service (BaaS). In BaaS, users access a blockchain node or blockchain-related services for a fixed monthly price [105, 106]. This integration scheme offers the benefits of high computational power, but it fails to achieve full decentralization. Thus, cloud-level integration might be suitable for a permissioned blockchain-based system.

3.2.2 Edge-level integration

IoT devices at the Edge Layer (e.g, a gateway) acts as a transaction issuer, but it can also serve as a Light Node or Full Node according to the device capabilities. Devices from the sensing Layer register with the gateway device, and the gateway issues transactions to the blockchain. The degree of decentralization achieved through this approach is not as fine-grained as device-level integration. This integration scheme is currently the most common approach for blockchain-based IoT systems. In the following paragraphs, we discuss relevant edge-level integrations architectures found in current literature.

One of the first architectures integrating blockchain and IoT is presented by authors in [23] for a Cloud-Based manufacturing platform. Figure 3.4a shows the system architecture based on a gateway device called BPIIoT. The device allows connecting existing Industrial Machinery with blockchain and other Industrial cloud-based services. Blockchain is a service inside the BPIIoT and interacts with an Ethereum implementation. Thus, the BPIIoT manages the identity of each IoT device in the system. This architecture is a clear example of edge-level integration, based on the device used and its role in the architecture.

A layered architecture is proposed in [80] and [107] for a Smart Home use case. Authors introduce the Smart Home Manager (SHM), a device acting as a bridge between IoT devices and the blockchain. The device's identities are managed by the SHM, as it keeps a shared key for local communications with the devices. Moreover, the architecture includes different blockchains, as each home has its private instance of Ethereum. Thus, this architecture presents another edge-level integration, focused on security and privacy in a smart-home environment, limiting the extension to other use-cases.

Authors of [108] proposed a *privacy-aware gateway*, which can connect BLE-based devices to a private Ethereum network. Blockchain provides a repository of access and privacy policies for each of the IoT devices. Devices send their information to the blockchain through the gateway. Moreover, users also must go through the gateway to access detailed information about each IoT device. Thus, the gateway becomes a mandatory intermediary for the interactions of both users and devices.

One of the most cited works integrating blockchain with IoT is by Novo [25]. The author presents an architecture for managing access policies for IoT devices through the blockchain. Figure 3.4b shows the proposed architecture based on a *management-hub*, a device that acts both as a gateway and Full Node in a private Ethereum implementation. Once again, the integration is at edge-level. Moreover, the architecture considers that the IoT devices are not linked nor connected to the blockchain. The IoT devices interact with the management-hub through typical IoT protocols (i.e., CoAP messages) for sending and receiving blockchain-information.

To summarize, although this approach is innovative, it does not removes the intermediary (i.e., the edge-level device) to access the blockchain system. The constrained capabilities of the sensing IoT devices may oblige using these intermediaries to compensate for non-blockchain (e.g., communications, storage, processing power). However, an intermediary managing the devices identity is a step back towards a decentralized IoT environment. On the one hand, the sensing device is linked to a particular component, reducing its mobility and flexibility. On the other hand, the intermediary introduces security concerns in the system, weakening the trustworthiness of the acquired data [32]. Moreover, since the intermediary is at the edge layer, it provides a bigger attack surface for the IoT system.

3.2.3 Device-level integration

Integrating blockchain at the device layer, as mentioned in [13] and [12], is the optimal approach to extend the benefits of blockchain to the IoT systems. In this scheme, the device itself is the transaction issuer. The IoT device manages its own identity and does not rely on any other component in the system. The device interacts directly with the blockchain, creating a trustworthy data-source [32]. The trade-off here is a higher degree of autonomy of IoT devices and applications, versus increased computational complexity of IoT hardware. Thus, the main limitation to achieve this scheme is the lack of computational power to perform the cryptographic primitives necessary to manage the blockchain identity.

Few works have addressed this scenario and, to the best of our knowledge, no framework grants constrained sensing IoT devices as direct actors on public blockchains networks.

3.3 Interactions through smart contracts

After integrating blockchain from a component perspective, it also necessary to integrate blockchain from an application perspective [33]. Smart contracts present a platform to implement the logic enabling the autonomous interactions of the IoT devices. With this new paradigm, mutually untrusted parties interact by placing the trust in the secure and correct execution of the smart contract[109]. Thus, smart contracts must provide the functionalities required by the IoT sensing devices as services for the framework.

According to the recent literature, smart contracts mainly provide "assets management" capabilities to IoT applications, i.e., the ability to create, transfer, and track an asset [15], [110]. Using this functionality, authors have proposed use cases where autonomous interactions among devices occur in the context of a data marketplace [12, 13, 28, 110]. In these cases, the blockchain provides a billing platform where devices can sell data directly to users or other devices. Similarly, other authors have also presented a service marketplace [12, 110, 23], where the devices can rent some of their capabilities (e.g., storage and CPU power). In both cases, smart contracts provide only implement the basic functionalities for these types of renting transactions.

Supply Chain Traceability [12, 13, 23] is another application where smart contracts are used in conjunction with IoT devices. Traceability is a complex process where several untrusted actors are involved. Blockchain also provides an additional layer of transparency and trust, enabling a common shared ledger of transactions. Currently, IoT devices automatize some of the steps of this process. For example, devices such as RFID tags and RFID readers can autonomously indicate when a particular asset arrived at a particular destination, creating a new transaction in the blockchain visible to all the actors interested in that asset.

However, there is no approach where smart contracts serve as autonomous agents representing IoT devices. Once the smart contract is deployed, the inherent properties of blockchain guarantee that the software will continue to execute as intended[87]. Thus, a smart contract can take information and make decisions on behalf of an IoT device enabling autonomous interactions and not storing data.

To this goal, smart contracts need to overcome several issues. Smart contracts have proved to be the weakest link for the security of blockchain-based systems, particularly in public networks. When a smart contract is stored on the blockchain, it becomes immutable and can not be updated as traditional software [12]. This inherent feature of blockchain translates into a very peculiar software life-cycle, where updates or new releases are not straightforward [109]. Additionally, the code of a smart contract is public since it is deployed by a transaction, making any possible vulnerability easier to find [23].

The combination of these two properties (immutability and auditability) may become a threat to applications, as shown by the “DAO exploit”, where a vulnerability was uncovered and exploited in a smart contract deployed in the Ethereum network [4]. The DAO was a “Decentralized Autonomous Organization” based on a smart contract that collected venture capital later spent according to the votes of those who had invested. Without the possibility of an update to fix the vulnerability, the software remained immutable. An attacker exploited this vulnerability several times, obtaining over 3.6M of Ether (near 60M of USD at that time).

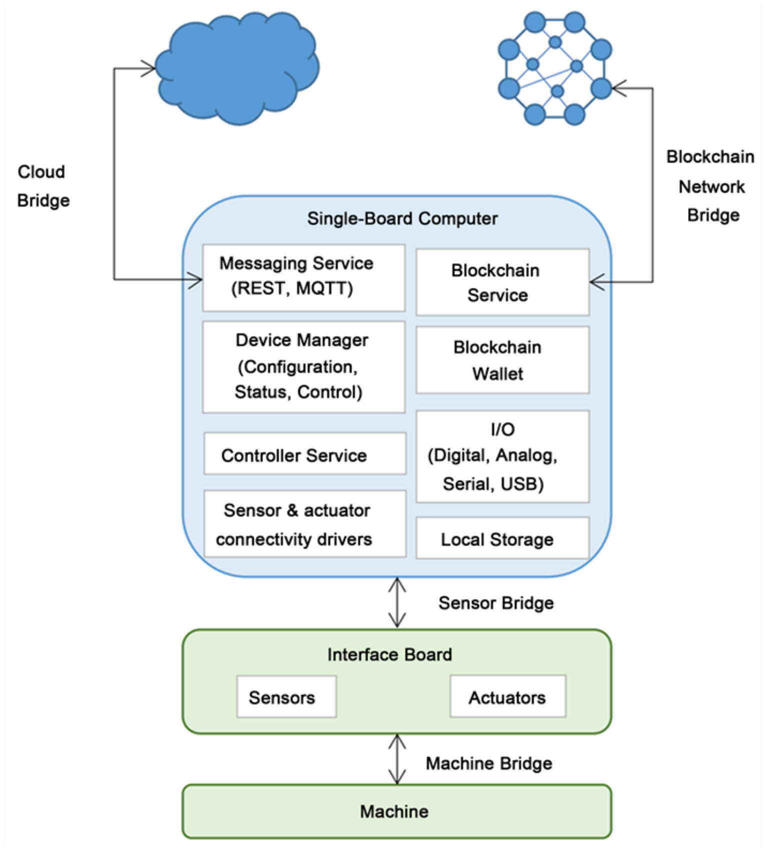
Similarly, another vulnerability was found and exploited in the smart contract used by the Parity wallet application [5]. A function without the appropriated access restrictions allowed an attacker to become the owner of the contract. This attacker later killed the contact (meaning that it was not usable anymore) and froze over 500.000 Ethers (200M USD at the time) belonging to the users of the wallet.

Based on the unique properties of blockchain-based software, current literature highlights the necessity of adapting and adopting software-engineering best-practices to the design and development of smart contracts [111, 109,

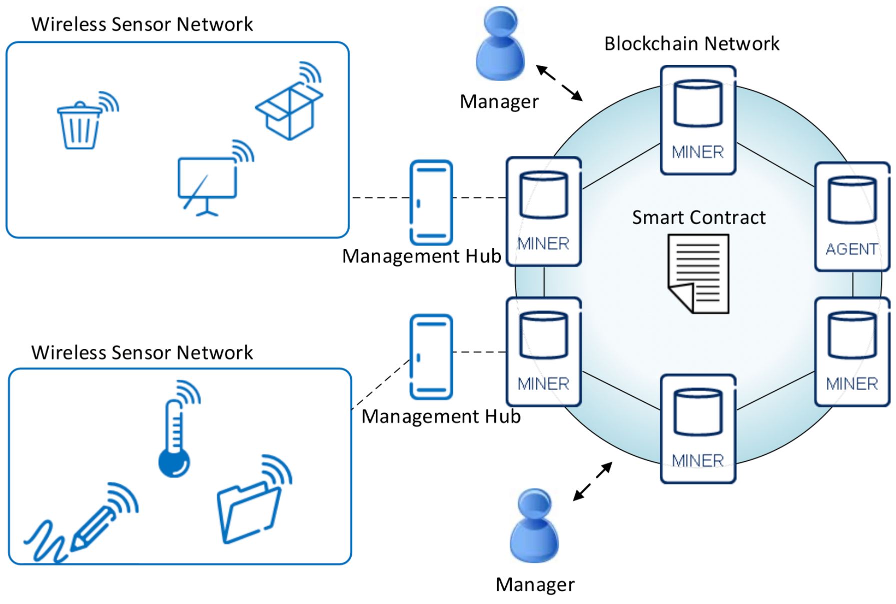
[4]https://www.coindesk.com/dao-attacked-code-issue-leads-60-million-ether-theft/

[5]https://www.parity.io/a-postmortem-on-the-parity-multi-sig-library-self-destruct/

112]. On the one hand, smart contract security is an open and growing research topic[109]. On the other hand, the use of software patterns could help mitigate the current challenges of blockchain software [113, 109]. A recent study [89] proved that more than 20% of existing smart contracts are cloned from industry-approved contract templates, implementing the most common functionalities.


Cloud Bridge
Blockchain Network Bridge
Single-Board Computer
Messaging Service (REST, MQTT)
Blockchain Service
Device Manager (Configuration, Status, Control)
Blockchain Wallet
Controller Service
I/O (Digital, Analog, Serial, USB)
Sensor & actuator connectivity drivers
Local Storage
Sensor Bridge
Interface Board
Sensors
Actuators
Machine Bridge
Machine

(a) Blockchain-based IoT architecture presented in [23].



(b) Blockchain-based IoT architecture presented in [25].

Figure 3.4: Two examples of IoT architectures integrating blockchain at Edge-level.

3.4 Summary and gap analysis

In this section, we have described IoT systems and their most crucial requirements. Then, we have identified the current challenges linked to the centralized cloud-architecture of modern IoT systems and how blockchain technology can alleviate some of these concerns. We followed these descriptions surveying the existing literature describing the integration of blockchain and IoT, organizing the works into three integration schemes.

Despite this growing interest in blockchains as open, transparent, and auditable tools for modern IoT applications, there has been little discussion on the impact that blockchain may have on sensing devices. These devices are the core of IoT systems and typically the most constrained hardware elements of the technology stack.

Most of the current works in the literature, aiming at enabling blockchain functionalities at the device level, still favor IoT devices with computing requirements (in terms of processing, memory, energy) more similar to those devices belonging to the Edge Layer (i.e., system-on-a-chip) than the Device Layer (i.e., microcontrollers) [18]. This approach contrast with the average requirements of many IoT applications (e.g., Agriculture, Smart Cities). In those applications, power efficiency and low-cost are central concerns to ensure cost-effective and long-term operations [21]. It is no coincidence that the authors of [20] identify the full understanding of the energy requirements and processing time of this class of devices interacting with a blockchain-based IoT architecture as one of the most challenging open questions.

When dealing with severely constrained sensing devices that interact with a blockchain-based system, the usual approach is to rely on other architectural components of the IoT technology stack (i.e., integration at the edge-level). A significant issue of this approach is that constrained sensing devices do not directly interact with the blockchain. Thus, they are not direct actors of the blockchain-based system. This intermediary still introduces security concerns in the system [6] and considering that this intermediary is typically at the gateway layer, there is a bigger attack surface that may compromise the data integrity of several sensing devices. Thus, adopting sensor nodes as direct actors on a public blockchain infrastructure needs detailed investigation.

IoT devices can be considered oracles of a blockchain-based system, i.e., external components that provide information not available on the blockchain system [114]. IoT sensing devices as direct actors of the blockchain is a considerable step toward trustworthy oracles for blockchain systems [32].

Another topic missing from the current literature is the benefits of using public blockchain infrastructures. The majority of the studied works focused on permissioned infrastructures [18, 26]. This approach is suitable for applications and domains with predefined restrictions and constraints regarding access and authorization. However, a permissioned blockchain does not provide the same level of openness, decentralization, and neutrality as a permissionless network. Moreover, the PoW consensus and the large number of honest miners found on existing public blockchains present a more secure platform for developing decentralized applications [28, 29].

Therefore, for the effective design and implementation of an integration framework for blockchain and IoT, we first need to identify and address the main technical challenges of having such constrained sensing devices as trusted oracles of a blockchain-based system [32]. The limitations imposed by the communication capabilities and the reduced energy budgets of some IoT deployments need to be identified, quantified, and analyzed by the framework, following a component-oriented approach.

Once the constrained sensing devices are direct actors of the blockchain-based systems, we need to design high-level software components to develop decentralized IoT applications. Smart contracts have the capabilities for implementing complex business logic; however, there is limited general knowledge about the best practices for blockchain-oriented software engineer [109]. Currently, using design patterns for smart contracts can help to avoid security pitfalls and prevent vulnerabilities [113]. Design patterns should describe the typical functionalities that the software component provides. The framework must identify these general functionalities that smart contracts should provide to IoT applications following a service-oriented approach.

Part II: Blockchain-based decentralized IoT applications

Chapter 4

A novel trustless architecture for blockchain-based IoT applications

This chapter contains text taken from the published work:

- *Pincheira, M., Ali, M. S., Vecchio, M., & Giaffreda, R. (2018, May).Blockchain-based traceability in Agri-Food supply chain management: A practical implementation. In 2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany) (pp. 1-4). IEEE.*
- *Pincheira, M., Vecchio, M., Giaffreda, R., & Kanhere, S. S. (2020, May). Exploiting constrained IoT devices in a trustless blockchain-based water management system. In 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-7). IEEE.*
- *Pincheira, M., & Vecchio, M. (2020, June).Towards Trusted Data on Decentralized IoT Applications: Integrating Blockchain in Constrained Devices. In 2020 IEEE International Conference on Communications Workshops (ICC Workshops) (pp. 1-6). IEEE.*

In this chapter, we present a novel trustless architecture for blockchain-based IoT applications using constrained devices. The architecture is the product of a DSR process over two case studies in the agricultural domain. The chapter is structured as follows. Section 4.1 presets a high-level overview of the architecture. Section 4.2 describes the novel contributions of the architecture. To conclude, section 4.3 conceptualizes the architecture into a layered framework and describes each of its components. The proposed architecture is further analyzed and evaluated with the case studies on Chapters 5 and 6.

4.1 High-level architecture

The proposed blockchain-based architecture relies on the benefits of blockchain technology on permissionless network (Section 2.5) to alleviate some of the current challenges in centralized IoT architectures (Section 3.1.2) considering the requirements of constrained sensing devices (Section 3.1.1). Figure 4.1 presents a high-level overview of the architecture.

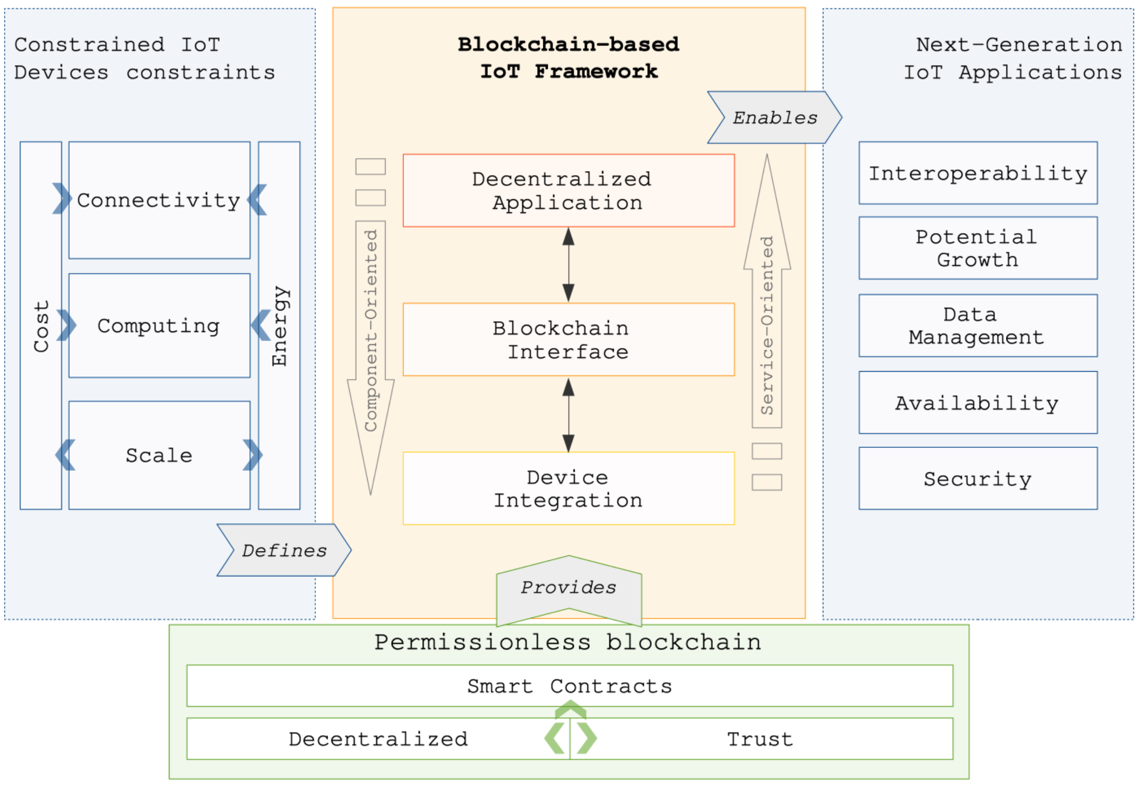


Figure 4.1: High-level overview of the novel trustless architecture for blockchain-based IoT applications

The architecture presents three novel characteristics: i) it consider constrained sensing devices as direct actors on the blockchain system, ii) it favors a permissionless blockchain infrastructure, iii) it provides a basic layer of services to develop new types of decentralized IoT applications. The architecture assumes that each device securely manages its cryptographic keys. Although this is a strong assumption, it is aligned with the state of the art [30, 31]. Furthermore, blockchain holds the potential to more complex identity schemes as key-building blocks for realizing completely decentralized public key infrastructures [11]. However, this research falls outside the focus of this thesis.

4.2 Novel contributions

The proposed architecture provides several benefits to mitigate some of the current challenges of IoT systems, as previously described in Section 3.1.3. Concerning the current gap in the literature (Section 3.4), the architecture presents the following three novel contributions:

4.2.1 Constrained sensing devices as direct actors on the blockchain system

The proposed architecture considers low-cost and energy-efficient sensing devices. Sensors are the core for modern IoT systems, and the sensed data is the base to develop IoT applications [5]. Constrained devices align with the requirements of modern IoT systems described in Section 3.1.2. In the proposed architecture, sensing devices are direct actors on the blockchain system, have a unique blockchain identity, and generate digitally signed transactions. The use of cryptography at the root of the architecture aligns with the current security challenges in IoT applications [115]. This approach guarantees a *root of trust* for the sensed data towards trust-worthy blockchain oracles [32].

4.2.2 Permissionless blockchain as the underlying network

The proposed architecture favors a permissionless/public blockchain network. Permissionless blockchain networks based on PoW consensus are the most secure platform for developing decentralized applications [28]. The main two public blockchain networks, i.e., Bitcoin and Ethereum, are estimated to have 10,000 [37] and 8,000 [116] actives nodes respectively. This network size provides high levels of redundancy, and fault-tolerance, increasing the availability of a blockchain-based IoT system. In the proposed architecture, a permissionless environment also guarantees the auditability and transparency of the information. These properties alleviate the concerns regarding data management on centralized IoT systems. A permissionless system also provides better scalability to address the potential growth of modern IoT systems. On the one hand, permissionless networks are easier to scale (Section 2.3.3). On the other hand, it provides an open platform for the integration of several unknown actors to increase the system value.

4.2.3 Smart contracts as a software platform

The proposed architecture uses smart contracts to provide the basic services for decentralized applications. Smart contracts provide a platform to define complex business logic, autonomously enforcing agreements between untrusted actors [33] based on trusted values coming from ubiquitous IoT devices. This approach allows true IoT interoperability among multi-vendor devices and applications [7], with the increased security of underlying blockchain network.

4.3 Blockchain-based decentralized IoT framework

We conceptualized the proposed architecture as the layered framework shown in Figure 4.1. The framework is composed of three main modules, i.e., Blockchain, Gateway, and Device.

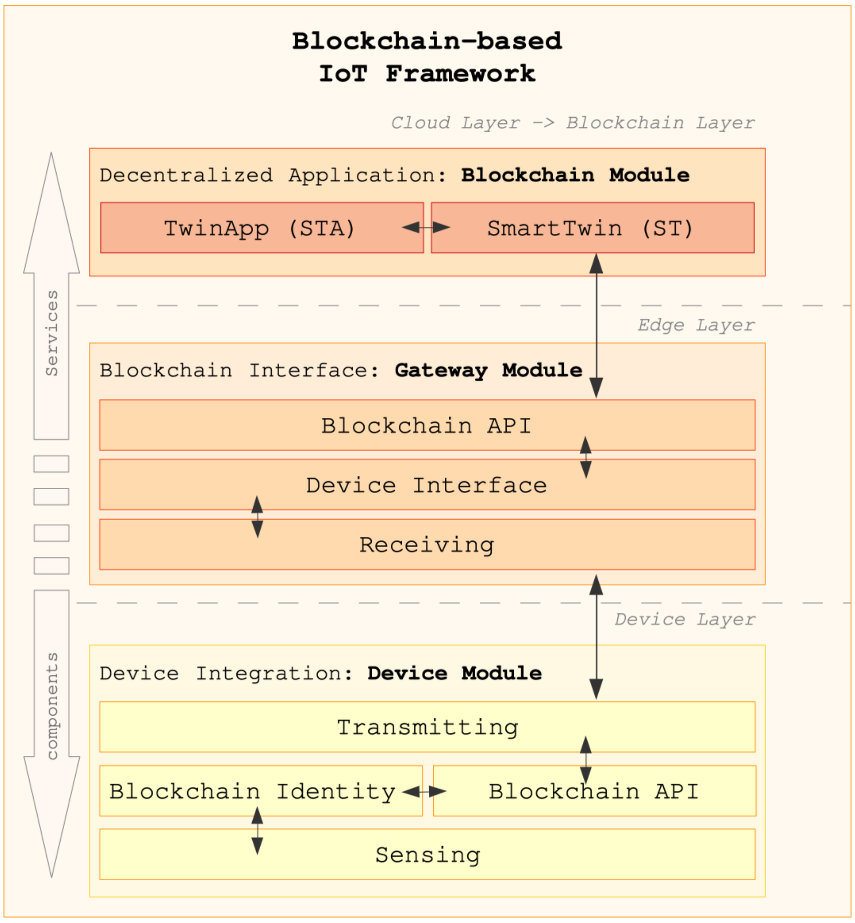


Figure 4.2: Blockchain-based decentralized IoT framework.

Following the component-oriented perspective of modern IoT architectures, our framework is divided into three modules. The **Device module** is located at the Device layer and provides the necessary tools to transform a constrained sensing device into a direct actor on a blockchain system. The **Gateway module** is located at the Edge layer and is a simple relay component between the two modules. The **Blockchain module** replaces the centralized Cloud Layer with a decentralized Blockchain Layer and provides an interface for developing new types of decentralized IoT applications.

From a service-oriented perspective, the lower modules address the low-level requirements of constrained sensing devices. Conversely, upper modules provide high-level services to develop new types of decentralized IoT applications.

Formally, the framework can be described as follow: Given a sensing device d than produces a sensed value v, an IoT system D composed of n sensing devices is described as :

$$D = [d_1, d_2, ...d_n] \tag{4.1}$$

For such IoT system D, our framework considers that

$$\forall \; d \in D \quad \exists! \; st \in ST \tag{4.2}$$

where

$$ST = [st_1, st_2, ..., st_n] \tag{4.3}$$

Where each st_x is the “digital twin” of the device (i.e., SmartTwin) as an autonomous agent in a permissionless blockchain system B.

This assumes that each device d has its own identity dI based on a unique pair of cryptographic keys K such that

$$\forall\ d \in D \quad \exists! \quad K = (dK_{public}, dK_{private}) \tag{4.4}$$

This architecture enables the direct interaction between two devices (d_a, d_b) through their corresponding smart contracts (sc_a, sc_b) using a transaction Tx.

Conversely, given a group of x unknown users U such that

$$U = [u_1, u_2, ..., u_x] \tag{4.5}$$

The framework enables the creation of a group of y of "Smart Twin Applications" (i.e., TwinApp) STA such that :

$$STA = [sta_1, sta_2, ..., sta_y] \tag{4.6}$$

Thus, the framework enables the trustless architecture for blockchain-based IoT applications without the need for intermediaries between users and devices. In the framework, constrained sensing devices (D) act as trustworthy data sources on a permissionless blockchain system (B). Smart contracts provide an abstraction to develop next-generation blockchain-based applications (STA) that can benefit from any sensing device (ST) and can be used by any unknown user (U). Figure 4.3 shows the envisioned application of the framework, and the following sections describe each of the framework components.

4.3.1 M1: Device module

The device module enables the integration of constrained sensing devices as direct actors on a blockchain-based system. This module provides the standard functionalities of an IoT device (i.e., sensing and transmitting) and the cryptographic functionalities to interact with the blockchain system (i.e.,

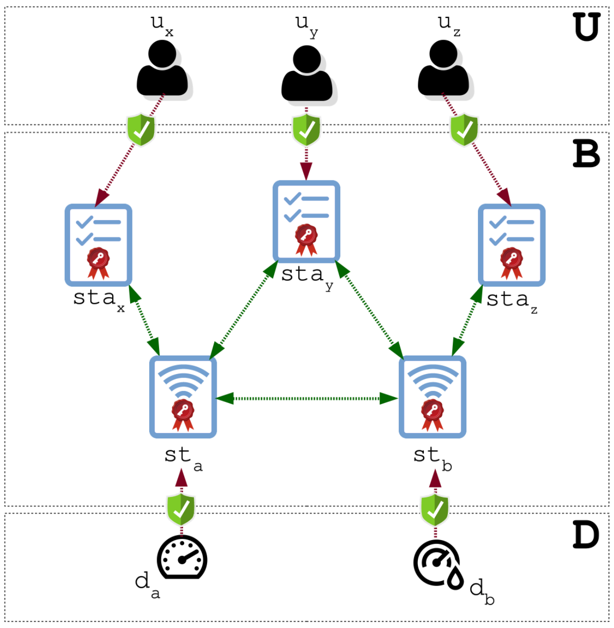


Figure 4.3: Trustless architecture: Constrained Sensing devices (D) provide trustworthy information to a permissionless blockchain (B) where unknown users (U) securely interact with decentralized IoT applications.

identity and api). The following sub-modules describe the four functionalities of the device module.

- **M1.1) Sensing:** Controls the sensor that transforms a reading of the physical world into a sensed data value v. For better compatibility with existing IoT systems, this sub-module works as traditional sensing operations in IoT devices.

- **M1.2) Blockhain API':** Provides a limited set of methods to interact with high-level functionalities of the blockchain network, i.e., access and retrieve information and interact with the smart contracts. In particular, this module specifies the transaction format Tx to match the corresponding ST. Considering the general structure of a transaction (Section 2.1.1)

$$Tx = \{T_{header}, T_{destination}, T_{data}, T_{value}, T_{signature}, T_{extra}\} \tag{4.7}$$

The module provides the definition of Tx and also the values for $\{T_{header}$, $T_{destination}$, $T_{extra}\}$. Additionally, the module also provides the operations to transform Tx into a network-compatible version Tx' (e.g., compressing/decompressing, encoding/decoding) according to the communications capabilities of the device (M1.4)

- **M1.3) Blockhain Identity:** Provides all the cryptographic functions required to create a direct interaction with the blockchain (i.e., a valid transaction Tx). The functions include Elliptic Curve Digital Signature Algorithm (ECDSA) and hashing algorithms optimized for constrained devices. The key pair K is managed in this sub-module and provides a unique identity dI to the device d.

  Considering the general structure of a transaction Tx and given a sensed value v, this modules creates the data of the transaction T_{data}, T_{value}, and the signature $T_{signature}$.

- **M1.4) Transmitting:** Provides the methods to transmit Tx (or Tx') to the gateway, using the communication interface available in the sensor. For better compatibility with existing IoT systems, this sub-module works as traditional transmitting operations in IoT devices.

4.3.2 M2: Gateway module

The gateway module is a transparent gateway connecting two (or more) communication interfaces as traditional IoT architectures at the edge layer. The gateway module can not modify the transactions Tx as is signed by the sensing devices, guaranteeing data integrity through this layer. The two sub-modules of this module work as follows:

- **M2.1) Receiving:** Receives Tx (or Tx') using the communication interface that connects with the sensing device. For better compatibility with existing IoT systems, this sub-module works as usual receiving operations in edge devices.

- **M2.2) Blockchain API:** Provides all the methods to interact with high-level functionalities of the blockchain system, i.e., access and retrieve information and interact with the smart contracts. The module also provides the operations to recover Tx from Tx' (e.g., compressing/decompressing, encoding/decoding). The submodule M1.2 (Blockchain API') is a reduced version of this component.
- **M2.3) Blockchain Access:** Sends Tx to a blockchain node using the network interface at the edge device. Since blockchain is a peer-to-peer network, and this sub-module can choose the best peer (e.g., with the lowest latency) among several options.

4.3.3 M3: Blockchain module

The blockchain module enables a distributed application platform. It relies on the scripting capabilities of the blockchain platform (i.e., smart contracts). The framework defines two models as high-level smart contract abstractions that specify a minimum set of primitives. Applications can extend these models to fill the requirement of each particular use case. The base models are SmartTwin and TwinApp

- **M3.1) Smart Twin:** Defines a base smart contract to create a SmartTwin (ST) that represents an IoT device on the blockchain. The contract implements three methods, namely *setValue()* to register a value from the IoT device, *getValue()* to return the registered value, and *callApp()* to interact with a TwinApp. The contract also stores an internal variable called device, representing the identity dI of the device d that can change the internal state of the SmartTwin. A call to *setValue()* is only accepted if it comes from the device d.

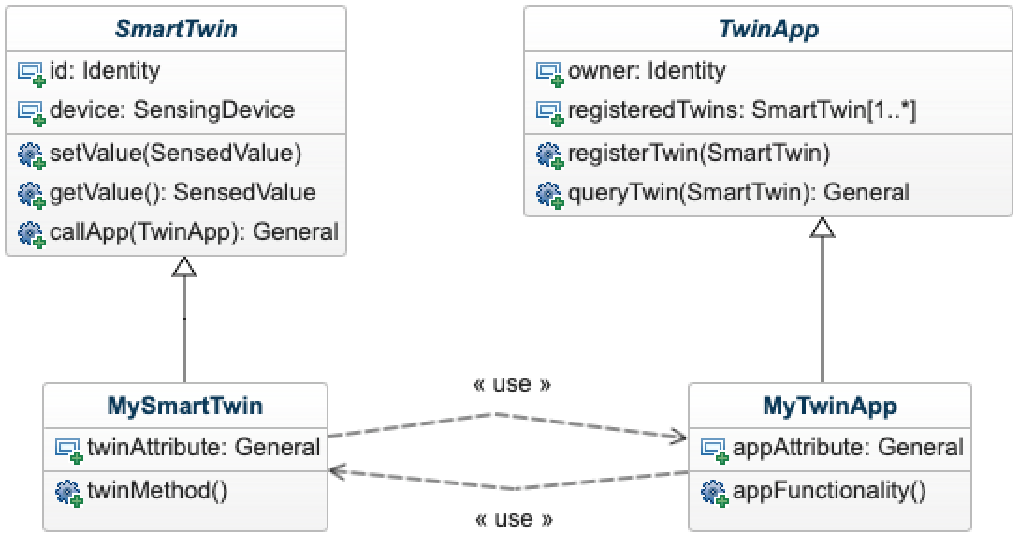


Figure 4.4: Class diagram of a sample application using Smart Twins, and Smart Twin Apps.

- **M3.2) Twin App:** Defines a base smart contract to create a TwinApp (STA) that interacts with a SmartTwin contract. The TwinApp implements two methods, namely *registerTwin()* and *queryTwin()* and stores an internal variable called *owner*, representing the user *uinU* that created the TwinApp. The method *queryTwin()* obtains the latest value stored on a SmartTwin. This method is only valid if SmartTwin has previously register with the TwinApp using the *registerTwin()* method. Registration does not necesary impose restrictions and aims to internal intializations on the application.

The blockchain module aims to provide high-level components for developing applications and thus, UML diagrams can provide a better description. Figure 4.4 shows a class diagram representing the SmartTwin and the TwinApp. Figure 4.5 show a sequence diagram of SmartTwin and a TwinApp interacting in the context of a sample application.

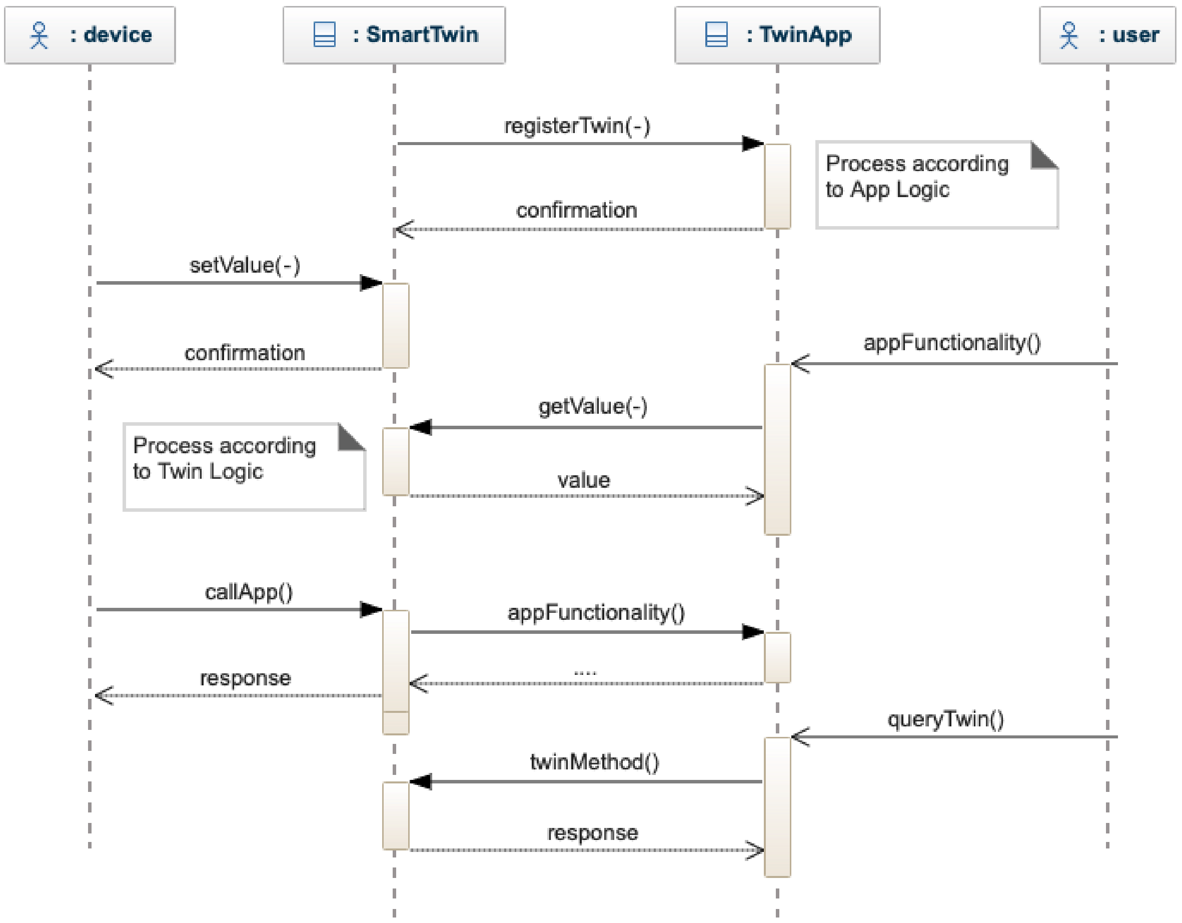


Figure 4.5: Example interactions over the proposed architecture.

Chapter 5

A blockchain-based IoT system for traceability in Agri-Food

This chapter contains text taken from the published work:

– *Pincheira, M., Ali, M. S., Vecchio, M., & Giaffreda, R. (2018, May).Blockchain-based traceability in Agri-Food supply chain management: A practical implementation. In 2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany) (pp. 1-4). IEEE.*

This chapter presents AgriBlockIoT, a case study with the first version of the novel trustless architecture for blockchain-based applications. The proposed architecture seamlessly integrates IoT devices into a blockchain-based traceability solution for Agri-Food supply chain management. To effectively assess AgriBlockIoT, we defined a classical use-case within the agricultural domain, namely from-farm-to-fork. We developed and deployed such use-case, using two different blockchain implementations, namely Ethereum and Hyperledger Sawtooth. We evaluated and compared the architecture performance in both implementations in terms of latency, CPU, and network usage. Finally, we highlighting their main pros and cons of the architecture when using different types of blockchain implementations.

5.1 Introduction

Today, the vast majority of traditional logistic information systems in Agriculture and Food (Agri-Food) supply chains merely track and store orders and deliveries, without providing features as transparency, traceability and auditability. These features would surely improve food quality and safety, therefore they are more and more requested by consumers [117]. Thus, several Research & Development communities are concentrating their efforts on adopting some specific Internet of Things IoT technologies such as RFIDs and Wireless Sensor Networks, or everyday-cheaper connected devices, to enabled remote monitoring of the conditions in food transportation scenarios and at a very fine granularity along the whole Agri-Food supply chain, *e.g.*, from production to consumption [118].

However, the majority of the current IoT solutions still rely on heavily-centralized cloud infrastructures, where there is usually a lack of transparency, and by nature presents security threats including availability, data lock-in, confidentiality and auditability [10].

In the Agri-Food domain, in order to maintain trust and reliability along the whole supply chain, it is essential for the stored records to be tamper-proof, while the best case would be if each actor issuing transactions could do that without relying on any centralized third-party intermediary. A potential solution to alleviate all of such issues and concerns is the Blockchain technology, which is a peer-to-peer digital ledger that does not rely on centralized servers. Since all the records stored in a blockchain are based on a consensus reached at least by the absolute majority of peers of the network itself, this distributed ledger is immutable by design and offers an auditable and transparent source of information. And from an IoT perspective, instead of requiring connectivity to a central cloud, sensor networks in a blockchain-based traceability solution would only require stable connection to their closely located peer. Thus, blockchains exposes all the required properties for decentralizing food traceability systems, while making traceable data available at every step of the supply chain.

In this chapter, we present AgriBlockIoT, a fully-decentralized traceability system for the Agri-Food supply chain management. Specifically, the proposed solution can rely either on the Ethereum [46] or the Hyperledger Sawtooth[1] publicly available blockchain implementations, while it is able to integrate various IoT sensor devices. By directly producing and consuming valuable information from the IoT devices along the whole supply chain and storing such data directly in its underlying blockchain, AgriBlockIoT guarantees transparent and auditable asset traceability. To assess the feasibility of the proposed solution, we engineered and deployed the so-called *from-farm-to-fork* use-case: a classical food traceability scenario fostering certified traceability of food along the whole supply chain, *e.g.*, from agricultural production (the farm-side) to consumption (the fork-side). Then, we compare the two implementations, in terms of three performance metrics, namely latency, CPU load, and network usage.

The remaining of this chapter is organized as follows: Section 5.2 summarizes the current state of the art in the adoption of blockchains as an enabling technology for the traceability in Agri-Food supply chains; Section 5.3 describes the system architecture of AgriBlockIoT; Section 5.4 contains the analysis of our preliminary results; Section 5.5 presents the conclusions and future work.

5.2 Related work

The last few years have witnessed an explosion of research and development activity around blockchain technology, mainly within the financial technology

[1]https://www.hyperledger.org/projects/sawtooth

(FinTech) industry. Blockchain's intrinsic capability of providing immutable and tamper-proof records, together with its potential of enabling trust and reliability among untrusted peers, are extremely attractive features, preventing this technology to stay relegated into a single vertical sector. For this reason, several industries beyond the FinTech sector have already identified the Blockchain technology as a driver for a paradigm shift. For data reliability, ProvChain [119] explored the use of the Blockchain technology in a cloud storage scenario to verify three levels of data provenance: collection, storage and validation. In this work, the use of blockchains showed good results in terms of tamper-proof records and user privacy, with very low overhead for the storage itself. In a similar context, the authors on [120] explored the use of blockchains with smart-contracts to achieve secure data provenance, using the Open Provenance Model (OPM) with an access control-based privacy-preserving solution.

Similarly, the adoption of IoT devices and technologies in the supply chain management sector has attracted a lot of research interest in the last few years. From the impact of autonomous identification system [121] to the application of RFID technologies in logistics [122], the technological maturity of devices and sensors is revolutionizing each step of the supply chain management process. Specifically for the Agri-Food domain, the authors of [123] presented an inventory transparency use-case adopting IoT devices. There, the goal was to explore the use of RFID and NFC-based devices to achieve transparency and real-time information production directly on the field, enabling persistence by means of a centralized, cloud-based database. This is indeed the classical paradigm adopted by far the majority of the current IoT-based solutions.

However, the use of both the Blockchain and the IoT technologies in the Agri-Food domain is still an under-explored, yet worth-to-explore, research field. A traceability system based on the blockchain and the RFID technology was proposed in [124], with a sharp focus on Chinese food markets. The work considered fresh food asset tracking as fruits, vegetables and meat, by means of RFID-based devices for the data acquisition and blockchains for data persistence. The authors of [125] presented a supply chain traceability system for food safety, based on HACCP (Hazard Analysis and Critical Control Points) and focus on transparency. There, they described the process of crop plants in different phases, from harvesting to retailing, without going into the details of a performance analysis. Overall, to the best of our knowledge, some key-features offered by certain blockchain implementations remain either not explored, or not fully exploited, one for all being the autonomous transactions capability (often referred to as smart contracts [120]).

5.3 Proposed system architecture

The unique constrains and requirements of the modern Agri-Food industry pose some major challenges to achieve a transparent, auditable and reliable supply chain management process . Some of these challenges are the heterogeneity of the involved actors, stakeholders and business models, their different levels of confidentiality, the lack of interoperability among the involved systems and, most notably, the complete lack of a clear data governance [126]. Figure 5.1 depicts a simplified version of such process, whose involved actors are briefly introduced in the following:

A) **provider:** providers of raw materials, such as seeds and nutrients, but also pesticides, chemicals, etc;

B) **producer:** usually the farmer *e.g.*, the responsible of the actions from seeding/planting to harvesting;

C) **processor:** this actor may perform various actions, from simple packaging to more complex processes (*e.g.*, pressing of the olives);

D) **distributor:** this actor is responsible of moving the output of the processor (*e.g.*, the product) from processor's site to retailers;

E) **retailer:** this actor is responsible of selling the products, representing it either small local stores or big supermarkets;

F) **consumer:** the final element of the chain.

Along the whole process, authorities provide standards, regulations, laws, rules and policies that the involved actors have to comply with.

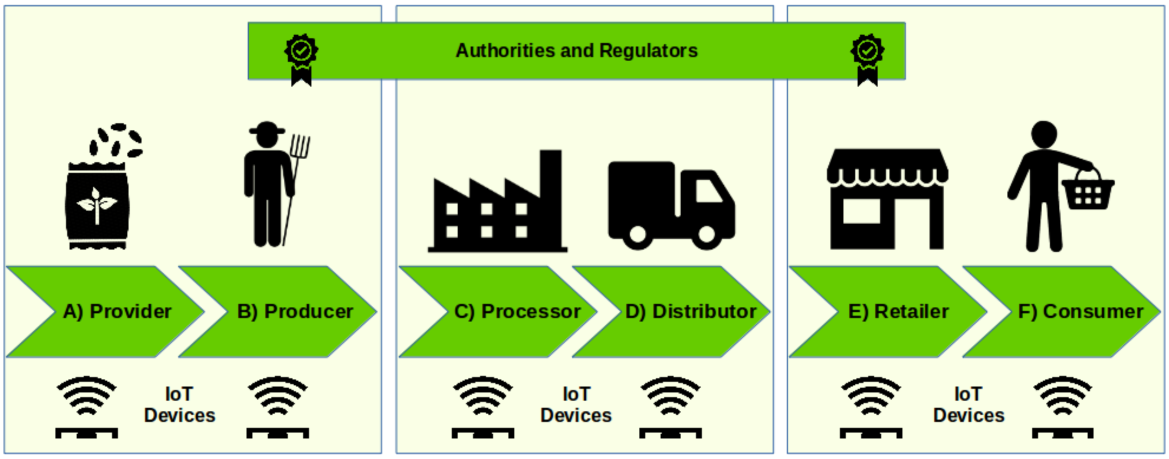


Figure 5.1: Simplified version of the Agri-Food supply chain management process.

We propose a layered architecture able to rely on blockchain and IoT technologies to achieve transparency, auditability and immutability of the stored records in a *trustless* environment. We consider the blockchain as a layer of our system (see Figure 5.2), allowing AgriBlockIoT to be blockchain-independent, while it can be integrated into existing traditional software systems (ERP, CRM, etc.).

The proposed architecture takes advantage of the increasing capabilities offered by modern edge devices (*e.g.*, gateways, mini-PC, etc.), which may be directly used as full nodes of our layered blockchain implementation, hence extending the resistance, decentralization, security and trust of the whole network. The main modules of AgriBlockIoT are :

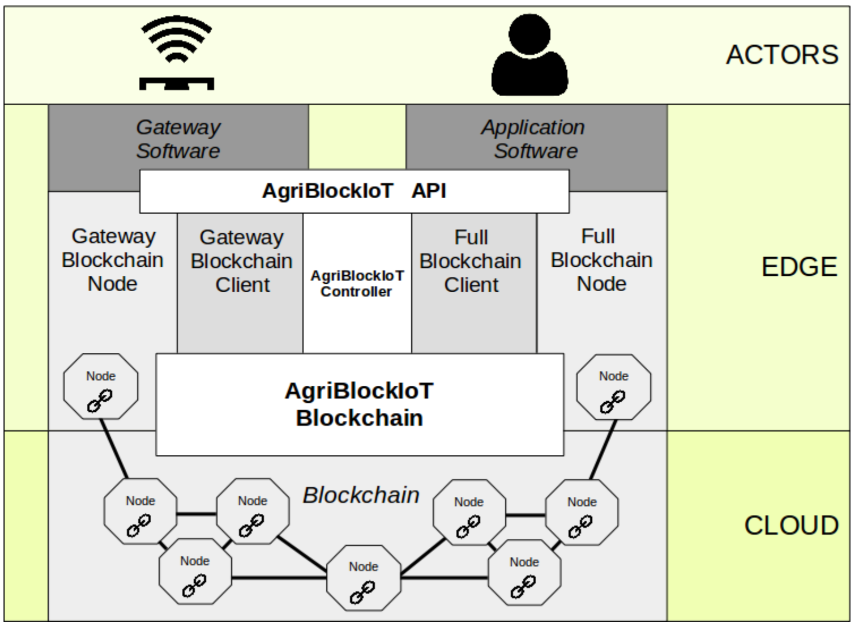


Figure 5.2: Layered architecture of AgriBlockIoT.

- **API:** a REST Application Programming Interface exposing the capabilities of AgriBlockIoT to other applications, with a high level of abstraction, allowing easy integration with existing software systems;
- **Controller:** a component responsible of transforming the high-level function calls into the corresponding low-level calls for the blockchain layer, and viceversa (*i.e.*, querying and converting the data records stored in the blockchain, into high-level information for the upper layer);
- **Blockchain:** The main component of the system, containing all the business logic, implemented through smart-contracts on the blockchain, as a gateway to the blockchain itself. Depending on the selected blockchain, this module will vary in complexity, according to the program capabilities and the client interfaces for each particular blockchain.

Then, to coherently define the high-level functionality of AgriBlockIoT, we had a bottom-up approach through which we extracted the set of requirements starting from a complete use-case, namely *from-farm-to-fork*. The latter is, indeed, a classical food traceability use-case that fosters certified traceability of food along the whole supply chain, from agricultural production to consumption. In other words, AgriBlockIoT shall provide consumers with complete history of the food he is buying. The only pre-condition is that all the participants (so including the IoT devices) are registered users of the underlying blockchain, meaning that they have the correct public/private key-pairs to digitally sign each operation on the distributed ledger. In the following, we summarize the list of extracted requirements:

R1: **Raw Materials Purchasing:** producers and providers store in the blockchain the details of sales and purchases of raw materials, including technical information of products and amounts. Note: smart-tags (*e.g.*, barcode, QR codes) can be used to automatize this process;

R2: **Planting:** producers store in the blockchain information about the planting process (*e.g.*, the amount of seeds used). Note: sensors can automatize such data entry process (*e.g.*, connected weight scales), while smart contracts can autonomously fire, hence creating records whenever anomalies are detected (*e.g.*, more seeds than the ones registered as purchased);

R3: **Growing:** sensors, at regular intervals, autonomously store in the blockchain information about the growing plants and environment. Note: smart contracts can asynchronously fire, hence creating records whenever anomalies are detected (*e.g.*, sensor values outside certain thresholds);

R4: **Farming:** farmers store in the blockchain information about each stage of the process (*e.g.*, irrigation, fertilizing, etc.), including amounts of inputs applied. Note: sensors can automatize such data entry process (*e.g.*, chemical sensors and multisensory systems), while smart contracts can autonomously fire, hence creating records whenever anomalies are detected (*e.g.*, sensor values outside certain thresholds);

R5: **Harvesting:** farmers store in the blockchain details about the harvesting. Note: sensors can automatize such data entry process (*e.g.*, connected weight scales), while smart contracts can autonomously fire, hence certifying that the process from seeding to harvesting is compliant with certain regulations (*e.g.*, organic, fair trade, etc.);

R6: **Delivery to processor:** farmers transfer the ownership of the products to distributors, directly through the blockchain. Note: sensors (*e.g.*, GPS sensors) and smart contracts can automatize this process, or create records whenever anomalies are detected during the delivery phase (*e.g.*, sensor values outside certain thresholds);

R7: **Processing:** considering the simplest case of a packaging processor, the latter store in the blockchain details about the received amount of product from distributors, the packaged amount and, eventually, the amount of product lost during the processing phase. Note: sensors can automatize such data entry process (*e.g.*, connected weight scales), while smart contracts can autonomously fire, hence creating records whenever anomalies are detected (*e.g.*, the packaged amount is larger than the received amount);

R8: **Delivery to retailers:** processors transfer the ownership of the processed product to distributors, directly through the blockchain. Note: sensors (*e.g.*, GPS sensors) and smart contracts can automatize this process, or create records whenever anomalies are detected during the delivery phase (*e.g.*, sensor values outside certain thresholds);

R9: **Retailing:** retailers store in the blockchain details about the received amount of product from distributors. Then, at regular intervals, sensors autonomously store in the blockchain information about the status of the retail environment. Note: smart contracts can asynchronously fire, hence creating records whenever anomalies are detected (*e.g.*, sensor values outside certain thresholds);

R10: **Consuming:** retailers store in the blockchain details about the sold products, while consumers are able to transparently verify the whole history of a product before buying it. Note: smart-tags can be associated to each package, so that consumers can easily retrieve the whole history of the product.

5.4 Architecture evaluation

We assess the performance of AgriBlockIoT implementing the functionality of an IoT sensing device producing digital values that are directly stored in the blockchain. The stored data can be then retrieved, while it is possible to implement smart-contracts that are autonomously executed upon the occurrence of certain conditions on the data produced by the sensor itself. Since AgriBlockIoT is blockchain-agnostic, we implemented the underlying blockchain module over two different, private, six-nodes-based implementations, namely Ethereum and Hyperledger Sawtooth. The reasons of choosing these implementations are the different levels of customization for the records included on the ledger (transactions). While both platforms allow to implement complex business logic, Ethereum works with a single transaction structure, while Hyperledger Sawtooth allows the definition of a custom transaction structure. Additionally, Ethereum can be used on both public or private blockchain networks, while Sawtooth is defined for private networks only.

Both the networks were configured with the default settings, and deployed in dedicated virtual machines equipped with 4GB of RAM, 2 Intel(R) Core(TM) i5-6440HQ CPUs 2.60GHz and 20GB of hard disk. Regarding the Operating System, we opted for a fresh Linux Ubuntu 16-04 basic distribution, only installing the packages needed to deploy the corresponding blockchain node. A series of 100 test where run independently for each scenario. During each test, AgriBlockIoT simply set the value of a sensor, as done by an environmental IoT sensing device through a gateway, and issued a transaction in the blockchain. For each test we measured the time necessary to set the value in the blockchain (latency), the processing power of each node (CPU load), and the network usage (in terms of bytes transmitted and received); the average values are summarized in Table 5.1. From these results, we observe that Hyperledger Sawtooth has better performances with respect to the Ethereum counterpart.

Table 5.1: Performance of AgriBlockIoT in terms of latency, network traffic, and CPU load.

	latency [seconds]	**network tx [bytes]**	**network rx [bytes]**	**CPU load [%]**
Ethereum	16.55	528’108	682’415	46.78
Sawtooth	0.021	19’303	20’641	6.75

5.5 Conclusions

AgriBlockIoT enables the integration of IoT and Blockchain technologies, creating transparent, fault-tolerance, immutable and auditable records which can be used for an Agri-Food traceability system. Regarding the preliminary, very practical test: even if the Hyperledger Sawtooth-based implementation had better results in terms of measured metrics with respect to the Ethereum one, both implementations have different properties and capabilities that need to be considered before choosing one over the other. In some cases it may be convenient to trade off the high-latency of Ethereum with its scalability and reliability, since it enables larger numbers of participants and its software maturity is far higher than Hyperledger Sawtooth. Moreover, from an economic perspective, recall that the monetary cost of using the Ethereum public network can be avoided by using private networks. However, in this environment, the limitation of having a single language for implementing smart-contracts, as well as a fixed structure for the records, may represent a drawback when developing more sophisticated business logic. Last but not least, the current consensus algorithm of Ethereum is quite CPU-intensive and this may represent a barrier for computationally-limited devices, such as edge gateways and IoT devices. Conversely, the Hyperledger Sawtooth implementation offers a novel consensus algorithm which may be more suitable for constrained IoT devices. Furthermore, the ability of implementing the logic using different languages, as well as the customization of the records, may enable faster implementations and easier integrations with other systems. However, Hyperledger Sawtooth is still far for being considered a mature implementation at the level of Ethereum.

As future works, we plan to extend the performance analysis to more constrained hardware architectures, in order to assess the suitability of the proposed framework to applications comprising real IoT devices and gateways along the Agri-Food supply chain.

Chapter 6

Cost-effective IoT devices in a blockchain-based water management system

This chapter contains text taken from the published work:

– *Pincheira, M., Vecchio, M., Giaffreda, R., & Kanhere, S. S. (2021).Cost-effective IoT devices as trustworthy data sources for a blockchain-based water management system in precision agriculture. Computers and Electronics in Agriculture, 180, 105889.*

This chapter explores how the energy efficient-integration of IoT-based sensing and blockchains can be used to incentivize virtuous behaviors in agricultural practices. We present an architecture where constrained sensing IoT devices work as trustworthy data sources for a permissionless blockchain. We validate our proposal by implementing a complete use case using Ethereum as a public blockchain network. We evaluate the impact of the architecture on constrained IoT devices in terms of energy, processing time, and available memory using six different types of IoT hardware platforms. The validation results show new means to energy-efficiently integrate IoT data sources in a permissionless blockchain, making our proposal a strong candidate for use in automated and incentive-based irrigation water management systems. Thus, the proposed architecture holds the potential to be a key component in fostering increased sustainability of the whole agricultural sector.

6.1 Introduction

Water management systems in agriculture have a deep socio-economical impact that goes beyond end-users and service providers' benefits [127, 128], as the water used for irrigation accounts for over 70% of the available freshwater resources[129]. This calls for more sustainable water management processes, while service providers face the need for precise systems to measure water consumption of their subscribers, as emphasized by transnational directives such as the European Water Policy[1]. Nowadays, the Internet of Things (IoT) can offer energy-efficient sensing devices embedded within the water distribution systems. Moreover, the recent advances in Low Power Wide Area Networks LPWAN technologies enabled the deployment of such devices at very large scales, also reaching remote agricultural areas with minimal economic impact for both service providers and end-users [130].

However, these types of systems are based on centralized architectures, often managed by the service provider, hence restricting the actors involved in the water management process. In recent years, there has been a growing interest in integrating blockchain technologies into IoT systems [11] for trustless architectures. Blockchain, the technology behind Bitcoin[40], uses a unique combination of cryptography, data structures, and incentive mechanisms to maintain a peculiar type of distributed database (*i.e.*, a *ledger*) in a peer-to-peer network. This distributed ledger is immutable by design and offers an auditable and transparent source of information. For managing resources such as water and energy, IoT and blockchain directly benefit several business processes, such as accounting, billing, and distribution [14]. Moreover, the trustless nature of a blockchain-based system enables the seamless inclusion of several external actors that can directly reward and/or certify certain end-users' behaviors, based on the actual use of the monitored re-

[1]`https://ec.europa.eu/environment/water/index_en.htm`

source [131].

Despite this growing interest in blockchains, previous works have failed to address the impact that such technology may introduce to existent agricultural IoT systems. Current studies consider highly connected environments, without communications or energy restrictions for the blockchain-enabled devices, while agricultural IoT systems favor low-cost and power-efficient devices [21, 132]. These requirements translate into severely constrained architectural elements that cannot guarantee the fulfillment of the central requirement of being direct actors of a blockchain-based infrastructure. Consequently, current research has only focused on sensing devices that rely on another system component (e.g., a gateway) to interact with the blockchain [24, 25]. However, the presence of this intermediary introduces security concerns on the data flow [6], reducing the trustworthiness of the sensed data [11]. Furthermore, since the intermediary is typically a gateway, there is a bigger attack surface that may compromise the data integrity of several sensing devices. Thus, a need emerges to quantitatively judge whether constrained IoT devices can be a direct actor within a blockchain-based system, acting as a trustworthy data source [32].

In this chapter, we present a system architecture based on constrained IoT devices for measuring water consumption, working as direct data-source actors in a public blockchain infrastructure, where smart contracts represent the interests of different water management stakeholders and regulate the distribution of incentives amongst virtuous farmers.

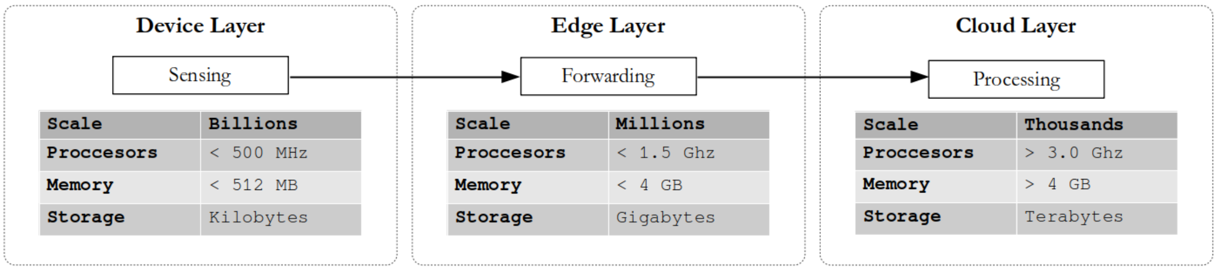
The novelty of the study lies in the use of constrained sensing devices as trustworthy data sources for a permissionless blockchain, supporting the interests of a diverse set of water management stakeholders and enabling the smooth inclusion of additional external participants. The contribution of this chapter is threefold. First, we highlight the unique benefits that permissionless blockchain networks provide to multi-actor scenarios. Second, we unveil the fundamental role that can be played by constrained IoT sensing devices as trusted data sources for such permissionless blockchain infrastructures. Third, we quantitatively assess the impact of this architectural choice on the constrained devices commonly used in agricultural IoT deployments. To this end, we consider typical metrics found in current literature such as available computing power and energy budgets for the IoT devices [130, 21], and transaction costs and times for the blockchain operations [131, 51].

To this aim, we developed a cross-platform software library that allows constrained devices to be direct actors on a public blockchain network. We provide a basic structure for the smart-contract that can benefit from these types of blockchain-enabled devices. Then, by using dedicated equipment able to measure power consumption at very high resolution, we evaluate the performance of this software implementation on six different IoT platforms belonging to three hardware families, namely ARM, AVR, and MIPS. Despite some specific characteristics of the selected water management use case, our study reaches a general conclusion: even severely constrained, low-cost, battery-powered devices like those used in agricultural IoT deployments are capable of interacting directly with a public blockchain. Quantitatively, we demonstrate that the blockchain-related operations require an additional 6% of the average energy budget needed for normal (*i.e.*, non-blockchain) device operations.

The remainder of the chapter is structured as follows: Section 6.2 offers a brief literature overview about IoT and blockchain integration. The proposed system architecture is presented in Section 6.3, followed by the implementation of a fully-working proof-of-concept (PoC) described in Section 6.4. Based on this PoC, Section 6.5 presents an evaluation of several metrics regarding the integration of IoT devices in a blockchain system. The chapter concludes with Section 6.6, summarizing the main results and outlining potential future research works.

6.2 State of the art and rationale

Modern IoT systems typically consist of several components, each of them with unique constraints and capabilities. In line with current literature, we consider a classical 3-layer technological architecture as shown in Figure 6.1 [6]. The Device layer group together those sensing devices capable of acquiring data from the physical world. These devices are typically small and favor low-power consumption modes against powerful capabilities, such as computational power, memory, and connectivity. The Edge layer (sometimes also referred to as the Gateway) group together less-constrained devices providing additional connectivity and computational power to the lower layer. Finally, the Cloud layer represents powerful servers responsible for processing and store the data collected by the Device layer and forwarded through the Edge layer.



Scale	Billions
Proccesors	< 500 MHz
Memory	< 512 MB
Storage	Kilobytes

Scale	Millions
Proccesors	< 1.5 Ghz
Memory	< 4 GB
Storage	Gigabytes

Scale	Thousands
Proccesors	> 3.0 Ghz
Memory	> 4 GB
Storage	Terabytes

Figure 6.1: Typical high-level 3-layer architecture of modern IoT systems.

The unique combination of features offered by blockchain technology, such as its openness, transparency, auditability, and non-repudiation, can directly benefit several IoT-related processes [11]. These features are particularly attractive when IoT technology is monitoring and managing resources such as water and energy [14]. Using smart meters as sensing devices, authors of [82] propose a software platform for sustainable management of water supply systems. In their proposal, after monitoring water usage, a blockchain provides a trading platform for exchanging tokens that reduce the cost of energy based on this usage. Similarly, authors of [83] propose a privacy-friendly *gamifi-*

cation approach based on water consumption data acquired by IoT devices owned by the utility company. The collected information is then used by the gamification process, profiting from the blockchain transparency to prevent dishonest behaviors. Recently, the authors of [133] presented a system to manage and coordinate the use of water by irrigation communities. In this case, several individuals gather for better access to the resource, and blockchain enables trust among community members based on the information collected by the IoT devices.

However, and to the best of our knowledge, there has been little discussion on the use of constrained sensing devices as trustworthy data sources for a permissionless blockchain system. On the one hand, studies integrating blockchain into the IoT systems tended to focus on sensing devices with capabilities more similar to the Edge layer (*i.e.*, system-on-a-chip) than the device layer (*i.e.*, micro-controllers), as described in [18]. We argue that this approach is in full opposition with the average requirements of typical agriculture applications, where power efficiency and low-cost are essential requirements to ensure cost-effective and long-term operations [21].

On the other hand, whenever considering more constrained sensing devices in the blockchain, the current approach is to rely on other architectural components of the IoT architecture ([24, 25]). A significant drawback of this approach is that constrained sensing devices do not send their data directly to a blockchain. Thus, they are not direct actors of the blockchain-based system. The presence of an intermediary introduces security concerns in the system [6], hindering the trustworthiness of the acquired data. Considering that this intermediary is typically at the gateway layer, there is a bigger attack surface that may compromise the data integrity of several sensing devices [11]. Last but not least, the communication technologies typically used by these constrained devices also present constraints and restrictions that few researchers have addressed. Indeed, agricultural applications require energy-efficient communication networks ([21, 133, 97]) such as those provided by LPWAN radio technologies (*e.g.*, LoRaWAN, NB-IoT, *etc.*), each of them with different capabilities and limitations [130].

This chapter introduces an architecture where constrained IoT devices are direct data-source actors for a sustainable water management system, where a permissionless blockchain supports the interests of a diverse set of water management stakeholders. We identify, analyze, and address the challenges for having sensing devices as trusted oracles in the blockchain system [32] with a focus on the limitations imposed by the communication capabilities and the reduced energy budgets of these types of agricultural deployments.

6.3 Proposed system architecture

We propose a system architecture for a decentralized water management system that incentivizes and rewards virtuous behaviors for sustainable agricultural practices. Our architecture considers low-cost and energy-efficient sensing devices able to accurately measure water consumption as direct actors of a public blockchain-based infrastructure. Thus, each device manages its own blockchain identity, autonomously creating a transaction and sending it over an LPWAN network. In our system, each actor is identified by its unique combination of public/private keys. In the following, we assume that each actor of our system securely manages its own private key. While this is a strong assumption, it is important to recall that blockchains have also the potential for more complex identity schemes, being them key-building blocks for realizing completely decentralized public key infrastructures [11]. However, this research segment falls outside the focus of this thesis. Since each transaction is digitally signed by the sensing device itself, the information can be considered as coming from a trustworthy oracle and directly feed a smart contract. Moreover, the use of cryptography at the root of the system is aligned with the current challenges in IoT applications [115]. This process guarantees data integrity by creating immutable, auditable, and non-repudiable records that are easily verifiable by other users. Considering the broad socio-economical impact of water management systems [127], our architecture favors a public blockchain infrastructure to provide a permissionless environment among several unknown actors. Last, but not least, to leverage on smart contracts as the foundation for decentralized applications represents a novel approach to realize true IoT interoperability among multi-vendor devices and applications [7].

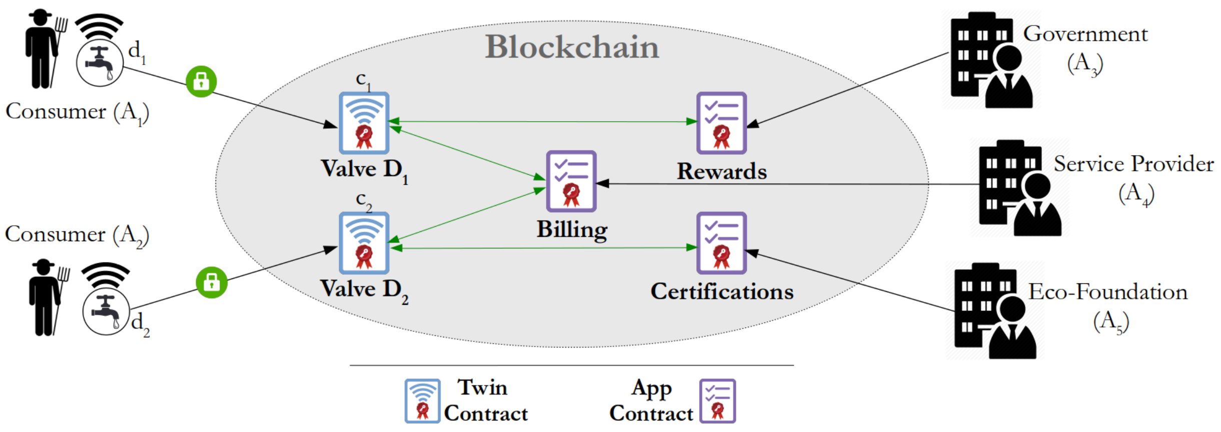


Figure 6.2: High-level architecture of the proposed blockchain-based sustainable water management system.

To keep low the transaction rate of the sensing nodes, we allow for aggregating multiple readings at sensor side. This means, for instance, a sensor node can transact hourly, daily, or even weekly-aggregated chunks, depending on the requirements of the application. However, individual data points can still be stored in more traditional IoT platforms. In this way, the users (or the service provider) can set the frequency of the reports based on the trade-offs in terms of the energy budget of the sensor nodes, transaction cost in the public blockchain, and maximum delay tolerable by the business process.

In our system architecture, each device is represented by a unique smart contract deployed in the blockchain. More formally, if an IoT deployment D is composed by n sensor devices d_i (with $i = 1, \dots, n$) such that $D = [d_1, \dots, d_n]$, then $\forall\ d \in D \quad \exists!\ c \in C$, where $C = [c_1, \dots, c_n]$ represents the group of smart contracts c_i mapping the i-th device in the blockchain. Therefore, each smart contract can be seen as the device's "digital-twin" in the blockchain (for analogy, we refer to it as the device's *"smart-twin"*). This contract has a template interface, including both public and private methods. Private methods update the state of the twin in the blockchain and can be invoked only by the device owning the blockchain identity, while public methods simply provide a standardized interface for other smart contracts. The applications in our architecture are based also on smart contracts interacting with the smart-twins. Thus, a billing contract or a rewarding contract can use the devices' contracts as transparent source of information in a trustless way, while the government or an environmental organization can grant rewards to foster sustainable behaviors on water management interacting with the same smart-twins. A high level overview of our architecture is shown on Figure 6.2.

To better characterize our architecture and similarly to modern IoT systems (Figure 6.1), we split it into three modules, namely the Device, Gateway, and Blockchain modules, as shown in Figure 6.3. On current IoT systems, sensors at the device layer capture a real-world phenomenon such as temperature, humidity, or resource consumption. In our architecture, the Device module is responsible for converting the sensed values into blockchain transactions that, later sent to the corresponding smart-twin counterparts. The Gateway module is a dumb relay component between the device and the blockchain layer, exactly like the majority traditional IoT architectures at the edge layer. Notice that the gateway module is unable to modify the transactions already generated and signed by the sensing devices. Finally, the Blockchain module gathers together all the smart contracts representing the smart-twins and the distributed application itself. In contrast to traditional cloud layers in IoT systems, smart-contracts provide a fully decentralized interface for developing new types of IoT applications. The following sections thoroughly describe the three modules of our proposal.

This system architecture makes noteworthy contributions to the current state of the art. First of all, it adopts constrained IoT devices as trusted oracles of a blockchain-based system as instruments to certify very specific behaviors. Currently, these certification processes are heavily intensive on manual labor and are often able to provide only discrete snapshots in time. Conversely, the presented architecture makes a step towards the full automation of those processes requiring adherence to specific sets of rules, granted that IoT devices can monitor such adherence. Overcoming energy and communications constraints, and without relying on any additional architectural component, these IoT devices become a convenient alternative to vouch for the truthfulness of the collected data. Moreover, in a water management system that fosters water savings, the benefits of using a permissionless blockchain in terms of openness and transparency far outweigh the disadvantages concerning transaction throughput and latency. The permissionless network allows any stakeholder willing to put a value tag to those savings, to use this architecture not only for the truthfulness of the collected data, but also as a platform to directly and securely reach the farmers.

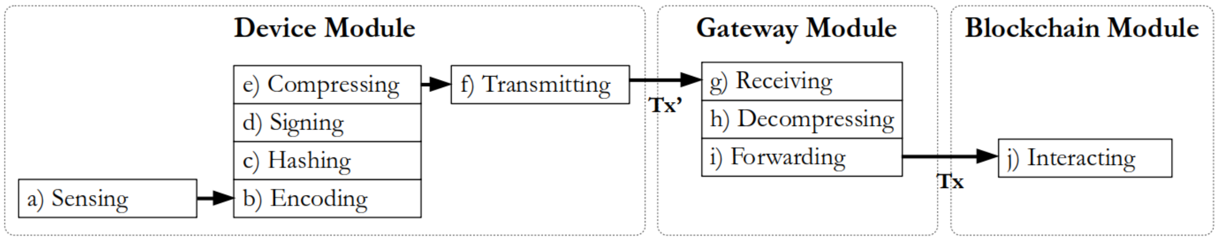


Figure 6.3: Software modules of the proposed architecture.

6.4 Implementation

To better describe the proposed architecture, we implemented it as a fully-working showcase. First of all, to wisely select the most suitable distributed infrastructure, we had to consider the primary requirement of our architecture to leverage on a permissionless network with scripting capabilities. Although some recent DLT protocols (*e.g.*, Hyperledger, and IOTA) offer attractive features for distributed IoT applications, we opted for the Ethereum[46] blockchain since it is still considered as the reference public blockchain implementation for smart contracts [89]. However, to migrate our implementation to a different infrastructure with scripting capabilities should not be an issue.

Secondly, in this work, we are sharply focused on the role of constrained IoT sensing devices as trusted data sources in a blockchain-based architecture, while other infrastructure-related challenges (*e.g.*, synchronization of blockchain nodes, transaction updates, etc.) are beyond the scope of this thesis. The interested reader is referred to [51] for a detailed description of such open challenges.

Finally, considering that there are no additional communications requirements for the application (in terms of existing infrastructure, range, or band licensing), we chose LoRaWAN over other LPWAN alternatives such as SigFox and NB-IoT. Briefly, when compared to the latter technologies, the former offers a comfortable balance between coverage, maximum payload size, and infrastructure costs. However, for more accurate technical details and comparative analysis on the advantages and limitations of various LPWAN technologies, the readers are referred to [130] and references therein.

6.4.1 The Device module

The values acquired by the sensing devices need to be converted into a blockchain transaction. In Ethereum, a transaction is defined as a “single cryptographically-signed instruction” sent from one address to an other. A transaction T is formally described in [42] as:

$$T = (T_n, T_p, T_g, T_t, T_v, T_d, T_w, T_r, T_s),$$

where T_n is the nonce indicating the number of transaction previously sent by the issuing address; T_p is the gas price; T_g is the gas limit; T_t is the destination address; T_v is the value that is transferred to the particular address within the operation; T_d is an array with the input data of the message call (in the case of a contract call, it specifies the function identifier and the parameters to be passed to that function); T_w is the recovery identifier (or chain identifier); finally, T_r and T_s are the (r, s) values of the elliptical curve digital signature for the transaction. All the components of T are summarized in Table 6.1.

Table 6.1: Components of a transaction in the Ethereum blockchain.

Component	**Description**
T_n	Number of transaction
T_p	Gas price
T_g	Gas limit
T_t	Destination address
T_v	Value transferred within the transaction
T_d	Input data of the message call
T_w	Recovery identifier (or chain identifier)
T_r	R component of the ECDSA for the transaction
T_s	S component of the ECDSA for the transaction

The process for creating a transaction T, though completely performed at device level, can be split into three stages, namely Encoding, Hashing and Signing. Considering that there is no official Ethereum implementation that can be used in constrained IoT devices and, to the best of our knowledge, no third-party library providing cross-platform compatibility exists, we implemented our own software library. Based on open-source initiatives and favouring cross-platform compatibility over code optimization, we used C language within the Arduino development framework. In the following paragraphs, we detail each function carried out by the Device module.

Sensing implements the logic for measuring the water consumption. For our PoC we utilized the YF-201 water flow sensor [2], a low-cost valve, based on a magnetic Hall-Effect sensor that outputs an electrical pulse as water passes through. The sensor is managed by a micro-controller as a general-purpose input with an attached interrupt, and the measurement is based on the elapsed time between two consecutive interruptions, without the need for any additional software library.

Enconding this stage creates T_u, a byte-array concatenating an encoded version of each field of T. The encoding is a custom serialization method called Recursive Length Prefix (RLP) defined by Ethereum in [42]. The sensed value of the previous stage is combined with the smart contract method name identifier to create the field T_d.

Hashing applies the Keccak-256 cryptographic hashing algorithm to T_u. Despite the size of T_u, this cryptographic unidirectional function always returns a 256 long bit-array that unequivocally and uniquely represents the hashed transaction.

[2]`http://www.mantech.co.za/datasheets/products/yf-s201_sea.pdf`

Signing at this stage, T_{hash} is digitally signed using the Elliptic Curve Digital Signature Algorithm (ECDSA). The result is a unique tuple (r, s) based on a private key P_k and the input T_{hash}. This tuple is encoded using RLP to create the byte-array representing the transaction T. The size of T is variable, and in our PoC started from 140 bytes increasing over time. This eventually becomes an issue for LPWAN transport networks, as bigger payloads have an impact in both power consumption and coverage ([130]). Thus, we include a dictionary-based compression stage to reduce the average length of T.

Compressing this stage applies a simple dictionary-based compression algorithm to the input byte-array T. As first step, we remove the RLP encoding to T_d, T_w, T_r, T_s to obtain T'_d, T'_w, T'_r, T'_s . Then, we added T'_{d1} and T'_{d2} as codewords taken from the dictionary, replacing (T_p, T_g) and (T_t, T_v, T_d) respectively. The result T' is a compressed version of T defined as $T' = (T'_h, T'_n, T'_{d1}, T'_{d2}, T'_w, T'_r, T'_s)$.

Transmitting for our PoC, we choose the LoRaWAN radio technology. We used an off-the-shelf communication module based on the RFM95W chip, relying on the Lmic free library[3]. Our network infrastructure is based on the Over-The-Air-Activation (OTAA) which requires to store an application-key and a device-id on-board the device. Devices used SF7 and 125kHz bandwith in the EU868 band.

6.4.2 The Gateway module

From the perspective of current IoT systems, this module is a typical IoT gateway agent, receiving data from one layer and forwarding to another. In particular, it receives blockchain-signed packets from the LoRaWAN stations

[3]`https://github.com/matthijskooijman/arduino-lmic`

and forwarding them to a blockchain node. This module was implemented in Python 3.6 and deployed it on a Raspberry Pi 3B board[4] running a clean install of Raspbian 9, with kernel 4.14.

Receiving this stage represents the interface with a wide area network infrastructure, providing the compressed transaction T' sent by the device module. In our particular case, it interfaces with the LoRaWAN radio stack.

Decompressing this stage symmetrically reverses the dictionary compression of T', applying the RLP encoding if needed. The output of this stage is, consequently, the original T byte-array. It is important to highlight that even a single bit-swap while reconstructing T, would invalidate the original signature of the transaction, hence invalidating the transactions itself.

Forwarding at this stage, the re-assembled transaction T is finally sent to a blockchain node. For our PoC, the node is an Ethereum full node hosted on a physically different computer.

6.4.3 The Blockchain module

This module of the architecture groups the smart contract realizing the decentralized application of the water management system. Interactions in our architecture are based on the interactions between the corresponding smart contracts. To achieve this goal, we envision two types of smart contracts *Twin* and *Apps*, described in the following paragraph.

Interacting a *Twin* contract is a simplified representation of the IoT device in the shape of a "*smart twin*" (see Valves in Figure 6.2). Besides maintaining the values sensed by the device, they also provide a common interface for the

[4]`https://www.raspberrypi.org/`

Apps. For this PoC, a *Twin* contract stores only the owner of the device it corresponds to and the timestamped measured values (water measurements). As an interface, the *Twin* implements two methods, namely `setValue()` to update the measurements (this method can be invoked only by the device the contact represents) and `getValue()` to let any transaction issuer read the measurements.

The *App* contract, on its hand, implements all the business logic of the system, directly interacting with the *Twin* contracts by calling the latter's public methods to, for instance, accredit water consumption, assign rewards, or grant certifications. To alleviate concerns deriving from the security vulnerabilities that a smart contract may introduce in our system, we adopt industry-approved libraries in our implementation. Specifically, we created a rewarding *App* based on the ERC20 Token contract provided by OpenZepellin[5], a library for secure smart contract development considered as a community-standard. Finally, the *App* contract implements a method called `Reward(`c_x`)` that distributes tokens based on the water usage of a *Twin* c_x relative to a threshold previously defined.

6.5 Architecture evaluation

In this section, we evaluate the performance of our software implementation on different micro-controllers boards (MCU). The main goal is to quantitatively analyze the impact that being a direct actor on a public blockchain infrastructure has at the IoT device level. By focusing on very constrained hardware families (all characterized by clock frequencies lower than 100 MHz, reduced memory and program space) we can benchmark low-cost, battery-powered devices suitable for typical precision agriculture scenarios. More in detail, we select six different IoT platforms of three different hardware

[5]`https://github.com/OpenZeppelin/openzeppelin-contracts`

families, namely AVR, MIPS, and ARM. While an exhaustive review of all possible IoT platforms goes beyond the scope of this research, the selected sample provides a reference for other scenarios and use cases. Finally, regarding the underlying blockchain infrastructure, we opt for the open source Ethereum codebase, as it is considered as the de-facto standard of permissionless, cross-industry, token-based blockchains with smart contracts capabilities, while benchmarking different blockchain implementations and consensus protocols is out of the scope of this research.

Table 6.2: Main characteristics of the hardware platforms used in the evaluation campaign.

Device	**Model**	**MCU**	**Architecture**	**Clock (Mhz)**	**Prog. Mem (KB)**	**SRAM (KB)**	**Price (USD)**
ATM	Arduino Uno	ATMega328P	8-bit AVR	16	32	2	18
M0+	STM32L031K6T6	Cortex M0+	32-bit ARM	32	32	8	10
M0	STM32F030R8T6	Cortex M0	32-bit ARM	48	64	8	10
PIC	ChipKit Lenny	PIC32MX270	32-bit MIPS32	40	256	64	23
M4L	STM32L452	Cortex M4	32-bit ARM	84	512	96	12
M4F	STM32F401RET6	Cortex M4	32-bit ARM	84	512	96	12

In our selection, the most constrained device is the *ATM*, an 8-bit microcontroller from the AVR family typically used on Arduino-like boards. On the opposite end, from the ARM family, we have the *M4*, a 32-bit board with 48x more memory, and 16x more program space than the ATM, within the same price range. This platform offers both low power (*M4L*) and high performance (*M4F*) alternatives to better accommodate the needs of each use case. Our pool includes also a board of the *M0* family, which is a 32-bit platform that aims at very low power consumption modes. Finally, we include a constrained board belonging to the MIPS family. Table 6.2 summarizes the six boards included in our pool, detailing their architecture, clock frequency, program space, memory size, particular model, and reference price (updated to April 2019).

6.5.1 Device module footprint

To estimate the footprint of the device module, we incrementally created the code by adding the functions and libraries needed by each software stage. Based on the statistics reported by the compiler of each board, we estimated the program size and memory usage of the code for each stage. These results are summarized in Table 6.3 and Table 6.4, respectively. Then, Figure 6.4 and Figure 6.5 graphically depict the results of the previous tables, as percentages of total program size and memory available at each device. It is important to notice that the ATM and the MP0 (both equipped with 32 KB of program space) are not able to host the whole module implementation. In both the graphs and the tables, the – mark highlights this circumstance.

Table 6.3: Device module program size footprint (in bytes).

Device	**Available**	**a) Sen**	**b) Enc**	**c) Hsh**	**d) Sig**	**e) Com**	**f) Trx**
ATM	32.256	2.570	3.018	7.364	14.396	594	-
M0+	32.768	15.672	3.980	2.188	6.900	416	-
M0	65.536	13.368	3.976	2.188	6.900	416	21.000
PIC	249.856	35.148	29.568	4.120	16.548	844	22.988
M4L	524.288	16.996	4.120	2.264	6.792	436	17.332
M4F	524.288	14.628	3.772	2.264	6.792	436	17.148

6.5.2 Device module performance

To evaluate the overhead that our architecture introduces into the constrained devices, each board created and transmitted 100 blockchain transactions. At the end of each stage (Sensing, Encoding, Hashing, Signing, Compressing, and Transmitting) the total elapsed time (in milliseconds) was reported by the board. Table 6.5 presents the average processing time overhead of 100 transactions. For the sake of completeness, for the boards that

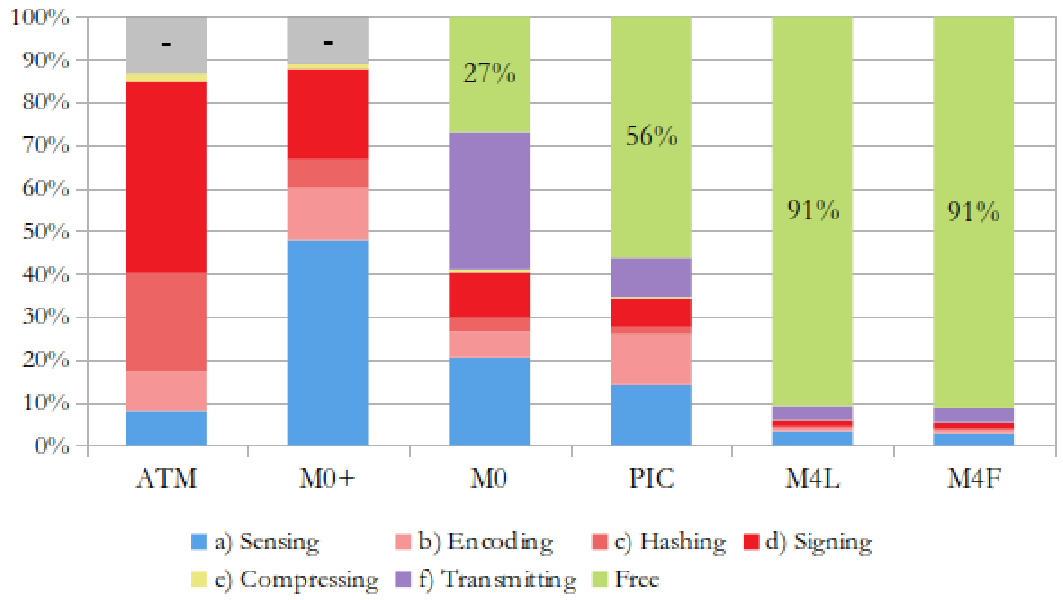


Figure 6.4: Device module footprint in terms of program size (in percentage).

were not able to host the full device module, we measured the time for the blockchain-related stages only (*i.e.*, Encoding, Hashing, Signing and Compressing).

6.5.3 Transaction compression

The experiments described in Section 6.5.2 were also used to calculate the difference in size between T' and T. On average, the compression stage was

Table 6.4: Device module memory footprint (in bytes).

Device	**Available**	**a) Sen**	**b) Enc**	**c) Hsh**	**d) Sig**	**e) Com**	**f) Trx**
ATM	2.048	206	96	440	407	81	712
M0+	8.192	1.012	616	104	164	76	796
M0	8.192	1.060	616	104	164	76	752
PIC	65.536	8100	60	0	164	76	628
M4L	98.304	1052	616	104	164	76	752
M4F	98.304	1072	596	124	164	76	740

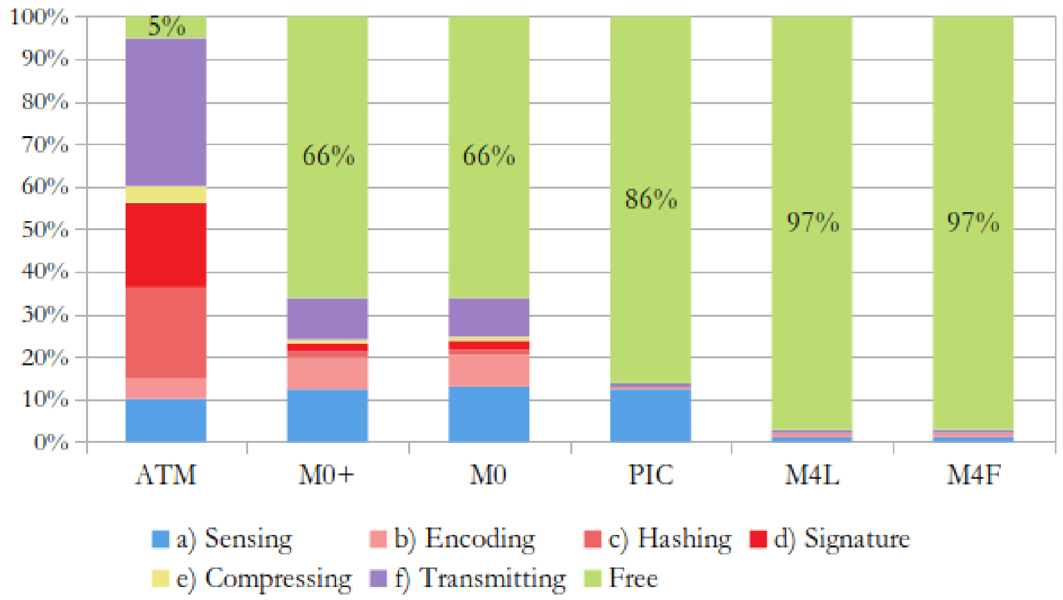


Figure 6.5: Device module footprint in terms of memory (in percentage).

Table 6.5: Average processing time overhead of the device module (in milliseconds).

Device	b) Enc	c) Hsh	d) Sig	e) Com	f) Trx
ATM	0,523	29,235	4.187,285	0,514	-
M0+	0,469	4,738	683,110	0,442	-
M0	0,282	3,324	471,009	0,235	2.452,982
PIC	0,129	1,651	304,567	0,109	2.461,227
M4L	0,111	1,563	124,938	0,085	2.448,768
M4F	0,083	1,204	118,254	0,069	2.455,960

able to reduce the size of T by 45% (*i.e.*, from an average of 140 to an average of 75 bytes). Moreover, this stage added minimal burden to the boards: in the most constrained device (ATM), this stage required less than 1% of the available program size, 4% of the available memory, and less than 1ms to execute, as shown in Table 6.3 and Table 6.5.

6.5.4 Transaction cost and processing time

According to the information provided by *Geth* (*i.e.*, the official Ethereum client), we obtained the amount of gas needed to create both the Twin and the App smart contracts. At the same time, we obtained the amount of gas needed for executing the two most common operations in our architecture, namely `setValue()` and `Reward()`. On a public Ethereum network, this amount of gas is directly translated into monetary cost, by setting the gas price T_p in Ether (ETH), the cryptocurrency of the Ethereum blockchain. Typically, such values are more concisely expressed in *gwei*, that is the ninth power of the fractional ETH(*i.e.*, $1\ gwei = 1.0 \times 10^{-9}\ ETH$ and, for this reason, also known as nano-ether). As a rule of thumb, higher values of T_p correspond to faster transaction processing times. However, to accurately estimate the response times of a blockchain network at different values of T_p is beyond the scope of this work. Nevertheless, the majority of Ethereum clients (also known as "wallets") presents three categories that relate T_p to transaction times (*i.e.*, the time needed for a transaction to be validated and included in the blockchain). Thus, 2-3 gwei is the typical gas price for a slow transaction time, 5-6 gwei is the typical range for average transaction time, and 10-12 gwei is more suitable for fast transaction time. For the sake of completeness, Table 6.6 translates the values of the system into USD, assuming an USD/ETHexchange rate of 205 (*i.e.*, $1\ ETH = 205\ USD$), based on the all time average price until July 2019, as reported by Etherscan[6].

Using $T_p = 5\ gwei$ we deployed a *Twin* contract in Ropsten[7]. Then, we evaluated the real transaction processing time for a executing the `setValue()` operation. We tested this operation, as it is the most frequent transaction in the architecture. We sent one transaction approximately each 30 minutes over a period of one week. The average blockchain processing time was 32

[6]`https://etherscan.io/chart/etherprice`

[7]at: 0x1449aeaaf3f18190b46b435b1258efb61257c71b

Table 6.6: Transaction costs for different values of T_p (slow, avg, and fast correspond to 2, 5, and 10 gwei, respectively, while 1 $ETH = 205$ USD is the exchange rate).

	Gas	**slow**	**avg**	**fast**
Twin Creation	143.947	0,059 USD	0,148 USD	0,295 USD
`setValue()`	26.821	0,011 USD	0,027 USD	0,055 USD
App Creation	3.343.572	1,371 USD	3,427 USD	6,854 USD
`Reward()`	156.580	0,064 USD	0,160 USD	0,321 USD

seconds, with a median of 21 seconds. Over this sample, only two transactions took more than 300 seconds to terminate (*i.e.*, less than 1% of the total).

6.5.5 Device power consumption and energy budget

For the boards hosting the full device module (namely, M0, PIC, M4L, and M4F), we measured the energy consumption while creating transactions, as described in Section 6.5.2. Before creating each transaction, we included an idle state of 8 seconds plus 5 seconds of sensing time, just as a reference. For measuring the energy, we used an Otii device[8] capable of measuring currents with accuracy of $\pm(1\% + 0.5\mu A)$ at 5V, with sampling rate of 1KHz (*i.e.*, 1000 samples per second). Figure 6.6 graphically depicts the average current consumption at 5V, while Table 6.7 shows the average energy consumption of each stage.

To estimate the energy impact that integrating blockchain has for constrained sensing device, we defined a simple energy budget model as follows:

$$\mathcal{E}_{daily} = \mathcal{E}_{idle} + \mathcal{E}_{sens} + \mathcal{E}_{tran} + \mathcal{E}_{block},$$

where $\mathcal{E}_{idle}$ is the energy of sensor in idle state, $\mathcal{E}_{sens}$ is the energy for sensing and processing a water consumption measurement, $\mathcal{E}_{tran}$ is the energy for

[8] `https://www.qoitech.com/`

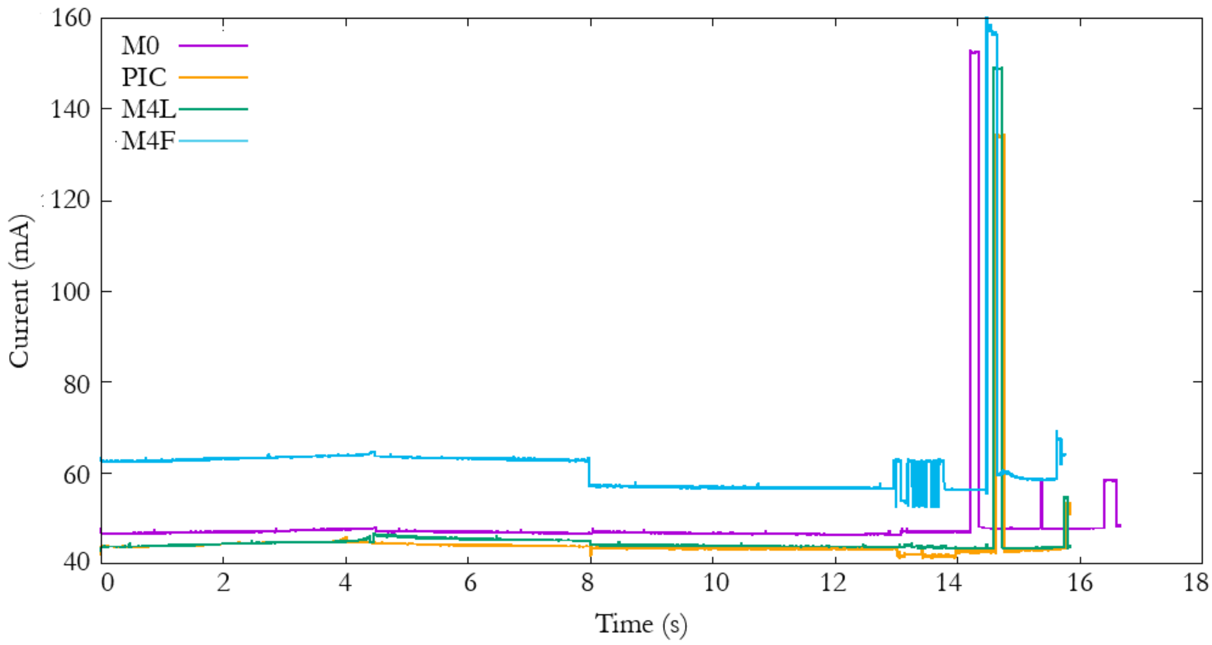


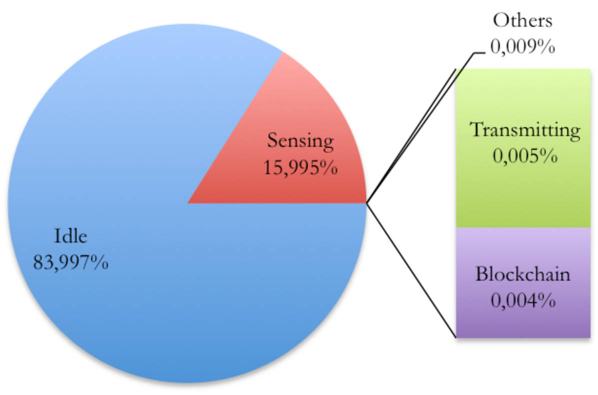
Figure 6.6: Average current consumptions at 5V.

Table 6.7: Average energy consumption of the device module at 5V (in Joules).

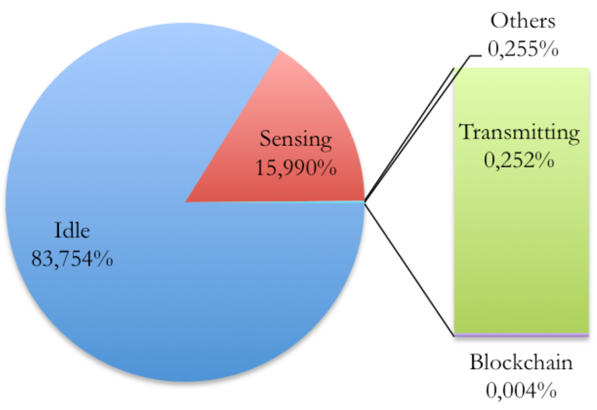
Device	**Idle**	**a) Sen**	**b) Enc**	**c) Hsh**	**d) Sig**	**e) Com**	**f) Trx**
M0	1,897	1,174	0,004	0,020	0,199	0,042	0,553
PIC	1,770	1,085	0,004	0,017	0,140	0,037	0,555
M4L	1,801	1,088	0,004	0,018	0,106	0,038	0,498
M4F	2,537	1,430	0,006	0,028	0,151	0,058	0,643

transmitting the water measurement, and $\mathcal{E}_{block}$ is the energy used for sending the daily usage to the blockchain. Though simple, this analytical model is aligned with the ones found in the current literature (see for instance [22]). To estimate the distribution of the energy budget in one day, we used our experimental results for time measurements (Table 6.5) and power consumption (Table 6.7).

However, defining and optimizing irrigation schedules (i.e., the sensing and idles times of the IoT device) is a challenging task. The schedule needs to consider administrative constraints (local policies, availability) and several



(a) Reactive Monitoring.



(b) Continuous Monitoring.

Figure 6.7: Estimated daily energy budget distribution for (a) reactive and (b) continuous monitoring.

parameters which are related to the field (e.g., size, type of crop, phenological phase, type of soil), the environment (average temperature, humidity, season), the irrigation system (e.g., type of installation, water flow)[128, 132]. For instance, the authors of [129] estimated that the watering schedules for different varieties of citrus fruit using a dripping watering system providing 4 l/h can vary from a minimum of 1 h/day up to 3.25 h/day depending on the month. On the other hand, authors of [134] reported that irrigation schedules for grapevines were carried every four days for 7 hours, following the existent administrative constraints. Since a precise watering schedule is a task beyond the scope of this research, we considered an average watering frequency of 2 h of watering twice a day. We also considered two different scenarios for the IoT system: reactive and continuous. More specifically, the energy budget for the reactive scenario considers that the measuring device only reports when using the valve. On the other hand, the continuous scenario considers a measuring device reporting every 15 minutes. These results are depicted in Figure 6.7.

6.6 Conclusions

In this chapter, we proposed an architecture based on IoT and blockchain to foster sustainable agriculture practices, by incentivizing and rewarding virtuous behaviors in water management. In our proposed architecture, constrained IoT devices are direct actors of a public blockchain network serving as trustworthy data sources for the smart contracts. We studied the impact that integrating blockchain capabilities has on the most constrained devices, typically employed for agricultural IoT deployments. By implementing a complete proof-of-concept of our proposal, we quantitatively evaluate this impact in terms of memory, program size, communications, and power consumption at the sensing device.The requirements of this proof-of-concept are perfectly aligned with those shown on similar works, where IoT is used in the agricultural domain without integrating blockchain [132, 135, 136]. For example, the data-flow presented in [135] can be easily adjusted to fit our proposal.

Our results have shown that off-the-shelf, cost-effective IoT devices can interact directly with a public blockchain even over Low Power Wide Area Networks (LPWAN). More concretely, 32-bit boards in the same price range of 8-bit microcontroller (*i.e.*, Arduino board), can seamlessly support the proposed architecture. Regarding energy budget, performing the blockchain operations required on average only an additional 6% of the energy concerning traditional operations without blockchain involvement. Moreover, even for a device reporting every 15 minutes while sending the water consumption to the blockchain once a day, the blockchain operations consume only 0.004% of the daily energy budget.

Future works include quantitative evaluations of the metrics related to the smart contracts supporting our framework, also including security-related considerations and a more formal security analysis. Smart contracts can also provide a starting point for an economic evaluation of our architecture. Finally, evaluating other communications technologies, such as Narrow-band IoT, could provide a better understanding of the possible alternatives to further develop the system.

Part III: Exploratory analyses of blockchain-based applications challenges

Chapter 7

Cost and user experience in blockchain-based applications

This chapter contains text taken from the published work:

– *M. Pincheira, M. Vecchio & R. Giaffreda (2020, July). Rationale and Practical Assessment of a Fully Distributed Blockchain-based Marketplace of Fog/Edge Computing Resources. In 2020 Seventh International Conference on Software Defined Systems (SDS) (pp. 165-170)*

This chapter is the first exploratory analysis on the challenges of blockchain-based applications identified from the previous two case studies. Its focus is the monetary cost and the impact on the user experience of using a permissionless network for all the functionalities of a blockchain-based application. Section 7.1 presents an architecture and the implementation of a blockchain-based application solely relying on a public blockchain network. To test its effectiveness, we evaluate several design decisions and trade-offs in terms of monetary cost versus time-response as a metric for the user experience. Section 7.2 proposes a cost model for estimating the monetary cost of a blockchain-based application. The model includes a transaction taxonomy, an application life-cycle, and a series of parameters that help to characterize blockchain-based applications and thus, estimate their monetary costs.

7.1 Rationale and practical assessment of a fully distributed blockchain-based marketplace of Fog/Edge computing resources

7.1.1 Introduction

A marketplace can be defined as a place where people gather for selling and buying goods. They can be traced back thousands of years, and they have been reinvented several times [137]. Throughout history, goods evolved from cattle, vegetables, and hunting tools, to machine learning algorithms, cloud resources, and IoT data, just to mention a few examples. And the place migrates from a physical location to a virtual software platform, better known as "electronic marketplace" or, more simply, *e-marketplace*. However, the main objective remains the same: a buyer with a specific need browses through several offers published by the sellers negotiating the purchase of the one that best fits. By nature, traditional marketplaces are open and heterogeneous playgrounds where the main users are buyers and sellers. In e-marketplaces, however, the digital platform is provided by a trusted third-party intermediary who has increasingly gained more importance in the process. As a matter of fact, in exchange for usability and simplicity, these centralized e-marketplaces have reduced the transparency and flexibility of the process, allowing rules and procedures to be set by the provider [26]. Furthermore, intermediaries may influence the interactions of participants by favoring some specific offers over others, or by silently pushing buyers towards specific sellers [138, 139].

In the last few years, an increasing number of studies have focused on blockchain as a possible solution for decentralized e-marketplaces [140, 141, 142]. Blockchain, the technology behind the Bitcoin and other cryptocurrencies, can provide immutability, transparency, and traceability to the interactions without the need for a third-party intermediary [11]. This technology can help to overcome issues such as validating the integrity of the offers, auditing the negotiations, and enforcing the conditions of the sales [143]. Finally, the decentralized architecture of blockchain networks removes the single-point of failure of the whole e-marketplace infrastructure [144].

However, despite the extensive use of blockchain in the financial sector, in terms of research and development, the technology is still in its early stages of maturity. This translates into limitations (*e.g.*, scalability, throughput, costs, etc.) that need to be wisely considered before deciding to integrate a blockchain-based solution into a production system. In the specific case of decentralized e-marketplaces, one of the main drawbacks the technology is currently facing is the usability and the user experience of such platforms [140]. To overcome these limitations, the current literature is pursuing two approaches: a) using *private* over *public* blockchain implementations, and b) including other services, such as decentralized file systems (DFS) and decentralized databases (DDB).While the first approach provides decentralization at the architectural level, it still relies on a controlling entity. On the other hand, the second approach provides full decentralization yet introduces a dependency on these new services. Moreover, from the perspectives of the user and the developer, such services may present additional usability issues. These factors raise several questions about design decisions and architectural choices for these blockchain-based e-marketplaces.

In this section, we propose a fully decentralized e-marketplace framework, based only on a public blockchain infrastructure. To evaluate its effectiveness, we consider the use-case of a marketplace of Fog/Edge Computing resources. As metrics for the user experience, we focus on monetary cost and system performance. For particular design decisions, we evaluate different scenarios and present a final version of the marketplace based on our evaluations. This thorough analysis provides guidelines for designing blockchain-based marketplaces and can be particularly useful when assessing the feasibility of adopting public blockchain implementations.

The remaining of this section is structured as follows: Subsection 7.1.2 provides a brief overview of similar works related to blockchain-based marketplaces. In subsection 7.1.3, we describe the proposed architecture, while subsection 7.1.4 delves into the parameter setup of our evaluation campaign. Subsection 7.1.5 presents the implementation details and the analysis of the obtained results. Subsection 7.1.6 presents final remarks and proposes some interesting hints for future research works.

7.1.2 Related work

In recent years, there has been a growing interest in blockchain technology as a fabric for creating decentralized e-marketplaces. In general, while some studies have been conducted on private blockchain implementations [26], fewer have dealt with smart contracts on public blockchain implementations. In the context of cloud services marketplaces, the authors of [145] study the negotiation of service level agreements. However, this work provides only a use-case example, without deploying it on a real blockchain. Similarly, the authors of [143] use the web service agreement negotiation specification to define a custom blockchain. Nonetheless, they evaluate their proposal only in a simulation environment, without implementing any smart contract. An Ethereum-based green certificate marketplace is presented in [146], where the authors propose an architecture based on a single smart contract that is later deployed and tested. They present metrics regarding the cost of different pricing strategies for buying and selling certificates and compare these monetary results with a traditional centralized marketplace. Similarly, the authors of [142] present a system for renewable energy auctions. Their architecture is based on a single smart contract and they also focus on the auction prices. However, neither of these works address other marketplace's functionalities, such as creating and managing the market itself.

A general-purpose marketplace is presented in [139], where the authors propose, implement, and test a system based on a hybrid infrastructure combining Ethereum and a decentralized file system (DFS). The data of products are stored in the DFS, while the Ethereum network is used to store a reference to such data and to support the bidding process. This work presents metrics regarding the performance of the application based on gas consumption and transaction time. A similar hybrid architecture is presented in [140], where the authors present a thorough analysis of the data and associated costs for a marketplace of IoT data in a smart city context. However, both works miss to provide any evaluation of the overhead time required to create and browse the existing products using the proposed hybrid approach.

More aligned to our proposal is the Ethereum-based general-purpose marketplace presented in [144], where the authors first describe five functionalities of a marketplace, and then focus on the evaluation of two of them, namely “offer creation”, and “offers query”. More in detail, they propose a hybrid centralized/decentralized architecture where the consumer and the provider access their own type of nodes. The search is based on a smart contract, but it also relies on an external array maintained in the so-called “consumer node”. To evaluate the performance of this system, the authors deploy a private Ethereum network with three nodes, namely a miner, a provider, and a consumer. Then, a smart contract for adding offerings is added to the chain. Their results show quite good performance from the user perspective: in particular, a list of 4096 offers is obtained in less than 20 seconds. However, they miss to provide details about the smart contract, the offerings (*e.g.*, fields, size), as well as the time needed to create and maintain the external array. Last but not least, this approach is not fully decentralized, so if one consumer node was off-line, then a subset of consumers would not be able to access the marketplace.

On in all, these works highlight the critical task of evaluating the design decisions for a fully distributed system architecture, especially from the smart contracts perspective and the impact they can have on the user experience when using a public blockchain network.

7.1.3 Proposed system architecture

We propose a fully decentralized e-marketplace framework that relies solely on a public blockchain with smart contract capabilities. Regarding the application domain, we focus on the requirements of an e-marketplace of Fog/Edge Computing resources, as part of an ongoing project funded by the European Commission, namely "DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence" [1]. Although the proposed software architecture (depicted in Figure 7.1) takes into account very specific requirements of this project, it can seamlessly accommodate requirements of other domain-specific marketplaces we have found in the literature (*e.g.*, [140, 139, 144]). In our framework, a centralized infrastructure is completely replaced by a blockchain network and any administrative task is performed by a single market smart contract deployed in the network. It follows that the only two types of actors of this system are buyers and sellers, unequivocally identified in the blockchain by unique addresses. Thus, issuing a transaction from one address can be considered as the proof of ownership of that identity, while keys distribution and management issues are beyond the scope of this work.

We define a *marketplace* M as a collection of *advertisements* A_i such that $M = \{A_1, A_2, \dots, A_l\}$,where $l \in \mathbb{N}$ is the size of the marketplace. M can be represented by a single Market Smart Contract MSC, deployed in the blockchain by a generic actor X of the system. It is important to notice that X does not have any special privilege with respect to any other actor of the system; for instance, M can be instantiated by a seller (or by a group

[1] `https://www.decenter-project.eu/`

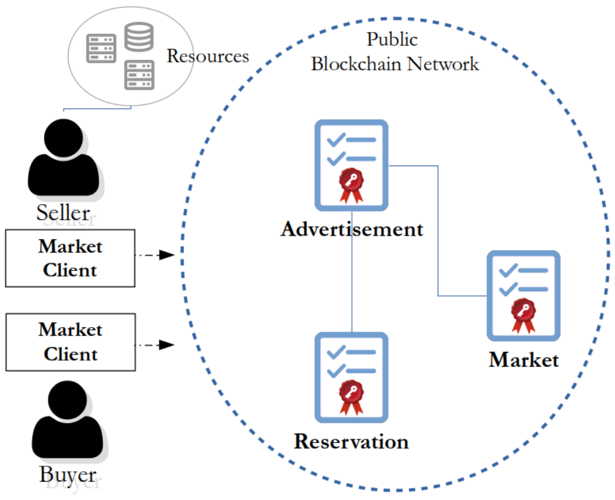


Figure 7.1: The proposed fully-distributed e-marketplace software architecture.

of sellers) or a buyer (or by a group of buyers). Each advertisement A_i represents a *resource* r_i belonging to a particular seller and is described by a finite set of fields $D_i = \{d_1, d_2, \dots, d_m\}$, where $m \in \mathbb{N}$. Moreover, each A_i is uniquely represented by its corresponding advertisement smart contract ASC_i deployed in the blockchain. It is worth to notice that this definition can be used for any type of resources, such as goods (*e.g.*, renewable energy, IoT data, etc.) and even services (*e.g.*, AI models, algorithms, etc.).

A buyer can browse the marketplace advertisements, choosing a specific $A_i \in M$ based on the values of D_i. If interested to A_i, he creates a *reservation* RA_i representing the offer for the resource advertised. At its turn, RA_i is described by a set of fields $G_i = \{g_1, g_2, \dots, g_n\}$, where $n \in \mathbb{N}$. If such an offer is accepted by the seller, then the agreement between the two parties is represented by a unique reservation smart contract RSC_i, also deployed in the blockchain. At this point, the process can be considered as finalized.

All possible interactions between actors and components of the proposed architecture can be grouped into seven functionalities ($F1$-$F7$, as shown in Figure 7.2). These functionalities are mapped onto four software components which are described in the following. It is worth to notice that $F1$ is only a transaction necessary to deploy the first contract and, for this reason, it does not belong to any specific component.

The Market Client (*CLI*)

it provides an application interface (API) to interact with the blockchain and access all the functionalities of the marketplace ($F2$-$F7$). Any off-chain processing and off-chain storage is done by this module, without having to rely on external services (*e.g.*, DFS).

The Market Smart Contract

it can be considered as the manager of the market. However, it cannot influence the market users' interactions. The main scope of this component is to act as a registry for the existing contracts ($F4$). Moreover, it is used as a factory for creating new advertisements ($F2$); this guarantees that all advertisements will have the exact same code. This smart contract can implement more complex logic holding the potential of converting a decentralized e-marketplace into an Autonomous Decentralized Organization (DAO) managing not only advertisements, but also reviews, scores, and reputations.

The Advertisement Smart Contract

it can be considered as the main component of the system since it contains the details of an advertisement ($F5$) and the method to update such details ($F3$). It also contains the logic to receive reservation requests ($F6$) and to accept/reject them ($F7$). The advertisement is fully managed by the seller, therefore the latter cannot be influenced by any third-party.

The Reservation Smart Contract

at current development stage of the project, a purchase is considered finalized with the creation of this smart contract. Thus, for the sake of simplicity, it can be considered as an escrow agreement. However, in principle, it could provide more complex logic (*e.g.*, automatic verification of an agreement, negotiation in case of failure of an agreement, etc.).

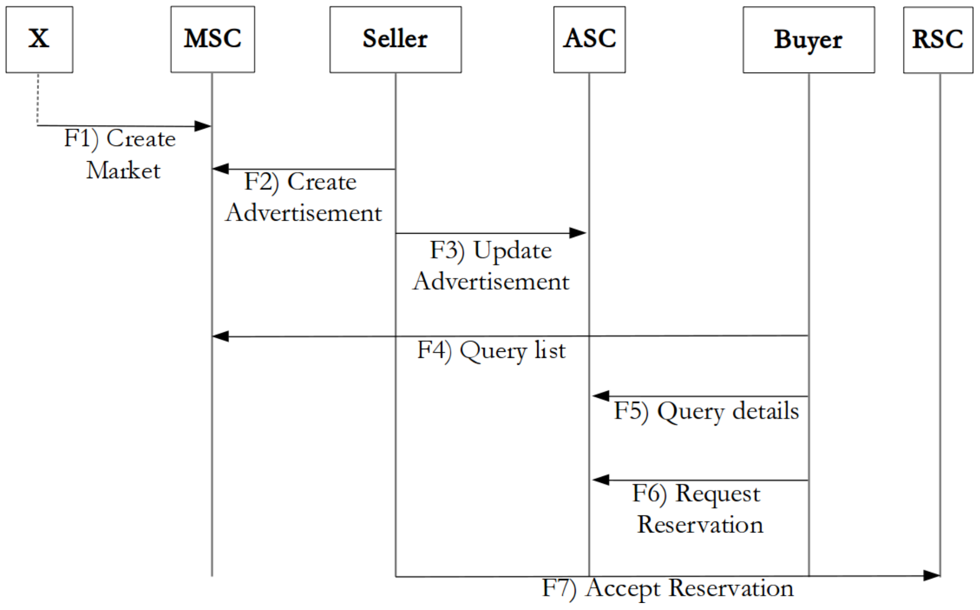


Figure 7.2: Sequence diagram of the interactions within the proposed e-marketplace.

7.1.4 Experimental setup

To quantitatively assess the benefits of the proposed architecture, we tailored our design to the Ethereum blockchain, being it the second largest public blockchain network and globally considered as a reference regarding scripting capabilities. The experiments were performed using a DELL Latitude E5470 laptop endowed with 4x Intel Corei5-6440HQ at 2.60GHz, 8GB RAM, 256GB SSD Disk and running a Linux Ubuntu Operating System (16.04.6 LTS). Regarding the software stack and tools, we implemented the *CLI* using Python 3.6, the web3 library as an interface for the Ethereum network, the flask library to provide a public REST API, and the gzip utility for file compression. To evaluate the time response of a smart contract we used Ganache (version 2.0.1), that is part of the Truffle suite [2]. This software provides the same results as sending a transaction to a real Ethereum node, without the network delay and the transaction processing time needed for the mining process. To assess the network performance we run a node of the Ethereum Ropsten[3] test network, using the official Ethereum client geth (version 1.9).

Basic data model parameters

Based on the requirements of our project, we set $m = 60$ for the advertisement description, meaning that D is composed of 60 fields (*e.g.*, unique identifier, type of resource, region, seller, status). For the *reservation* description G, we considered $n = 30$ on average. As done in similar works [140], D and G are encoded using JSON format in order to ease integration with the other components of the system (*e.g.*, user interface). For the experiments, random data is generated for all the fields, based on real examples of G and D, thus the size of D and G refers to the total bytes used by data encoded using JSON.

Reference cost for Ethereum

To obtain a reference monetary cost, we fixed a conversion rate C_E from ETH to USD equal to 205 USD. We obtained this value as the average daily ETH price until July 2019, as reported by Etherscan[4]). We also set a gas price T_p of 10 gwei, which is the value recommended by Metamask[5] for fast transactions processing.

[2]`https://truffleframework.com/ganache`
[3]`https://ropsten.etherscan.io/`
[4]`https://etherscan.io/chart/etherprice`
[5]`https://metamask.io`

7.1.5 Architecture evaluation

Cost evaluation

We evaluated the cost of our framework with a bottom-up approach, starting from the reservation as the basic unit of the system. Any design decision taken on the reservation will have a direct impact on the cost of both the advertisement and the market. For testing purposes, all contracts are created by calling a method contained in another smart contract that acts as a factory.

Cost of RSC

Our reservation is based on the escrow contract provided by OpenZepellin[6], that is an industry-approved library for secure smart contract development. This base escrow also stores the reservation details G as raw bytes on a single variable. Since such information does not need to be processed internally by the RSC, it can be also stored in compressed format, using CLI to perform this off-chain task. We evaluated these two scenarios (*i.e.*, compressed and normal) by creating 100 reservations with random data generated for G. Table 7.1 depicts the cost in gas and a reference cost in USD, according to the set conversion rates. The average size of G was 1585 and 653 bytes for the normal and compressed case, respectively.

Table 7.1: Average cost for creating reservations ($T_p = 10\, gwei$ and $C_e = 205\, USD$).

	Normal		**Compressed**	
	Size	***USD***	***Size***	***USD***
Create	1793288	3,7	1144229	2,3

[6]`https://github.com/OpenZeppelin/openzeppelin-contracts`

Cost of ASC

To implement the advertisement smart contract, we considered three levels of on-chain processing. From all the existing fields $d_i \in D$ we select a subgroup $D' \in D$ that could be used by the smart contract to provide additional functionalities on the market. The size of D' translates into three designs alternatives for the ASC according to the number of fields that are stored as variables and accessible by the contract, namely:

a) *Maximum*: D' is composed of 15 fields (including 3 dynamic lists), while the remaining 45 fields are stored as raw bytes using a single variable.

b) *Medium*: D' is composed of 10 fields (including 1 dynamic list), while the remaining 50 fields are stored as raw bytes using a single variable.

c) *Minimum*: D' is composed of 5 fields, while the remaining 55 fields are stored as raw bytes in a single variable.

Then, we evaluated the three alternatives (namely, a, b, c) by creating and updating 100 advertisements with random data generated for D. Table 7.2 shows the average cost in gas and a reference cost in USD. The size of D was 2522 and 1007 bytes for the normal and compressed case, respectively.

Cost of MSC

The final cost evaluation of our system is for the MSC, which depends on both the RSC and the ASC. However, this contract also needs to maintain a registry R of all the existing advertisements based on the unique identifier (d_{uid}) and the blockchain address (d_{add}) of each advertisement. Since this registry can be queried, we consider only two types of possible queries, namely a) All and b) Filtered. The *filtered query* is simply based on the region $d_{reg} \in D$, leading to three alternatives to implement this registry $R = \{d_{uid}, d_{add}, d_{reg}\}$ in the MSC:

Table 7.2: Average cost for creating and updating advertisements ($T_p = 10\,gwei$ and $C_e = 205\,USD$).

	Normal		**Compressed**	
a) Maximum	***Gas***	***USD***	***Gas***	***USD***
Create	4151378	8,5	4040785	8,3
Update	724653	1,5	488352	1,0
b) Medium	***Gas***	***USD***	***Gas***	***USD***
Create	3460243	7,1	2877232	5,9
Update	661437	1,4	270337	0,6
c) Minimum	***Gas***	***USD***	***Gas***	***USD***
Create	3352120	6,9	2467789	5,1
Update	587023	1,2	263409	0,5

d) On-Chain Registry: the full registry $R\ (d_{uid}, d_{add}, d_{reg})$ is stored as fields in the contract. This alternative allows the MSC, for instance, to detect when a d_{uid} is repeated, and also to filter the registry without off-chain processing.

e) Off-Chain Registry: we use events to store R, which is a cheaper alternative for storage purposes. In this case, the contract cannot detect if an d_{uid} is repeated, neither it can filter the registry. This processing must be done off-chain, since smart contracts cannot access event information.

f) Hybrid Registry: we minimize the internal storage, so it is used only to detect if an d_{uid} is repeated. We use events to store the rest of the registry and the filtering is done off-chain.

Finally, using the three different types of ASC (namely, a, b, c) from subsection 7.1.5, we could evaluate the three market alternatives (namely, d, e, f). Table 7.3 shows the average results of 100 tests performed for each type of ASC and MSC.

Table 7.3: Average transaction cost for creating the marketplace ($T_p = 10$ gwei and $C_e = 205$ USD).

	d) On-Chain		**e) Off-Chain**		**f) Hybrid**	
	GAS	**USD**	**GAS**	**USD**	**GAS**	**USD**
a) Maximum	5830842	12,0	4939778	10,1	5070884	10,4
b) Medium	4797904	9,8	3888487	8,0	4019514	8,2
c) Minimum	3807595	7,8	2897972	5,9	3032280	6,2

Performance evaluation

We evaluated the performance of our framework in terms of time response for getting all the advertisements from a deployed market. This process is divided in two steps: 1) Query the list of existing advertisements from the marketplace ($F4$), and 2) Query the details of each advertisement ($F5$). We evaluated all the queries considering seven different sizes of the market $l = [2^7, 2^{13}]$, evenly distributed over 5 regions (for filtering purposes). This values were chosen considering the scenarios presented on current literature.

Query list

Using the three versions of MSC described in subsection 7.1.5, we could measure the total time needed for retrieving the complete list or a filtered list from the MSC ($F4$). The average time of 100 tests is shown in Table 7.4 and depicted in Figure 7.3. It is worth to notice that, when $l > 1024$, the MSC reaches the gas limit before processing the entire registry R.

Query details

We measured the total time needed for retrieving the details of all the existing advertisements ($F5$), according to the list obtained from the previous step ($F4$). The average time of 100 tests for each market size is shown in Table 7.5

Table 7.4: Average time response (in seconds) for querying the list of advertisements from the MSC.

	d) On-Chain		**e) Off-Chain**		**f) Hybrid**	
size(l)	***All***	***Filter***	***All***	***Filter***	***All***	***Filter***
128	0.78	0.93	0.12	0.13	0.13	0.12
256	1.34	1.88	0.21	0.21	0.21	0.21
512	2.64	3.71	0.35	0.34	0.35	0.36
1024	5.33	7.56	0.65	0.69	0.72	0.69
2048	11.35	-	1.31	1.31	1.32	1.33
4086	23.06	-	2.67	2.62	2.69	2.67
8192	46.01	-	5.65	5.63	5.54	5.55

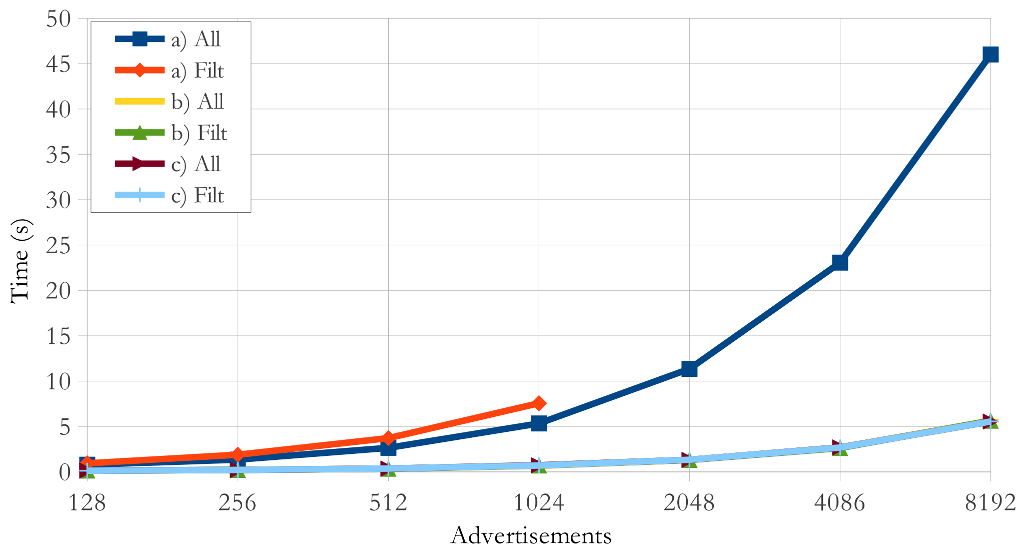


Figure 7.3: Average time response (in seconds) for querying the list of advertisements

and depicted in Figure 7.4.

Local buffers for queries

The major delay on both queries, is the network call to access the blockchain, and not the time needed by the smart contract. We implemented a buffer in the CLI in order to reduce the network calls. Thus, the time displayed

Table 7.5: Average time response (in seconds) for querying details of all the advertisement.

	a) Maximum		**b) Medium**		**c) Minimum**	
size(l)	***All***	***Filter***	***All***	***Filter***	***All***	***Filter***
128	15.4	3.8	15.9	3.3	14.4	3.1
256	29.2	7.4	28.7	5.9	28.3	5.8
512	59.2	15.1	57.5	11.8	57.3	11.8
1024	118.9	30.1	116.1	23.5	114.2	23.4
2048	257.8	-	231.9	47.5	229.8	46.5
4086	503.7	-	470.1	96.1	465.2	95.1
8192	980.1	-	975.7	198.9	966.9	196.5

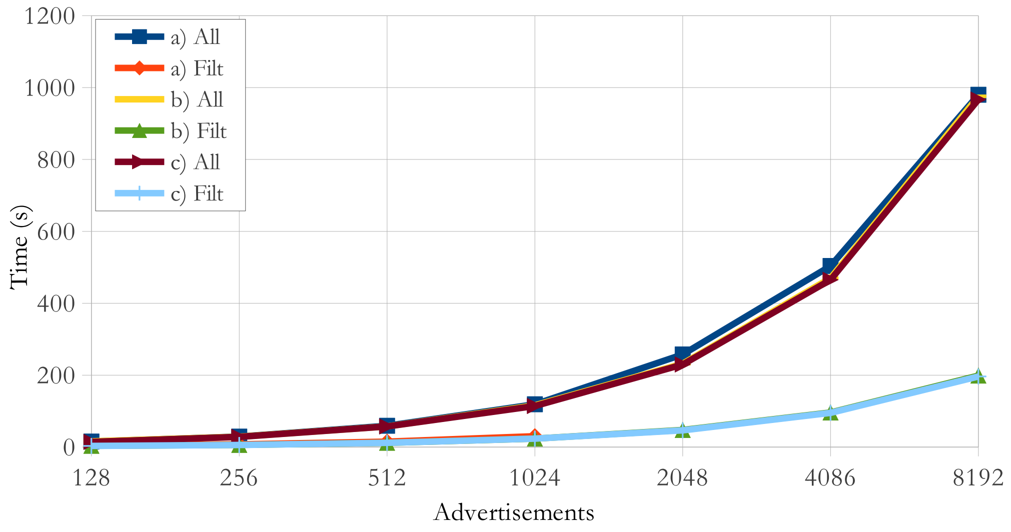


Figure 7.4: Average time response (in seconds) for obtaining the details of all advertisements.

in Table 7.5 becomes the initialization time for CLI, enabling later calls to be answered by the local buffer. New advertisements and updates are monitored by the client on background. To avoid out-dated information, for example when creating reservation, the selected advertisement is always retrieved from the blockchain before any write operation. Using this buffer, we repeated the experiments described in subsection 7.1.5. The results are summarized in Table 7.6 and displayed in Figure 7.5.

Table 7.6: Average time response (in seconds) for querying the detailed list of advertisements using local buffer.

	a) Maximum		**b) Medium**		**c) Minimum**	
size(l)	***All***	***Filter***	***All***	***Filter***	***All***	***Filter***
128	0.007	0.004	0.006	0.005	0.005	0.005
256	0.014	0.007	0.012	0.008	0.011	0.007
512	0.024	0.011	0.025	0.011	0.025	0.011
1024	0.043	0.022	0.045	0.021	0.051	0.021
2048	0.091	0.036	0.089	0.032	0.096	0.041
4086	0.173	0.071	0.172	0.069	0.181	0.075
8192	0.379	0.133	0.383	0.128	0.404	0.135

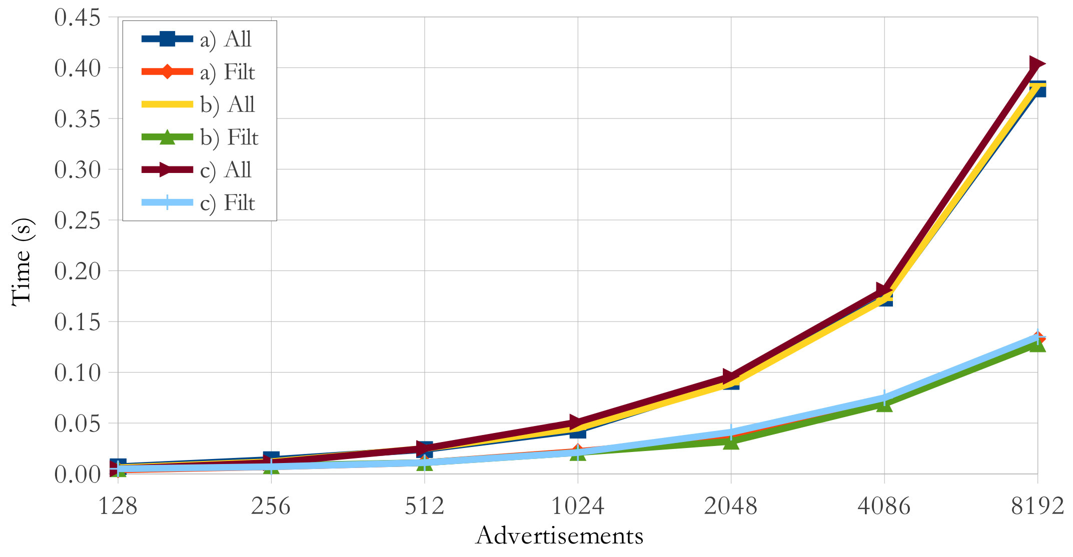


Figure 7.5: Average time for all advertisements using local buffer.

Network processing times

As a final step, we combined all the selected elements into the final version of the architecture. We evaluated the processing time for the transactions on a live network. To this aim, we deployed a single Hybrid MSC [7] and 128 Minimum ASC on Ethereum Ropsten network. Each advertisement was also updated, and a reservation was requested and accepted on it. For a better reference under different network conditions, the advertisements were created approximately every 10 minutes during a period of 24 hours. Table 7.7 shows a summary of the processing time and cost for each of the 7 market functionalities. It is important to notice that queries $F4$ and $F5$ do not require transactions, hence do not have associated costs nor require processing times in the network.

Table 7.7: Average processing time (in seconds) and cost of the functionalities in a live network ($T_p = 10\, gwei$ and $C_e = 205\, USD$).

	Functionalities	***Processing Time***	***Cost USD***
F1)	Create Market	20s	6.2
F2)	Create Advertisement	22s	5.1
F3)	Update Advertisement	23s	0.5
F4)	Query List	0s	0
F5)	Query Details	0s	0
F6)	Request Reservation	23s	0.1
F7)	Accept Reservation	22s	2.0

7.1.6 Conclusions and future works

In this section, we have proposed a fully decentralized marketplace where smart contracts alone provide all the functionalities needed by the application, without the need for additional services (*i.e.*, storage). To validate the proposed framework, we have developed a full-fledged e-marketplace of Fog/Edge Computing resources in the context of an ongoing EU research project. We evaluated several design decisions in terms of cost and perfor-

[7]available at :0xFED47d92904602cCb7324Dc67dF1A2E833E7a9D7

mance as metrics for assessing the user experience. Our results have shown that a fully blockchain-based marketplace can be created at the cost of around 6 USD and the different operations costs range from few cents to 5 USD. On a live network, these costs translates into processing times of around 20 seconds, which is among the fastest processing times the network is currently capable of. When browsing the marketplace, a detailed list of over 8000 advertisements could be obtained in less than 1 second with minimal off-chain computation and without off-chain storage. Compared to the state-of-the-art counterparts, our fully decentralized architecture provides response times within the same range as those provided by hybrid architectures. These encouraging results validate the effectiveness of the proposed architecture and clearly define thresholds in terms of performance and cost within which our fully decentralized marketplace can operate to also guarantee seamless user experience. We expect that these constraints and thresholds can be used as guidelines for designing smart contracts within different application domains.Future works include evaluating further complex smart contracts at the reservation level, enabling more elaborate functionalities such as automatic negotiation and automatic conflict resolution.

7.2 Cost Model for blockchain-based applications

7.2.1 Introduction

Current models studying blockchain-based application focus on factors such as scalability, security, and performance. However, they fail to provide a monetary cost analysis, particularly when a permissionless infrastructure is required.

In this section we present a cost model for the infrastructure of a blockchain-based application on permissionless networks. To this aim, we propose a transaction taxonomy and an application life-cycle to describe a typical blockchain-based application. This taxonomy and life-cycle can be used to provide an overview of the system at the design stage. To illustrate the usability of the model to quantitatively analyse the cost and benefits of the blockchain-base application, we study the application described in Section 7.1.

The model is a step towards identifying the diverse factors characterizing blockchain-based applications in a public blockchain network and the costs behind such applications.

General overview of the model

The proposed model consider interactions among actors (i.e., transactions) as the functional unit to provide Life Cycle Assessment (LCA) of the application. Given a group of actors A and a group of stake-holders S, the cost of the blockchain-based application CI is given by the infrastructure needed to support the interactions I of the actors. These interactions generate value units K which are are the benefits BK for the stake-holders. Thus, for a given month m we define the cost of the system as $C(m)$ as the cost of the interactions CI over an infrastructure supporting the blockchain-based application. These interactions are expected to a generate a value BK which represents the benefits for the stakeholders in that given month $B(m)$ such that:

$$\sum_{m=1}^{n} C(m) = CI \longrightarrow BK = \sum_{m=1}^{n} B(m) \tag{7.1}$$

Our model propose a series of key parameters which allow to characterize the application and the interactions of their actors. To better understand these parameter, we first need to define what we consider a typical blockchain-based application. To this end, subsection 7.2.2 describes the requirements of these types of applications and the life-cycle that will provide the scope of our analysis. Based on these definitions and requirements, subsection 7.2.3 proposes a transaction taxonomy to generalize the interactions of the actors, making the model usable in a wider range of applications and domains.

7.2.2 Application description and requirements

In our model, we consider that the application supports the interactions of a group of unknown actors, identified only by the use of private/public keys. Anyone with a set of keys is considered an actor after being registered in the application. Actors have a limited group of predefined interactions that aims to create and transfer value among them. Finally, all the information in the application is immutable, auditable, and accessible by anybody. To simplify the analysis, we consider that the blockchain-based applications have the four requirements detailed on Table 7.8.

Similar to the analysis presented on [52], we consider that these blockchain-based applications have a two-phase life-cycle: Bootstrap and Operation. It is important to highlight that we do not consider the costs of developing the software nor the cost of updating the software. We only focus on the cost of the infrastructure to support the interactions of the actors.

Table 7.8: Basic requirements of blockchain-based application on permissionless networks.

	Requirement
$R.1$	All the application logic is on the blockchain
$R.2$	Actors are unknown and identity is proven by digital signatures
$R.3$	Actors need to be registered in the application
$R.4$	Actors interact to generate and transfer value
$R.5$	Information is public, immutable, and auditable

7.2.3 Proposed transaction taxonomy

Interactions among actors on a blockchain-based application are represented by transactions. From the group of all possible interactions I, our model focus only in a subset of transactions $Tx \in I$ that create new information for the application. Based on the basic requirements presented on Table 7.8 and the life-cycle of the application, we proposed $CRIV$, a simple taxonomy to classify the transactions. $CRIV$ categorizes the transactions into 4 types of: Creation (Tx_C), Registration (Tx_R), Interaction (Tx_I), and Value (Tx_V). Figure 7.6 shows the $CRIV$ taxonomy in reference to the life-cycle phases of the blockchain-based application.

- **Creation Transaction (Tx_C)** These are the transactions needed to deploy the application logic into the blockchain. Tx_C may include one or many transactions and are expected only during the Bootstrap phase.
- **Registration Transaction (Tx_R)** These transactions are required when an unknown user interacts with the application for the first time, making the user an the actor of the application.
- **Interaction Transaction (Tx_I)** Are the most common types of interactions between actors on the application. These types of transactions produce information that needs to be kept in the blockchain, but do not include the transfer of value.

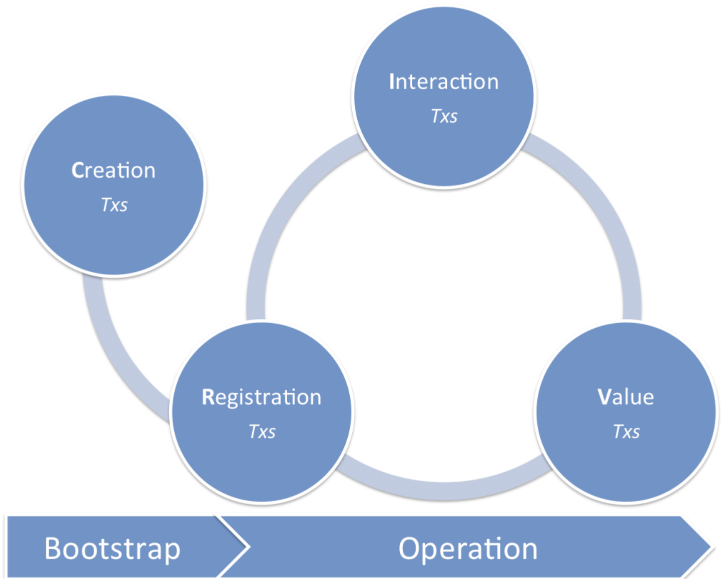


Figure 7.6: Life-cycle of a blockchain-based application

- **Value Transaction (Tx_V)** These type of transactions are the most important on the system as they include the transfer of value between unknown actors on a decentralized system.

7.2.4 Cost model for permissionless networks

On permissionless blockchain networks (i.e., Bitcoin [40], Ethereum [46]), transactions that create new information (i.e., modify the blockchain state) have a monetary cost. The cost is expressed in terms of the cryptocurrency used in the blockchain and pays for the use of the public network infrastructure. We divided our proposed cost model into two components,i.e., C_B and C_O one for each phase of the application life-cycle. Using the proposed taxonomy, the bootstrap component C_B includes Tx_C and Tx_R for the initial actors. Conversely, the operation component C_O considers Tx_R for other new actors, Tx_I, and Tx_V. This component is evaluated every month during the operation phase. Table 7.9 summarize all the parameters of our model, which are described in the following paragraphs.

Table 7.9: Parameters for the cost model of infrastructure using a permissionless blockchain.

Parameter	**Description**
A_0	Number of expected initial actors
$P(m)$	Price of the crypto-currency
μ	Time factor for processing transactions
CC_C	Computational cost of creation transactions
CC_R	Computational cost of registration transactions
CC_I	Computational cost of interactions Transactions
CC_V	Computational cost of value Transactions
F_g	Factor of growth
F_i	Factor of interactions for the actors
F_v	Factor of value transfer
F_k	Value unit factor
V_s	Total Monetary value of a value-unit

Therefore, in any given a month m where $m = 0$ is the bootstrap phase, and $m > 0$ is a month on the operation phase, we define the cost of permissionless blockchain network infrastructure $C(m)$ as:

$$C(m) = \begin{cases} C_B, & \text{if } m = 0 \\ C_O(x) & m > 0 \end{cases} \tag{7.2}$$

For both phases the costs are the fees required to execute the transactions on the public network. The fee links the computational cost CC of a transaction with the cryptocurrency of the network P using a processing time factor μ. Given the high volatility of the cryptocurrencies price, an accurate estimation for P is a very complex task beyond the scope of this work. In our model, we reflect this variation using a function $P(m)$ for the price at a given month m. This function should be define by the user of the model, according to the particular application implementations. The values of the function can be fixed (e.g., the average for a year), a fixed set of values (e.g., based on historical monthly prices), or an estimation function. The factor μ increases the price paid for a transaction, which typically translated into faster processing times since higher fees are more attractive to the node operators. Using the $CRIV$ taxonomy, for each transaction $Tx \in \{TxC, TxR, TxI, TxV\}$, the total cost in a month m is given by:

$$C_{Tx}(m) = P(m) \cdot \mu \cdot CC_{Tx} \cdot Q_{Tx} \tag{7.3}$$

Where Q_{Tx} is the total number of transactions, and CC_{Tx} as the computation of that type of transaction. Thus, the cost of the bootstrap phase is given by:

$$C_B = C_{TxC}(0) + C_{TxR}(0) \tag{7.4}$$

Where the first component is the cost to create the application on the blockchain and the second component is the cost to register the initial actors of the system A_0.

Similarly, the costs of the operation phase at a given month m is defined by:

$$C_O(m) = C_{TxR}(m) + C_{TxI}(m) + C_{TxV}(m) \tag{7.5}$$

To calculate the costs using these equations, we need to define how to estimate the number of transactions $Q_{Tx}(m)$. Since the number of transactions is directly related to the number of actors in the system, we first define this parameter as $A(m)$ such that:

$$A(m) = \begin{cases} A_0, & \text{if } m = 0 \\ A(m-1) \cdot F_g & m > 0 \end{cases} \tag{7.6}$$

Where A_0 is the initial numbers of actors in the system, and F_g is the expected monthly growth of the system (in terms of actors). Therefore, the number of a transactions $Q_{Tx}(m)$ for each type of transaction T is given by:

$$Q_{Tx}(m) = \begin{cases} 1 & \text{for } Tx = Tx_C \text{ and } m = 0 \\ 0 & \text{for } Tx = Tx_C \text{ and } m > 0 \\ A(m) - A(m-1) & \text{for } Tx = Tx_R \\ A(m) \cdot F_i & \text{for } Tx = Tx_I \\ A(m) \cdot F_v & \text{for } Tx = Tx_V \end{cases} \tag{7.7}$$

Q_{Tx_C} is 1 in the bootstrap phase and 0 for the operation phase, Q_{Tx_R} is the number of new actors, Q_{Tx_I} is obtained from all the actors using a factor of interaction F_i, and Q_{Tx_V} is obtained using a factor of value transfer F_v.

As show on Equation 7.1, the costs of a blockchain-based application (CK) is the cost of the infrastructure to support the interactions I among the actors A. These interactions are expected to produced a certain value VK which provides benefits to the stake holders. Therefore, the benefits of a given month m is defined by:

$$B(m) = F_k \cdot Q_{Tx_V}(m) \cdot V \tag{7.8}$$

Where F_k is the factor of value units in a transaction of type Tx_V, Q_{Tx_V} is the number of transactions of type Tx_V, and V is total monetary value of a value-unit. This monetary value is the sum of all the monetary value assigned by each stake-holder S (i.e., the benefit for each stakeholder).

Thus, to estimate the cost and benefits of a blockchain-based application, we need to define the parameters on Table 7.9 which helps us to characterize the application and its behavior in a given time-frame.

Parameters specifications

The proposed cost model is blockchain-agnostic. To estimate the costs and benefits of a blockchain-based application, the parameters of model need to adjusted according to the particular blockchain implementation selected for the application. For our evaluation, we choose Ethereum, as it is considered a reference implementation for smart-contracts [89] and migrating the model to other implementations should not present a major issue.

- The price of the cryptocurrency $P(m)$ is the price of Ether (ETH) which is quite volatile. However, historic values can provide a good reference for evaluating different scenarios. For Ethereum, the historical price (and many other statistics) can be obtained from several websites (e.g., Etherscan [8], Coinmarketcap [9])

- The computational cost of the interactions correspond to the gas need to execute a transaction on the Ethereum network. The gas needed is defined by the computational operations performed by the transaction (i.e., write a byte, transfer cryptocurrency). Every computational operation has a predefined gas usage [42]. Therefore, this computational cost can be obtained from the smart contracts that implement the the logic of the application.

- If the smart-contracts are not yet designed, some reference values can be estimated for the computation cost. Recent studies on the smart-contracts currently deployed on the Ethereum, shown that the majority of these applications are very similar to each other [89, 147]. In fact, 1/3 of the contracts are Token applications cloned from openZepellin [10], a community-approved library for secure smart contract development.

[8]https://etherscan.io/chart/etherprice
[9]https://coinmarketcap.com/es/currencies/ethereum/
[10]https://github.com/OpenZeppelin/openzeppelin-contracts

Thus, if the smart contract are not yet implemented, codes from this application library (e.g., ER20 Token, ERC721 Token) can provide a very good reference for the computation cost of the interactions.

- The μ parameter on Ethereum can be directly mapped to the gas price. Higher gas price translates into faster processing time for the transaction, however, there is no direct relation that provide a time reference. On average network condition (i.e., number of actors and number of transactions on the network), values of gas around 10-20 provide an average of processing time within the limit of the network ([148, 149]. However, events such an initial coin offering (ICO) will create higher traffic and thus, an increase of the gas value needed to achieve that average processing time [150].

7.2.5 Evaluation of the marketplace application

To test our model, we study the fully distributed blockchain-based marketplace for Fog-Edge computing resources (FECR) presented in Section 7.1. In this application, the group or unknown actors A is composed of both buyers and sellers of FECR. The unit of value is the FECR, expressed as a SLA (Service level agreements) between the buyer and the seller. In this case, the stake-holders are a group of sellers (service providers) wanting to avoid an intermediary to provide the marketplace functionality. For the marketplace, seven different transactions describe all the possible interactions among the actors (see section 7.1.3). Table 7.10 shows these transactions with the respective gas cost and the classification in the $CRIV$ taxonomy. From the table we can obtain the values for CC_c, CC_r, CC_i, and CC_v. It is important to notice that functions F1.4 and F1.5 are not considered in the taxonomy, as they read-only transactions.

For the other parameters, we will estimate an scenario based on similar

Table 7.10: Basic transactions for a marketplace classified using the CRIV taxonomy.

	Transaction	Estimated Gas	CRIV
F1.1)	Create Market	3032280	TxC
F1.2)	Create Advertisement	2467789	TxR
F1.3)	Update Advertisement	263409	TxI
F1.4)	Query List	0	-
F1.5)	Query Details	0	-
F1.6)	Request Reservation	63681	TxI
F1.7)	Accept Reservation	1144229	TxV

marketplace use case (subsection 7.1.2). We assumed a initial market size of 100 users (A_0) with a monthly growth of 1% (i.e., factor of growth). We estimated that all the actors will perform 2 interactions per month (i.e., factor of interaction). We assumed that 50% of the actors will rent their resources on any given month (i.e. factor of value transfer). The value-units are SLA between buyers and sellers for a FECR rented for a month at 60 USD. For the stakeholder (service provider) we assume that costs of the FECR is 50% of the price, providing a benefit of 30 USD per value-unit ($V_s = 30$).

In this first scenario, we define a fixed price of Ether at 307 USD, which is the year average price for Ethereum in 2020, as reported by [11]. The average gas price was 30 gwei, which becomes our as $\mu = 30x10^{-9}$.

Table 7.11 summarizes all the parameters considered to characterize our application and create a first scenario. We want to highlight that these parameter can be set at any other values to adapt to other situations or scenarios, based on the reader experience, or the particular case to study.

Once we have characterized our application, we can evaluate the scenario. From equation 7.4 we obtain that the bootstrap costs is 2300 USD, which includes deploying the marketplace and register the first 100 actors.

[11] https://etherscan.io/chart/etherprice

Table 7.11: Parameters for the cost model of the marketplace using in a public blockchain.

Parameter	Description	Value
A_0	Number of expected initial actors	100
$P(m)$	Average Month Price in 2020 (USD)	307
μ	Time factor for processing transactions	30 gwei
CC_C	Computational cost of creation transactions	3032280
CC_R	Computational cost of registration transactions	2467789
CC_I	Computational cost of interactions Transactions	327090
CC_V	Computational cost of value Transactions	1144229
F_g	Factor of growth	0.01
F_i	Factor of interaction	2
F_v	Factor of value transfer	0.5
F_k	Value unit factor	1
V_s	Total Monetary value of a value-unit	30

Using Equation 7.6 for the number of actors ($A(m)$), Equation 7.5 for the cost ($C(m)$), and Equation 7.8 for the benefits ($B(m)$) we evaluate the operation phase for 12 months. The results are displayed on Table 7.12. The evaluation shows that, on average, the monthly costs are lower than the benefits by approximately 26%. Thus, in this scenario, the application is economically viable.

The first scenario considered a fixed price for cryptocurrency, however, the volatility of the cryptocurrency price could greatly change results. Thus, a second scenario is evaluated using a variable function for $P(m)$ based on the average price for each month of 2020. Figure 7.7 shows the benefits per transactions ($\sum$B(m)/I(m)) and the costs per transaction ($\sum$C(m)/I(m)) compared with the first evaluation. Results show that the price volatility drastically changes the costs of the system. Furthermore, the system is not economically viable in the last two months of the year.

Table 7.12: Cost scenario for a decentralized marketplace on a public blockchain.

m	**A**	**I(m)**	**C(m)**	**B(m)**	**∑C(m)**	**∑B(m)**
		Txs	*USD*	*USD*	*USD*	*USD*
1	100	251	1152	1500	1152	1500
2	101	254	1163	1515	2315	3015
3	102	256	1175	1530	3490	4545
4	103	259	1187	1545	4677	6090
5	104	261	1199	1561	5876	7651
6	105	264	1211	1577	7087	9228
7	106	266	1223	1592	8310	10820
8	107	269	1235	1608	9545	12428
9	108	272	1247	1624	10792	14052
10	109	275	1260	1641	12052	15693
11	110	277	1272	1657	13324	17350
12	112	280	1285	1674	14609	19024
AVG	106	265	1217	1585		
SUM		3184	14609	19024		

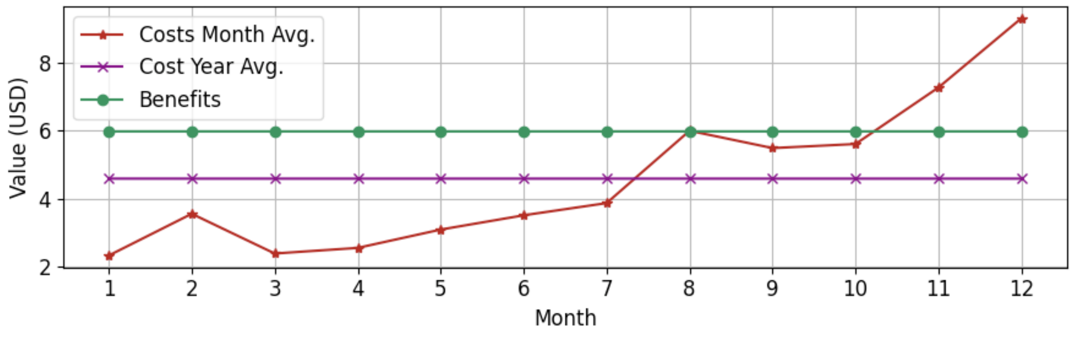


Figure 7.7: Cost and benefits per transaction in the marketplace reflecting the volatility of the cryptocurrency in 2020.

7.2.6 Conclusions

In this section, we presented a cost model for a blockchain-based application on permissionless networks. The model includes a transaction taxonomy and an application life-cycle to characterize blockchain-based applications. To illustrate the model usability, we quantitatively analyze the cost and benefits of an existing blockchain-based application. Results highlighted that the number of interactions is the main factor defining the cost of a blockchain application. Results also showed how the price volatility of the cryptocurrency radically changes the economical feasibility of the application. Thus, the proposed model is a useful tool for identifying the diverse factors characterizing blockchain-based applications and their costs in a public blockchain network infrastructure.

Future works include the evaluation of other applications to better define the parameters and provide reference values for the model. Another interesting research path is the development of a similar model for permissioned networks.

Chapter 8

Data-sharing and decentralised storage in blockchain-based applications

This chapter contains text taken from the published work:

- *M. Pincheira, E. Donini, R. Giaffreda & M. Vecchio (2020, March). A Blockchain-Based Approach To Enable Remote Sensing Trusted Data. In 2020 IEEE Latin American GRSS & ISPRS Remote Sensing Conference (LAGIRS) (pp. 652-657). IEEE.*

This chapter presents the second exploratory analysis on the challenges of blockchain-based applications. The main objective is to investigate the potentials and limitations of using a blockchain-based application for secure data sharing. Section 8.1 proposes an architecture and its analysis in the contexts of a remote sensing use case of precision farming. Section 8.2 extends the proposed architecture to include decentralized storage and provides an experimental evaluation of the proposed solution.

8.1 A blockchain-based approach to enable remote sensing trusted data.

8.1.1 Introduction

Recently, space agency policies on open data access encourage the development of various automatic methods to extract useful information for a wide range of Earth and Planetary science and applications. For instance, the European Space Agency provides Sentinel-2 data with a globe coverage and a high revisit frequency at no cost to the public though the Copernicus Programme. The spatial resolution (10 to 60 m) of these images is relatively high, but not enough for all applications. Hence, data fusion methods consider data from different short-range sources as complementary of remote sensing sensors.

The philosophy of open data becomes more relevant with the enormous quantity of data collected nowadays by close-range sensors, e.g., personal drones, IoT sensor networks, and open government data. Even if close and far range sensors acquire data on the same scene, they focus on different properties. These two types of data are complementary, and remote sensing (RS) can take advantage of close-range data and integrate them to generate more consistent, accurate, and useful products. For instance, close-range data as IoT measurements have a higher spatial and temporal resolution than ESA's Sentinel-2, but limited coverage. On the contrary, far-range data provide full coverage of the terrestrial surface with a relatively low spatial resolution. Moreover, close-range data available from in-situ (e.g., IoT measurements) are of particular importance for the validation of RS methods and products. However, owners of short-range data are part of a heterogeneous group, e.g., research institutions or startup owners, and are often unknown. Therefore, the owners and the data are not fully trustable, and thus, using the data may compromise the associated research outcome.

Currently, sharing and retrieving close-range data is possible through some tools mostly managed by intermediaries, e.g., open data portals. Intermediaries define the policies for data ownership and access and set the rules to evaluate data reliability and integrity. These tools tend to favor usability over transparency since they aim at connecting data owners with users. Therefore, a need emerges for a system that enables sharing and retrieving data without a limiting central authority.

This section proposes an architecture based on the blockchain to build a network to share and validate data acquired by untrusted sources. Blockchain acts as an intermediary connecting data owners and users– its is a decentralized database of trusted data accessible to all the participants. The intrinsic properties of blockchain enable trust and remove the necessity to have a third party validating the interactions among the participants. To investigate the proposed architecture, we analyze a use case to identify the critical issues and the potentialities of the system. We consider precision agriculture as it is well known for combining different types of data, e.g., IoT measurements and satellite optical images, to monitor the status of crop fields.

Related work

Recently, the interest in blockchain increased since users can monetize their resources and profit from them without an intermediary. Energy trading [151] and IoT data marketplaces [140] are just two examples where blockchains enable direct interactions between participants without a validating intermediary. Moreover, the auditability property allows the identification of the owner of each resource and the direct incentivizing of the participants contributions.

In remote sensing, only a few works explore the benefits of blockchain. In a whitepaper, ESA highlighted the importance of integrating blockchain in remote sensing applications [152]. On the one hand, blockchain enables the transfer of value– it is a public network where to share and retrieve data without a central authority. Moreover, smart contracts allow automating the actions, such as elaborating information. On the other hand, short-range sensors acquire everyday a large amount of data useful for Earth Observation. However, data owners lack a tool where to share data while keeping ownership. Further, the data cannot be trusted since data owners are unknown and thus, not reliable. Blockchain is the solution to build a system to generate trust among users without an intermediary.

Blockchain is a decentralized database that keeps track of the flow of information. Blockchain technology is verifiable and immutable by default– all the actors can access the information and its changes over time. For this reason, blockchain is suggested as a distributed database to share the knowledge on land ownership [84] [85], geohashing [84], i.e., to create a geodesic grid of the world [84], and to share geodata in an open way, e.g., public map, without relying on a central authority, e.g., Google Maps or OpenStreetMap.

In the literature, there exist few works exploiting blockchain to store and share data in a trustworthy and auditable manner. [153] propose a blockchain as a database to track satellites and debris orbits. The database has two configurations– history data configuration and sliding window configuration. In the first, all the information is available to all the network peers. In the second, only the information on the last 48 hours is available to all the network, while selected peers stores all history. Here, blockchain technology preserves data integrity and provides smart contract to automatize some actions on the data. However, the system does not exploit other advantages of the technology, e.g., trust and incentives, as all the actors are already known. In [154], a blockchain database stores the national data on water quality acquired by an irrigation system spread in the Taiwan area. The data benefit of the blockchain characteristics, i.e., decentralization, immutability, and auditability, however, trust is not a necessary since all the data sources are known. Moreover, the system stores only data from IoT sensors, which is far from the purpose of this research. Finally, [155] proposes a blockchain system to store and share geospatial data among users, particularly researchers. The system uses blockchain and smart contracts to develop a complex reward mechanism to engage the participants. However, the data are in global storage, which eliminates the decentralization and introduces a controlling authority that decide the rules to access the data.

The rest of this section is organized as follows– Section 8.1.2 proposes architecture with the details on the actors, the interactions, and the smart contracts. In Section 8.1.3, we analyze the proposed architecture with a use case on precision agriculture to identify its potentialities and issue. Finally, Section 8.1.4 concludes the case study and presents future works.

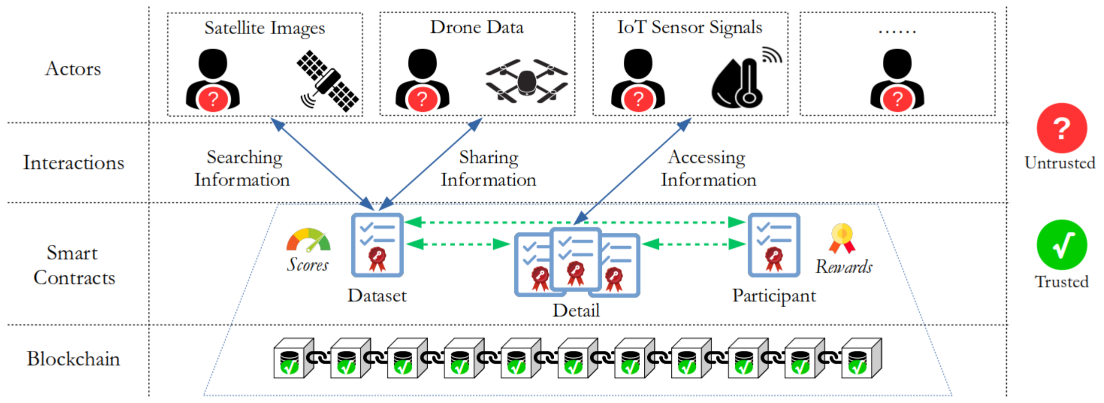


Figure 8.1: Proposed architecture – Untrusted actors interact using the smart contracts to share, search, retrieve, and score metadata and receive rewards.

8.1.2 Proposed architecture

This section aims at defining a distributed architecture that collects the information on the data, i.e., the metadata, shared by untrusted data sources. We use a blockchain-based approach to collect, validate, and track the metadata. Blockchain acts as a distributed infrastructure that provides a secure, immutable, and transparent record of the metadata. This Section will describe the architecture with a focus on the actors and the interactions. We first describe the actors, which are the blockchain peers that share and validate the metadata. Then, we analyze the possible interactions between the actors and the blockchain. Finally, we describe the smart contracts that enables the interaction between the actors.

Actors

Data comes from different sources that are trusted, e.g., space agencies and universities, or untrusted, e.g., private companies and volunteers. In this architecture, the peers are untrusted since we aim at providing value to the data that otherwise cannot be trusted. We consider as untrusted actors those that voluntarily collect and share data, as Figure 8.1 shows. Some examples of common actors are– the company sharing data acquired by IoT sensors on the temperature and the humidity of a crop field; the research group sharing a database on multitemporal images acquired by optical and SAR sensors; and finally, the people that shares pictures taken by a drone or a cellphone.

Interactions

Interactions happen between actors and the blockchain-based system. Here, actors can (i) share the metadata of a dataset via a transaction to a smart contract, (ii) use smart contracts to search in the dataset list, and finally (iii) use smart contracts to retrieve the dataset metadata. These interactions are associated with a reward for the actor and a quality evaluation of the dataset as an incentive to use the system. The score indicates the experience that the actor had with the dataset, including how representative the metadata are of the dataset, the quality of the data and of the dataset structure. However, the quality score should be define with criteria specific for the use case. The following paragraphs describe the interactions and the reward and evaluation mechanisms.

Sharing information. In this architecture, the blockchain is a container of information on different datasets. Blockchain maintains a list whose records are the metadata of the dataset shared by actors. The metadata is shared by sending transactions to the smart contract in a standard format. The smart contract verifies the transactions with predefined rules and rewards the actors. Moreover, the contract inserts the metadata in the dataset list– it adds a new record for a new dataset or updates the record for an existing one. Blockchain stores a history of all the events so that changes are traceable and accountable.

Searching for information. Actors looking for a specific dataset can query the smart contract for matching records in the dataset list. The smart contract searches the list with some criteria given by the actor. The contract retrieves the information on the matching datasets and provides it to the actor. Thus, the actor receives the part of the metadata and the current quality score of each the dataset. With this information, the actor can decide which dataset to use and require the complete metadata.

Accessing information. Considering the blockchain as a list of datasets, an actor can ask the smart contract to access the information related to a particular dataset. Retrieving the data implies that the smart contract transmits the complete metadata of that dataset to the actor. With this information, the actor can access and use the data. In exchange for the metadata, the actor is asked to give an evaluation of the quality of the dataset. If the actor provides this evaluation, it receives a positive reward, otherwise a negative one. Based on these rewards, smart contracts can limit the actor to have further access to other datasets.

Smart Contracts

Transactions are the interactions between actors and the blockchain system, i.e., the smart contracts. The smart contracts define rules and methods to validate and process the information in the transactions. They provide an interface to access information, making it available for actors and other smart contracts. Moreover, a relevant characteristic of smart contracts is the ability of creating and adding information in the blockchain. Here, these interactions translate into rewards for the actors and a quality score for the datasets. We employ three types of smart contracts: Participant, Dataset, and Detail.

Participant smart contract. The Participant smart contract manages the rewards of each actor according to predefined rules. The Participant contract receives transactions by the (i) Dataset contract when the latter adds a new record in the dataset list and (ii) Detail contract when an actor uses a dataset. The transaction contains the information on the actor and its rewarding. Hence, the contract manages the list of the actors and the history of their rewards. The list is accessible by actors and other smart contracts at any time, which is central for granting or denying access to datasets.

Detail smart contract. The Dataset contract creates a new Detail contract for each new dataset shared in the network to store metadata. Each Detail contract manages the metadata of a database and accepts the updates coming from the actor who originally shared the data. Each Detail contract is an independent unit of information more accessible than the blocks in the chain. Saving the information in the blockchain grants trust, while the contract provides a simpler way to access the metadata. Moreover, the Detail contract can provide the information in a meaningful way to the actor, e.g., only specific fields of the metadata.

The Detail smart contract implements the methods to (i) update and (ii) access the metadata. (i) Updating the metadata requires a transaction with the changes of the dataset. The contract processes only the transactions coming from the same actor who originally shared the dataset. Detail smart contract keeps track of all the changes, which are accessible by anyone in the system. (ii) Accessing dataset metadata requires a transaction to the Detail contract asking for that particular dataset. The contract first verifies if the actor is allowed to access the data in the Participant contract. Secondly, after sharing the metadata, the contract requests the actor to provide a quality score to the Datasets contract. Evaluating the dataset grants a positive reward and the further use of the system. On the contrary, the actor receives a negative reward when refusing to provide the quality score. Evaluating a dataset is of critical importance for the system– it helps in validating the information shared by the actors and thus checking the quality of the data in the blockchain.

Dataset smart contract. The Dataset smart contract keeps a list of the existing datasets and their quality scores. Moreover, it indicates how to (i) share, (ii) browse, and (iii) score the datasets in the systems. (i) To share an dataset, an actor sends a transaction with the dataset metadata coded in a standard format. First, the contract verifies the format of the metadata, then it adds a new record in the dataset list, and finally, it rewards the actor. When adding a new record, the contracts generate a Detail contract with a unique blockchain address stored in the dataset list. (ii) Browsing a dataset implies that an actor sends a transaction with search criteria. The contract uses the criteria to filter the dataset list and returns the results to the actor. The results contain the identifier of the datasets matching the criteria, part of the related metadata, and the current quality score. (iii) Scoring a dataset is done by the actors analyzing the datasets. Actors are asked to send a transaction indicating the quality evaluation of the data they analyzed.

Challenges

The success of participatory systems strongly depends on the number of participants– here the participants are the actors sharing and using the information in the blockchain network. The more are the actors involved, the more the blockchain will be successful and collect further information. Thus, a crucial element is the definition of an incentive mechanism to attract actors in the system. The main incentive is the possibility to browse in the list of datasets stored in the blockchain and then acquiring the information on accessing the dataset. Nevertheless, we decided to add a rewarding mechanism based on a gaming approach to engage more the actors. Each actor, accord-

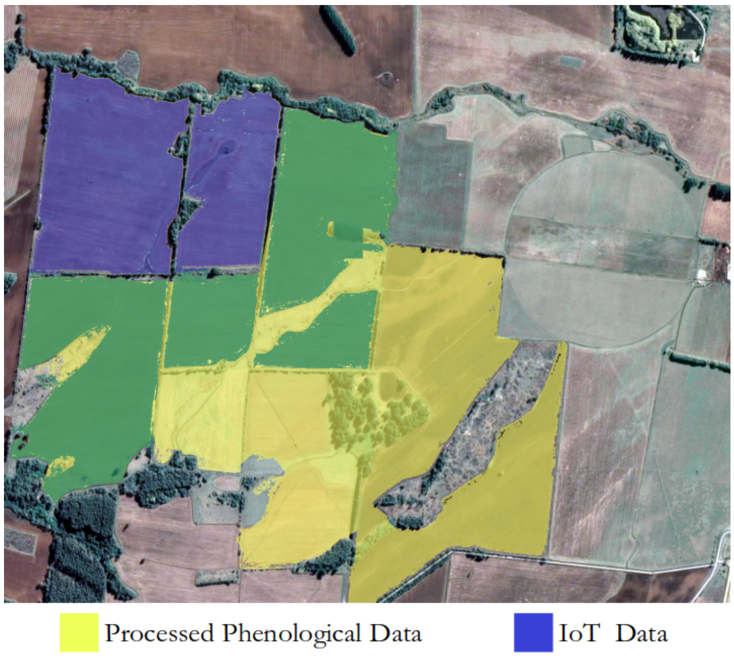


Figure 8.2: Geographic extension of the datasets shared in the blockchain system– in blue, the IoT raw and processed data, and in yellow, the processed Sentinel-2A data.

ing to its actions, receives a positive or negative reward that accumulates over time. Here, we propose a set of rules for the rewarding mechanism, but any other can be integrated since the system is modular. Thus, actors are encouraged to participate actively to increase their score and gain reliability.

Another critical issue is the definition of a standard format for the interactions between actors and smart contracts. The actors must send messages satisfying a shared and previously agreed protocol. The protocol indicates the message structure for all interactions, e.g., the fields required and their order. Having a structured information allows smart contracts to automatize the tasks required by the actor. Smart contracts automatically interact with the actors and elaborate the data in the blockchain. Thus, blockchains can be considered autonomous systems able to self-organize.

8.1.3 Use case: precision agriculture

This Section analyzes the proposed architecture by describing a use case where actors interact with the blockchain to share and retrieve information to investigate the potentialities and critical issues. We consider a use case in precision farming where the aim is monitoring the phenological cycle of crop fields by fusing different data. As an example, let us assume three actors interacting on the blockchain with different willings and needs. The following paragraphs describe the actors, their needs, and the issues that a blockchain-based system can help to overcome.

Actors

Actor A is a landowner that aims at developing an automatic system to manage crop fields, in green in Figure 8.2. He wants to estimate several parameters to monitor the evolution of crop fields and control actuators, such as those for watering and giving fertilizer. Actor A wants to retrieve these parameters, or in the worst-case estimate them, from a reliable list of datasets. Hence, actor A needs an easily accessible system, without a central control setting the rules to limit the actions of the users. Further, the system needs to guarantee the integrity of the data, i.e., the data can not be modified. As an extra feature, the system provides feedback as a quality score assigned by the other users according to their experience.

Actor B is a researcher developing methods for extracting parameters to monitor the phenological cycles of crop fields from Sentile-2A images, in yellow in Figure 8.2. B wants to share the research outcome while preserving the authorship of the data, and thus the credit. Hence, B needs a sharing system that i) tracks the origin and possible changes, ii) guarantees data integrity, i.e., data cannot be modified, and iii) assures access to everyone. As an additional feature, the system can provide feedback to the data owner based on the scores from the actors that used those data.

Actor C is an IoT startup that deploys sensors to monitor crop fields, in blue in Figure 8.2– the startup installed 20 sensors in 1 km^2 crop field, which acquire temperature and humidity measurements every 15 minutes. C owns raw data from the sensors and processed data, i.e., the temperature and humidity profile for the past years. C wants to have a record of the startup achievements and thus share the data in a transparent external system. Sharing the data is a way to present the outcome of the startup to investors and to attract new clients. However, actor C needs to preserve the ownership of the data, which is challenging with the existing centralized tools. The central authority sets the rules for sharing and retrieving data in the system. Hence, C may face several problems, e.g., with data ownership and data monetization.

The willingness and needs of the actors are diverse and hardly fulfilled by existing systems to share and retrieve data. Existing tools rely on a central authority that acts as an intermediary between untrused actors. The central control sets the rules of the system and thus should be blindly trusted by the actors. The rules indicate ownership and access policies– the former defines the possible change in ownership when uploading the data. The latter shows the accessing mechanism and requirements, which can be restrictive to some actors. Further, the intermediary is the single point of failure of the entire system– any interruption of the services directly compromises the data availability. Finally, actors must trust the central authority on the data integrity since the existing tools usually do not provide enough information to check the validity of the data, i.e., understand if the data are modified. In fact, the central authority has the complete control of the shared data and thus can modify the data or the quality evaluation.

A blockchain-based system overcomes these issues– the decentralized nature of the blockchain eliminates the controlling authority, and thus, the actors can directly interact with the network without the intermediary. Further, since blockchain is a distributed database, there is not a single point of failure, and the data is always accessible to any network peer. Moreover, the blockchain tracks all the interactions with the actors creating a history of the data origin and updates. This information is validated with cryptography techniques and agreed by all the network peers– this guarantees and preserves the data ownership and integrity. Finally, actors provide an evaluation of the data quality, removing any third-party interference as in the case of the existing tools.

A blockchain-based system is highly modular and enables the smooth integration of further services. For example, cryptocurrencies can easily be integrated to monetize the data and incentive participants. The blockchain system facilitates cryptocurrency payments with a marginal cost, which can be done automatically by smart contracts after each transaction. On the contrary, a centralized system does not enable such an effortless way to monetize the data.

Interactions

Sharing information. In our use case, B and C are the actors willing to share data– they send transactions to the blockchain with the information on the datasets. The blockchain validates the transactions with cryptography techniques and stores the data. Hence, blockchain-based systems remove the third-party intermediary that manages the data. Further, since the network peers have a copy of the blockchain, the system lacks a central authority overcoming the issues of existing tools to share and retrieve data.

Searching for information. Here, actor A wants to browse the list of shared datasets– A sends a transaction to the blockchain with filtering criteria. The blockchain system replies with the list of datasets matching the filters. Actor A can trust the integrity of the data since blockchain enables the immutability of the stored data. Moreover, the decentralized architecture of blockchain guarantees these results not to be accessible to everyone, without being censored or promoted.

Accessing information. Actor A is interested in the raw and processed data previously shared by B and C. Hence, A requests access to the datasets, by sending a transaction to the blockchain. The blockchain returns access information, which includes the integrity check and access credentials. The metadata accounts for the dataset updates and the change history of the dataset. Hence, the blockchain keeps track of the data owner and the update of the data, e.g., analyses and processing by other actors. Further, the blockchain provides the information to verify the validity of the dataset, i.e., if any modification occurred in the originally shared data.

8.1.4 Conclusions and future works

In this section we proposed a blockchain-based architecture that (i) enables data owners to share the data without an intermediary, (ii) keeps track of the data updates and provides a quality score, and (iii) overcomes the issue of the untrusted data owners. Any actor can share and retrieve data from the blockchain without caring for the reliability of the data source. Thus, there is no need for a third party that validates the data as the blockchain enables trust among the actors. We analyzed the critical issue and the potentialities of the proposed architecture with a remote sensing use case, i.e., precision farming. We conclude that such a system provides benefits to all the actors since sharing and retrieving data do not require the presence of an intermediary.

As future works, we plan to analyze other use cases with a higher number of shared datasets to examine the system scalability and robustness. Further, we plan to study an innovative storage system to decentralize not only dataset metadata but also the data.

8.2 Decentralized storage for trusted data sharing

8.2.1 Introduction

The architecture presented in Section 8.1 uses blockchain to maintain a record of all the interactions between actors on a data-sharing application. However, blockchain does not support the storage of a large quantity of data given different constraints imposed by each blockchain implementation (e.g., block size and transaction fees) [40, 46]. Therefore, other technologies, such as Decentralized File System (DFS), should be coupled with blockchain to store the data. In this section, we extend the architecture to include DFS to achieve a fully decentralized data-sharing application.

The rest of this section is structured as follows. Subsection 8.2.1 describes the working principles of DFS and its potential benefits to blockchain-based applications. Subsection 8.2.2 describes the proposed architecture. Subsection 8.2.3 reports the metrics for the evaluation of the system performances, the set-up of the experiments, and the performances of the proposed architecture. This section finalizes with conclusions on the proposed architecture based on its evaluation.

A primer of DFS

Distributed File Systems received a large amount of attention with the apparition of peer-to-peer sharing applications such as Napster, Gnutella, and Kazaa in the early 2000s [156]. Lately, DFS has re-gained attention from the blockchain space as an additional component to create decentralized applications.

A DFS provides a complementary layer of decentralized storage to a blockchain, while a blockchain can provide an additional layer of security and incentives for the peers to cooperate in storing and distributing data. [157] . Therefore, protocols such as Storj [158], Sia[159], and Arweave[160]

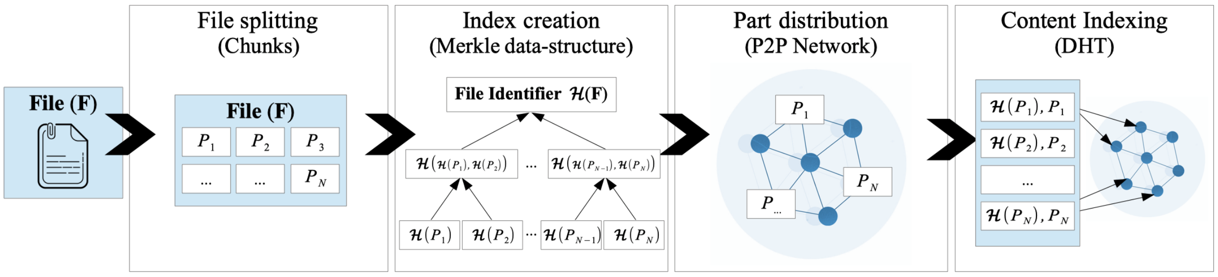

File (F)
File splitting (Chunks)
File (F)
P_1 P_2 P_3 ... ... P_N
Index creation (Merkle data-structure)
File Identifier $\mathcal{H}$(F)
$\mathcal{H}(\mathcal{H}(P_1),\mathcal{H}(P_2))$ ... $\mathcal{H}(\mathcal{H}(P_{N-1}),\mathcal{H}(P_N))$
$\mathcal{H}(P_1)$ $\mathcal{H}(P_2)$ ... $\mathcal{H}(P_{N-1})$ $\mathcal{H}(P_N)$
Part distribution (P2P Network)
P_1 P_N $P_{...}$
Content Indexing (DHT)
$\mathcal{H}(P_1), P_1$
$\mathcal{H}(P_2), P_2$
...
$\mathcal{H}(P_N), P_N$

Figure 8.3: Simplification of decentralized storage protocol (based on IPFS and Swarm).

aim to achieve this blockchain and DFS integration. According to authors on [161], two of the most representatives protocols are the InterPlanetary File System (IPFS) and Swarm. IPFS [162] is the most mature in terms of development and adoption, while Swarm is a DFS developed and integrated with the Ethereum protocol [116] profiting also from the smart-contracts of the blockchain. Both IPFS and Swarm aim to provide a multi-purpose decentralized distributed storage solution with a content delivery protocol, but with differences in both design and implementation (e.g., the network layer, the peer management protocol, the data structure used).

DFS Working principles

DFS implementations typically have different protocols. Nonetheless, IPFS and Swarm share several similarities in both design and implementation, see Figure 8.3. When uploading a file **F**, the DFS divides it into N smaller pieces P_n so that $\mathbf{F} = \{P_1, \dots, P_N\}$. Each P_n is processed by a cryptographic hashing function $\mathcal{H}(\cdot)$ that generates a unique cryptographic hash. For each P_n, the hash $\mathcal{H}(P_n)$ acts as unique identifier. A Merklee data structure (Merkle Tree for Swarm and Merklee DAG for IPFS, respectively) connects all the hashes of all the pieces at different levels. At the bottom level of the tree, the hashes $\mathcal{H}(P_n)$ and $\mathcal{H}(P_{n+1})$ of pieces P_n and P_{n+1} are connected to generate $\mathcal{H}(\mathcal{H}(P_n), \mathcal{H}(P_{n+1}))$. Finally, at the root of the tree, a unique hash for each file $\mathcal{H}(\mathbf{F})$ is generated (see Figure 8.3). The use of Merklee data-structures [163] creates a unique identifier for the file based on its content. Merklee data-structures provide beneficial properties for content addressing, optimizing disk usage, and file integrity. Each piece of file P_n is stored on different network peers along with the related hash $\mathcal{H}(P_{n+1})$. Using the hashes in a Distributed Hash Table provides an efficient routing mechanism to address uploaded files and their pieces among the network peers. If any P_n is corrupted or tampered with, the hash changes for the entire file, enabling quick integrity verification.

DFS Benefits

DFS overcomes several challenges of centralized cloud storage, such as data reliability, availability, and integrity. The underlying peer-to-peer network enables an efficient auto-scaling system without a single point of failure, creating a highly reliable storage infrastructure [161]. Since a DFS simultaneously stores the files in several locations, the content is censorship-resistant with higher availability despite individual failures of particular nodes [161]. Regarding security and privacy, the use of cryptographic data-structures provides embedded tamper-proof of the content, creating an additional layer of integrity verification.

8.2.2 Proposed Architecture

This section proposes a fully decentralized architecture to share and retrieve data in a trusted and traceable way without an intermediary. The architecture integrates a permissionless blockchain (BC) and a decentralized file system (DFS). BC provides an immutable and transparent record to all the users (actors or network peers) and thus, guarantees traceability and trust. The decentralized file system network guarantees data availability and integrity as data are encrypted and always available. Hence, the proposed architecture guarantees the integrity, ownership, and availability of the data on the network to all the actors in a trusted and decentralized way.

We assume a group of $\mathcal{X} = \{X_i, i \in [1, \ldots, N_X]\}$ actors that are willing to share and retrieve data. Since the actors are unknown to each other, they are untrustable. Therefore, the data they want to share and retrieve cannot be trusted, which means that the integrity, availability, and accessibility of the data cannot be guaranteed. The actors sharing the data are called data-owners and are the owners of the intellectual property of the group of datasets $\mathcal{Y} = \{Y_j, j \in [1, \ldots, N_Y]\}$. We assume that data-owners are willing to share the data, but in a secure and traceable way to preserve the ownership and the intellectual rights. In the context of scientific research, they also want recognition and acknowledgment for the data usage. The actors willing to retrieve data are called data-users, and they need a way to verify the dataset origin, integrity, and changes, i.e., evolution over time. Normally, this task is done by an intermediary that connects and mediates between two users – here, the BC acts as an intermediary, enabling trusted and direct interactions between unknown actors. Note that an actor can act both as data-owner and data-user within different interactions in the proposed architecture. Here, we assume that each datasets $Y_j, j \in [1, \ldots, N_Y]$ consists of the metadata $M_j, j \in [1, \ldots, N_Y]$ and the data $D_j, j \in [1, \ldots, N_Y]$. The metadata M_j collect all the information describing the data, e.g., the data owner, the time of acquisition, and the data source. The numbers and types of fields of the metadata depend on the data type. D_j is of the data to share that can have a different structure, e.g., a 1D signal measured by IoT sensors, a 2D matrix, such as a satellite image.

The proposed architecture consists of an interface that connects two networks (see Figure 8.4), i.e., a public blockchain (BC) and a distributed file system (DFS). The BC network stores the metadata and keeps track of all the actor interactions with each dataset. For each dataset, the BC records

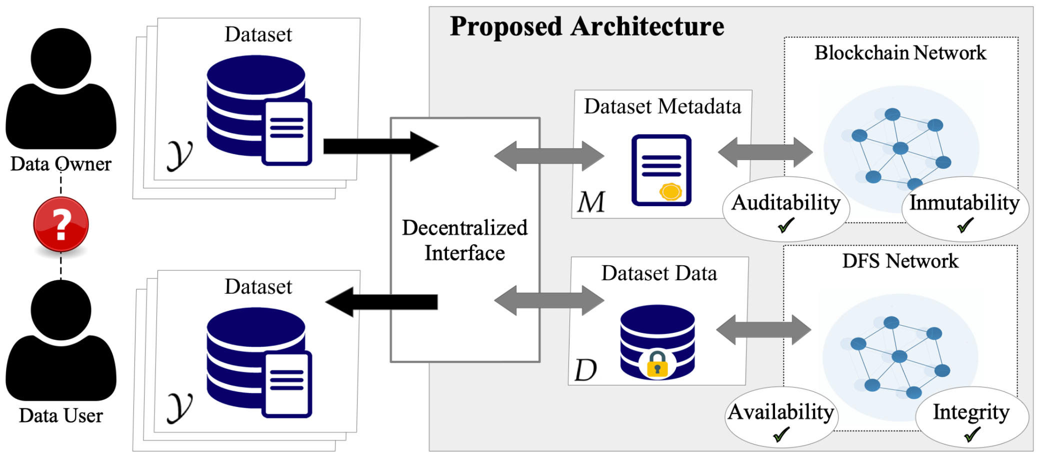


Figure 8.4: Proposed architecture based on blockchain, smart contracts, and DFS to provide an infrastructure for sharing datasets among untrusted actors.

the ownership and the evolution (i.e., updates and changes), the identification (ID) of actors downloading it, and the evaluation. Evaluations are given by actors that previously downloaded the dataset. They can provide a score of the dataset goodness, such as the structure and quality. To increase the system openness, we propose to integrate a permissionless blockchain that is open to any untrusted actor. The advantages of using a public blockchain in terms of transparency, availability, and openness are greater than the disadvantages, which include costs, latency, and transaction throughput. A public blockchain enables any data-owner to use the infrastructure as a trustless platform for directly interacting with unknown data-users. As blockchain storage capacity is limited [11], data are stored in a DFS network to avoid any centralization. DFS increases the system liveness as data is stored across multiple network peers. DFS provides each dataset with a unique identifier based on the data content. The ID is securely linked to the data-owner to reduce possible data mismanagement. Moreover, DFS has faster download and upload times than a centralized storing platform because of the multiples nodes storing the data. Finally, the decentralized interface (DI) acts as a coordinator, providing an interface to the blockchain and the decentralized storage networks. The decentralized interface interacts with the smart contracts and performs off-chain tasks, such as encryption and decryption of datasets. Each actor stores a copy of DI to enhance the system decentralization.

8.2.3 Experimental setup

This Section evaluates the performance, potentialities, and criticalities of the proposed architecture in terms of processing time, cost, resources for both the blockchain and distributed network. To this end, we implemented a fully functional Proof of Concept (PoC). We consider Ethereum as a public implementation of blockchain. We evaluated the performance of two different DFS protocols, i.e., IPFS and Swarm. The architecture setup and parameter range, e.g., data and file size, are defined by identifying a realistic use case, i.e., precision agriculture, that highly benefits from data-sharing among untrusted actors as described on Section 8.1.

Hardware and Software setup

As software tools for the Ethereum blockchain, we used the official Geth client (version 1.9.18-stable) to run two independent blockchain nodes: one node for the Ropsten network, working with PoW consensus, and the other node for the Goerli network, working with PoS. We used two identical virtual machines on an OpenStack server with 4 GB of Ram, 20 GB of SSD, and 4 vCPU, and running clean Linux Ubuntu installation (version 18.04). For the decentralized storage module, we choose the IPFS [1] and Swarm [2], two of the most representatives DFS implementations [161]. IPFS [162] is the most mature in terms of development and adoption, while Swarm is a DFS developed and integrated with the Ethereum protocol [37] profiting also

[1]https://ipfs.io/

[2]https://ethersphere.github.io

from the smart-contracts of the blockchain. For a complete comparison of these two DFS, challenges, and limitations, the reader is referred to [161]. As software tools for the decentralized storage module, we ran two independent nodes, using the official IPFS client (version 0.5.1) and official Swarm client (version 0.5.7-5ccfd995). We used two identical virtual machines on an OpenStack server, with 4 GB of Ram, 20 GB of SSD disk, and 4 vCPU, running a clean Linux Ubuntu installation (version 18.04).

The client module was implemented using Python (version 3.6) and ran on a Lenovo T490s notebook, with 16 GB of Ram 256 SSD disk, and an Intel i-7 processor at 1.90 GHz over a clean install of Linux Ubuntu (version 18.01).

Data and Metadata

Metadata are managed through transactions and stored in the blockchain. We defined 8 transactions to implement the functionalities of the PoC, i.e., Create, Update, Authorize, Reject, Search, Request, Details, and Score. Additionally, we consider a Bootstrap transaction to deploy the system on the blockchain network. To create a new dataset, we considered seven fields based on the ISO 19115-2 standard (a metadata profile for precision agriculture: [3]): identifier, title, abstract, keywords, spatial extension, time extension, and contact information). To update and search for a dataset, we considered only four fields. The transactions for request, details, accept, and reject access are only considered a message text field. Finally, the score dataset transaction is based on an ERC20 Token transfer transaction. The ERC20 is an Ethereum standard ensuring exchangeability and interoperability for all the blockchain tokens. The same token standard is used for rewarding the users.

For the data stored on the decentralized file system, we considered seven sizes of files: 01, 05, 10, 50, 100, 250, and 500 MB based on the file sizes

[3]https://www.iso.org/standard/67039.html

in literature [164]. Each file is a consolidated Unix archive (tar), including a single random bytes file, and a text file with a unique timestamp and description for each experiment. While an exhaustive list of all possible file sizes is beyond the scope of this paper, the selected sample provides a general overview suitable for different use cases.

Metrics

To evaluate the blockchain performance, we consider transaction cost and transaction processing time as main metrics from a user perspective working with the metadata. For each transaction, we run a series of experiments on each network (i.e., Ropsten and Goerly) at different times during the day. In each experiment, a transaction with random metadata was created and sent to the respective network node. We used a python script that created the transaction, sent it to the network, and monitored the time until the transaction was processed.

The proposed architecture uses a decentralized file system (DFS) to store the data. Regarding decentralized storage, we evaluate the upload and download times of datasets. For both experiments, we also provided an evaluation in terms of network traffic and CPU usage.

8.2.4 Evaluation results

Blockchain Transaction Costs

To evaluate the blockchain performance, we consider transaction cost and transaction processing time as main metrics from a user perspective working with the metadata. For each transaction, we run a series of experiments on each network (i.e., Ropsten and Goerly) at different times during the day. In each experiment, a transaction with random metadata was created and sent to the respective network node. We used a python script that created

Table 8.1: Transaction size and cost.

Transaction		Metadata (byte)	Tx Size (byte)	Gas (units)	Cost (USD)
0)	Bootstrap	10	18971	3946432	11.83
1)	Create	509	817	2141905	6.42
2)	Update	236	529	77068	0.23
3)	Authorize	50	239	93864	0.28
4)	Reject	50	238	31991	0.09
5)	Search	-	-	0	0
6)	Request	14	206	46665	0.13
7)	Details	-	-	0	0
8)	Score	10	239	73439	0.22

the transaction, sent it to the network, and monitored the time until the transaction was processed.

The proposed architecture uses a decentralized file system (DFS) to store the data. Regarding decentralized storage, we evaluate the upload and download times of datasets. For both experiments, we also provided an evaluation in terms of network traffic and CPU usage.

Table 8.1 shows the average size (in bytes) of the metadata and the resulting transaction, gas cost, and monetary cost (in USD) for each of the 7 transactions. The 'Bootstrap' and 'Create' transaction is the most expensive in terms of gas and cost as the cost in USD is equal to 11.83 and 3.42, respectively. In terms of blockchain memories, 'Create' and 'Update' transactions are the most expensive transactions, occupying 509 and 236 bytes, respectively. 'Search' and 'Detail' transactions have no gas cost as they require the querying of the local version of the blockchain.

Table 8.2: Transaction processing times in terms of minimum, maximum, average, and variation on a Ropsten and a Goerli network.

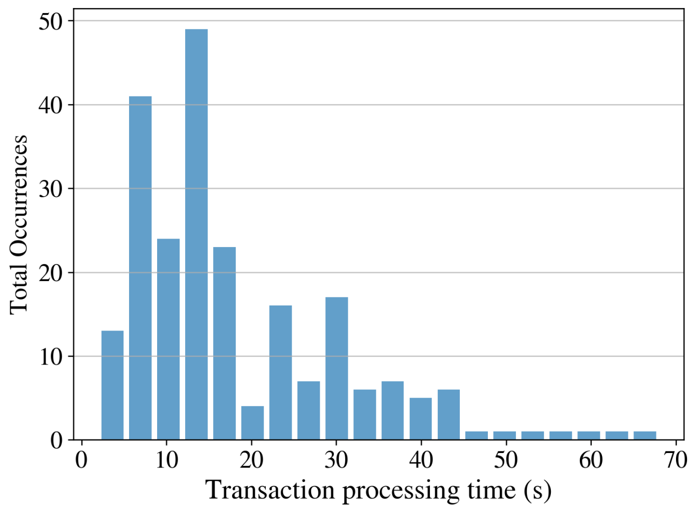
Transaction		Ropsten (PoW)				Goerli (PoS)			
		Min	Max	Avg	σ	Min	Max	Avg	σ
0)	Bootstrap	06	40	19.14	10.74	08	24	17.00	5.92
1)	Create	04	68	19.03	15.27	02	32	15.30	5.38
2)	Update	04	48	17.50	10.81	12	30	17.20	5.05
3)	Authorize	06	50	24.61	11.14	12	30	17.13	7.11
4)	Reject	08	60	22.83	14.26	10	30	17.40	7.08
6)	Request	06	42	17.69	10.19	02	32	16.57	5.23
8)	Score	04	40	13.56	8.82	14	32	17.53	5.72

Blockchain Transactions processing times

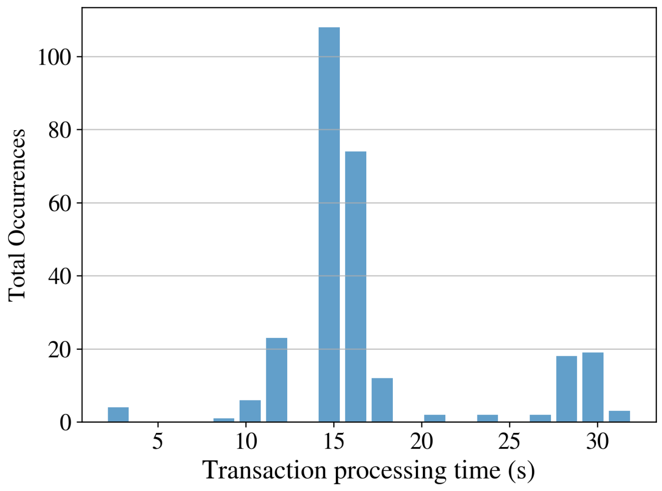
To measure the processing time, we created 36 instances of each transaction type (see Section 8.2.4). We send all the transactions approximately 15 apart from each other to have different network conditions. Table 8.2 shows the minimum, maximum, mean, and standard deviation of the processing time in seconds for ten instances using a gas price of 10 gwei for both PoW (Ropsten) and PoS (Goerli) networks. Figure 8.5a and 8.5b show the histogram of the processing times for all types of transactions on both PoW and PoS networks, respectively. In a Ropsten network (PoW) with $\mu = 18.35$ and $\sigma = 12.39$, the transactions were more frequently executed with less than 20 s but could take up to 68 s. In a Goerli network (PoS) with $\mu = 16.69$ and $\sigma = 5.8$, the transaction processing time was never larger than 32 s and more frequently around 15-18 s.

DFS Upload Times

We created and uploaded ten different files of each size. To evaluate different network conditions, we uploaded one file approximately every 30 minutes. To



(a) Ropsten network (PoW) transactions processing time histograms for $\mu = 18.35$ and $\sigma = 12.39$ for 36 transactions of each type.



(b) Goerli network (PoS) transactions processing time histograms with $\mu = 16.69$ and $\sigma = 5.8$ for 36 transactions of each type.

Figure 8.5: Transactions processing times histograms

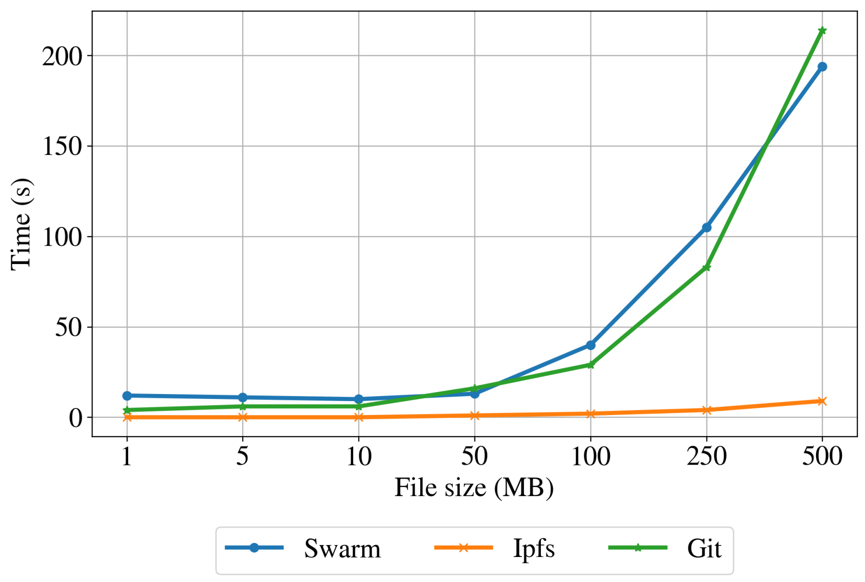
Table 8.3: Upload times expressed in seconds.

File Size	Swarm				Ipfs				Git			
-	Min	Max	Avg	σ	Min	Max	Avg	σ	Min	Max	Avg	σ
001 MB	1	72	12	16.11	<1	1	<1	0.47	4	6	4	0.75
005 MB	1	45	11	9.89	<1	1	<1	0.44	5	8	6	1.02
010 MB	1	40	10	8.42	<1	1	<1	0.49	6	7	6	0.49
050 MB	9	67	13	14.26	1	2	1	0.25	16	17	16	0.4
100 MB	7	98	40	30.49	1	5	2	0.91	29	30	29	0.49
250 MB	22	522	105	120.03	8	18	9	2.36	82	85	83	1.36
500 MB	58	741	194	155.79	8	18	9	2.36	210	218	214	3.12
	Avg. peers: 29 σ = **3.35**				**Avg. peers: 164** σ = **63.54**				**Ping time: 23 ms** σ = **0.55**			

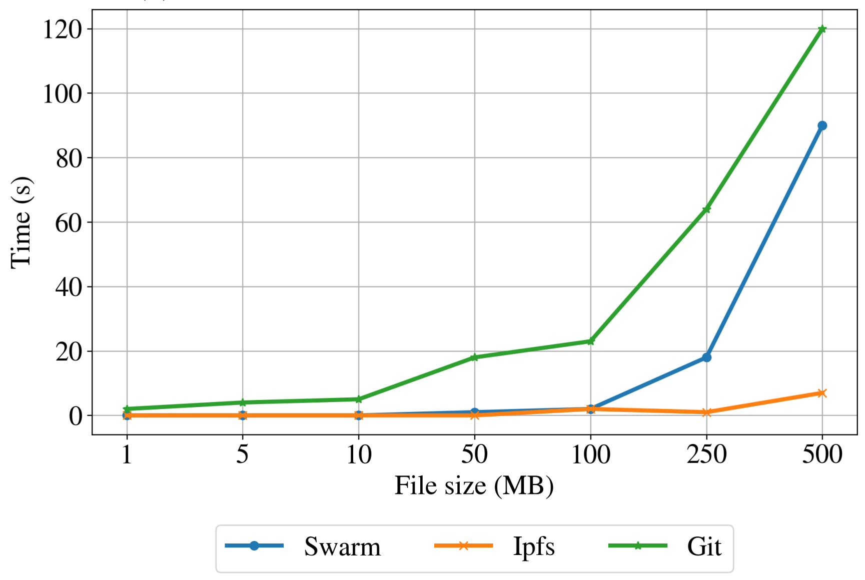
get a reference time, we also uploaded the files to a Git repository, pushing a new commit, and we used a shell script to measure the elapsed time. Table 8.3 shows the minimum, maximum, average, and standard deviation of the uploading time for Swarm, IPFS, and Git. The table includes the average peers connected to each node and the average ping time to the Git repository. The IPFS protocol requires on average a lower uploading time with a lower variance (see also Figure 8.6a) than Swarm and Git. This is because the IPSF normally split the data into fewer pieces and send them to a closer node. Swarm protocol privileges more split and farther nodes. finally, Git is a centralized system, and thus, the uploading time depends on the workload of the central node.

DFS Download Times

To assess the performance of the proposed architecture in retrieving the data, we downloaded each file approximately 30 minutes after the upload. As reference time, we download the test file from the Git repository. We made a pull from the command line, using a pair of register keys as the authentication



(a) Average upload times for different file sizes.



(b) Average download times for different file sizes.

Figure 8.6: Comparison of upload and download times for different file size.

method. Table 8.4 shows the minimum, maximum, average, and standard deviation of the download time for Swarm, IPFS, and Git. The table also includes the average peers connected to each node and the average ping time of the Git repository. For all the file sizes, the download time is lower for the Swarm and IPFS protocol than Git (see also Figure 8.6b). The download time for files up to 100 MB is almost constant for Swarm and IPFS protocols. For larger files than 100 MB, the performance degrades slightly for the IPFS and highly for the Swarm protocol.

Table 8.4: Download times expressed in seconds.

File Size	**Swarm**				**Ipfs**				**Git**			
-	Min	Max	Avg	σ	Min	Max	Avg	σ	Min	Max	Avg	σ
001 MB	<1	1.0	<1	0.22	<1	1.0	<1	0.24	2	3	2	0.49
005 MB	<1	1.0	<1	0.22	<1	1.0	<1	0.22	3	20	5	5.4
010 MB	<1	1.0	<1	0.22	<1	1.0	<1	0.25	2	9	5	2.42
050 MB	<1	2.0	1	0.46	<1	1.0	<1	0.25	12	43	18	12.12
100 MB	1	4	2	0.7	<1	28.0	2	6.77	5	52	23	17.55
250 MB	4	205	18	49.9	<1	2.0	1	0.32	53	87	64	13.63
500 MB	8	1218	90	301.38	1	86	7	21.03	104	156	120	18.29
	Avg. peers: 44 σ **= 11.32**				**Avg. peers: 933** σ **= 79.99**				**Ping time: 42 ms** σ **= 25.93**			

8.2.5 Conclusions

In this section, we extend the architecture described on Section 8.1 to include DFS as an additional decentralized component for blockchain-based application. Blockchain stores the metadata, and the DFS stores the encrypted data split among several peers.

To quantitatively evaluate the system performance, we implemented a fully functional PoC, considering Ethereum as a public blockchain and two DFS implementations, namely IPFS and Swarm. The experiments showed that a permissionless blockchain is viable in terms of monetary costs and performance for data sharing. The system costs 12 USD to deploy, and the majority of the transactions cost less than one USD. Regarding the time response, the average transaction processing time is less than 20 s, and it can go maximum up to 68 s in a Ropsten network and 37 in a Goerli network, respectively. The experiments showed that both implementations of decentralized storage achieve good results in terms of data, cost, and impact on the node resource. Under the same conditions, the DFS implementations (Swarm and IPFS) require less time than a centralized storing system (Git) for both data upload and download. IPFS provides a faster upload and download than Swarm at the cost of splitting of the data in a fewer number of pieces that are sent to close nodes. Swarm split the file into more pieces that are stored into several far and close nodes. Although this strategy increases the time response of the system, it provides a higher level of data availability.

Future works include evaluating and comparing additional DFS implementations and centralized platforms. Further evaluations also involve the time response of read-only transactions and the impact of different encryption methods on the shared data from both security and usability perspectives.

Chapter 9

Conclusions

In recent years, there has been a growing interest in integrating blockchain technologies into IoT systems to create a trustless architecture without the need for an intermediary. However, works in the literature have failed to consider a direct role for the sensing device in these blockchain-based IoT systems. These devices are the core of the IoT and typically the most constrained hardware elements of the technology stack. Furthermore, state-of-the-art works focused on permissioned networks that do not provide the same level of openness, decentralization, and neutrality as a permissionless network.

The thesis presents an architecture that integrates blockchain technology into the Internet of Things, enabling the development of next-generation IoT applications. The architecture relies on blockchain inherent features to mitigate some of the current challenges in centralized IoT systems. The proposed architecture considers constrained sensing devices as direct actors on a public blockchain network, i.e., the devices interact with the blockchain without an additional system component. This approach guarantees a *root of trust* for the sensed data that is later maintained and enforced by the permissionless blockchain network. In the proposed architecture, smart contracts provide a platform to define complex business logic autonomously enforcing agreements between untrusted actors based on trusted values coming from ubiquitous IoT devices.

The research followed an iterative DSR approach; designing, building, and evaluating new IT artifacts using two case studies in the agricultural IoT domain. The iterative process generated new knowledge about the strengths and challenges of blockchain-based IoT applications. These challenges fostered two exploratory studies that diverged from the main IoT domain; however, they also provide novel contributions to blockchain-based applications.

9.1 Novel contributions

The novel architecture tackles three problems of current blockchain-based IoT systems i) constrained sensing devices as direct actors on a blockchain system, ii) permissionless blockchain networks, and 1ii) Smart contracts as an IoT application platform.

Chapter 5 addressed the exponential rise in adoption of IoT devices applications in Agri-Food supply chains. The chapter described an architecture for a fully decentralized, blockchain-based traceability IoT application for Agri-Food supply chain management. The proposed solution is developed, deployed, and evaluated for a farm-to-fork use-case using two different blockchain implementations. The evaluation unveiled that a blockchain-based IoT system provides several advantages over a centralized system. Our results showed that a private implementation had better performance (i.e., latency, network traffic, and CPU load). Private blockchains also provide several programming languages for developing applications and allow customization of the blockchain records, allowing the faster development of more sophisticated applications. However, private blockchains are still far from being considered a mature implementation in terms of security and interoperability when compared with a public blockchain.

Chapter 6 explored how the energy-efficient integration of blockchain and IoT can incentivize virtuous behaviors in agricultural practices. The chapter described an architecture that includes constrained sensing IoT devices, smart contracts, and a public blockchain network for a water management system. By implementing a complete proof-of-concept, we evaluated the impact of the architecture in terms of memory, program size, communications, and power consumption. The results showed that off-the-shelf, cost-effective IoT devices are can interact directly with a public blockchain even over LP-WAN. Regarding energy budget, the blockchain operations required only an additional 6% of the energy concerning traditional IoT operations (i.e., without blockchain involvement) and account for only 0.004% of the total daily energy budget. Moreover, the use of a permissionless blockchain allows the seamless inclusion of several water management stakeholders, increasing the value of the system and the scope of the proposed architecture.

The exploratory analyses examine two challenges of blockchain-based applications i) user experience and monetary costs and ii) data sharing and decentralized storage.

Chapter 7 presented a blockchain-based application where smart contracts alone provide all the functionalities needed by a marketplace of Fog/Edge computing resources. We developed a full version of the proposed architecture and evaluated several design decisions in terms of cost and user experience. Compared to the state-of-the-art approaches, our fully-decentralized architecture provides response times within the same range as those with partially-decentralized architectures. Complementing this case study, Section 7.2 presented an economic cost model that includes a transaction taxonomy and an application life-cycle. The evaluation showed that the proposed model is a step towards a better characterization of blockchain-based applications and a better comprehension of their monetary costs.

Chapter 8 proposed a decentralized blockchain-based architecture for trusted open data sharing. We analyzed the proposed architecture with a remote sensing use case, i.e., data fusion for precision farming. The analysis showed that the proposed architecture fulfills the requirements of the use case and removes the intermediary. Section 8.2 extends the architecture to include DFS to compensate for the storage limitations of existing permissionless blockchain networks. We implemented a PoC to quantitatively evaluate the proposal using the two more mature DFS implementations, i.e., IPFS and Swarm. Our results showed both decentralized systems provide similar upload and download times of a centralized alternative (i.e., data storage) with a series of extended advantages such as availability and resiliency.

9.2 Future works

Future works for the proposed framework include evaluating metrics related to the smart contracts and a formal security analysis. Exploring the use of layer-2 architectures on public blockchains (e.g., lightning network, state channels, plasma network) is another intriguing research path for the proposed framework. Further, evaluating other constrained communications technologies (e.g., NB IoT) could provide interesting information for additional IoT use cases.

From the first exploratory analysis, new research paths include developing more complex smart contracts to improve the proposed cost model for the economic evaluation of blockchain-based IoT applications. Future works from the second exploratory analysis include integrating decentralized file systems on IoT architectures towards the next generation of blockchain-based decentralized IoT applications.

Bibliography

[1] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International Workshop on Open Problems in Network Security*, pp. 112–125, Springer, 2015.

[2] W. Viriyasitavat, L. D. Xu, Z. Bi, and D. Hoonsopon, "Blockchain technology for applications in internet of things—mapping from system design perspective," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8155–8168, 2019.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[4] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[7] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[8] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[9] A. R. Biswas and R. Giaffreda, "Iot and cloud convergence: Opportunities and challenges," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pp. 375–376, IEEE, 2014.

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[11] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2018.

[12] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[13] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future Generation Computer Systems*, 2018.

[14] C. Lazaroiu and M. Roscia, "Smart district through iot and blockchain," in *Proc. of the IEEE 6th International Conference on Renewable Energy Research and Applications*, pp. 454–461, 2017.

[15] F. Tschorsch and B. Scheuermann, “Bitcoin and beyond: A technical survey on decentralized digital currencies,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[16] D. Minoli, K. Sohraby, and B. Occhiogrosso, “Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems,” *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 269–283, 2017.

[17] K. Georgiou, S. Xavier-de-Souza, and K. Eder, “The iot energy challenge: A software perspective,” *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 53–56, 2018.

[18] J. Huang, L. Kong, G. Chen, M. Wu, X. Liu, and P. Zeng, “Towards secure industrial iot: Blockchain system with credit-based consensus mechanism,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, 2019.

[19] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, “Edge intelligence and blockchain empowered 5G beyond for the industrial internet of things,” *IEEE Network*, vol. 33, no. 5, pp. 12–19, 2019.

[20] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, “Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges,” *Mechanical Systems and Signal Processing*, vol. 135, p. 106382, 2020.

[21] O. Khutsoane, B. Isong, and A. M. Abu-Mahfouz, “Iot devices and applications based on lora/lorawan,” in *Proc. of the 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6107–6112, 2017.

[22] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, “Energy consumption model for sensor nodes based on lora and lorawan,” *Sensors*, vol. 18, no. 7, p. 2104, 2018.

[23] A. Bahga and V. K. Madisetti, “Blockchain platform for industrial internet of things,” *Journal of Software Engineering and Applications*, vol. 9, no. 10, p. 533, 2016.

[24] S. Cha, J. Chen, C. Su, and K. Yeh, “A blockchain connected gateway for ble-based devices in the internet of things,” *IEEE Access*, vol. 6, pp. 24639–24649, 2018.

[25] O. Novo, “Blockchain meets iot: An architecture for scalable access management in iot,” *IEEE Internet Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[26] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, “Consortium blockchain for secure energy trading in industrial internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3690–3700, 2018.

[27] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, “Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains,” *IEEE Transactions on Industrial Informatics*, vol. 13, pp. 3154–3164, 12 2017.

[28] M. Conoscenti, A. Vetrò, and J. C. De Martin, “Blockchain for the internet of things: A systematic literature review,” in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–6, 2016.

[29] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work

blockchains," in *2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, p. 3–16, ACM, 2016.

[30] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. M. Prabhu, "Building puf based authentication and key exchange protocol for iot without explicit crps in verifier database," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 424–437, 2019.

[31] M. Dammak, S. M. Senouci, M. A. Messous, M. H. Elhdhili, and C. Gransart, "Decentralized lightweight group key management for dynamic access control in iot environments," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1742–1757, 2020.

[32] J. Heiss, J. Eberhardt, and S. Tai, "From oracles to trustworthy data on-chaining systems," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 496–503, 2019.

[33] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 182–191, 2016.

[34] A. Hevner and S. Chatterjee, "Design science research in information systems," in *Design research in information systems*, pp. 9–22, Springer, 2010.

[35] R. Beck, J. Stenum Czepluch, N. Lollike, and S. Malone, "Blockchain– the gateway to trust-free cryptographic transactions," *Proceedings of the Twenty-Fourth European Conference on Information Systems (ECIS)*, 2016.

[36] S. Chatterjee, J. Byun, K. Dutta, R. U. Pedersen, A. Pottathil, and H. Xie, "Designing an internet-of-things (iot) and sensor-based in-home monitoring system for assisting diabetes patients: iterative learning from two case studies," *European Journal of Information Systems*, vol. 27, no. 6, pp. 670–685, 2018.

[37] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Crypto-Currencies.* O'Reilly Media, Inc., 1st ed., 2014.

[38] M. Pilkington, "Blockchain technology: principles and applications," *Research Handbook on Digital Transformations*, 2015.

[39] R. Beck, J. S. Czepluch, N. Lollike, and S. Malone, "Blockchain-the gateway to trust-free cryptographic transactions.," in *24^{th} European Conference on Information Systems (ECIS)*, (Istanbul, Turkey), 2016.

[40] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin white paper - Available at https://bitcoin.org/bitcoin.pdf*, 2008.

[41] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology – CRYPTO'87*, pp. 369–378, 2000.

[42] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Technical Report - Available at https://gavwood.com/paper.pdf*, pp. 1–32, 2014.

[43] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[44] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, 2014.

[45] J. Kwon, "Tendermint: Consensus without mining," *Technical Report - Available at https://tendermint.com/static/docs/tendermint.pdf*, 2014.

[46] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," *Technical Report - Available at http://ethereum.org/ethereum.html*, 2014.

[47] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.

[48] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Technical Report - Available at https://www.stellar.org/papers/stellar-consensus-protocol.pdf/*, 2014.

[49] G. Greenspan, "Multichain private blockchain-white paper," *Technical Report - Available at http://www.multichain.com/download/MultiChain-White-Paper.pdf*, 2015.

[50] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," in *Leibniz International Proceedings in Informatics, LIPIcs*, 2017.

[51] M. Pustišek, A. Umek, and A. Kos, "Approaching the communication constraints of Ethereum-Based decentralized applications," *Sensors*, vol. 19, no. 11, p. 2647, 2019.

[52] P. Rimba, A. B. Tran, I. Weber, M. Staples, A. Ponomarev, and X. Xu, "Quantifying the Cost of Distrust: Comparing Blockchain and Cloud Services for Business Process Execution," *Information Systems Frontiers*, vol. 22, pp. 489–507, 4 2020.

[53] F. Wessling, C. Ehmke, M. Hesenius, and V. Gruhn, "How much blockchain do you need? towards a concept for building hybrid dapp architectures," in *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 44–47, 2018.

[54] M. Schäffer, M. Di Angelo, and G. Salzer, "Performance and scalability of private ethereum blockchains," in *International Conference on Business Process Management*, pp. 103–118, Springer, 2019.

[55] K. Wüst and A. Gervais, "Do you need a blockchain?," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 45–54, 2018.

[56] H. M. Kim, H. Turesson, M. Laskowski, and A. F. Bahreini, "Permissionless and permissioned, technology-focused and business needs-driven: Understanding the hybrid opportunity in blockchain through a case study of insolar," *IEEE Transactions on Engineering Management*, pp. 1–16, 2020.

[57] V. Buterin, D. Reijsbergen, S. Leonardos, and G. Piliouras, "Incentives in ethereum's hybrid casper protocol," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 236–244, IEEE, 2019.

[58] Z. Zheng, S. Xie, H. N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, pp. 1–23, 2017.

[59] D. Malkhi and M. Reiter, "Byzantine quorum systems," *Distributed Computing*, pp. 203–213, 1998.

[60] L. Law, S. Sabett, and J. Solinas, “How to make a mint: The cryptography of anonymous electronic cash,” *National Security Agency Office of Information Security Research and Technology, Cryptology Division*, 1996.

[61] N. Szabo, “Bit gold,” *Technical Report - Available at https://nakamotoinstitute.org/bit-gold/*, 2005.

[62] W. Dai, “B-Money,” *Technical Report - Available at http://www.weidai.com/bmoney.txt*, 1998.

[63] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[64] M. Correia, G. S. Veronese, N. F. Neves, and P. Verissimo, “Byzantine consensus in asynchronous message-passing systems: a survey,” *International Journal of Critical Computer-Based Systems (IJCCBS)*, vol. 2, pp. 141–161, July 2011.

[65] A. Baliga, “Understanding blockchain consensus models,” tech. rep., Persistent Systems Ltd, 2017.

[66] J. R. Douceur, “The sybil attack,” in *International Workshop on Peer-to-Peer Systems*, pp. 251–260, 2002.

[67] A. Miller and J. J. LaViola Jr, “Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin,” *Technical Report - Available at https://socrates1024.s3.amazonaws. com/consensus.pdf*, 2014.

[68] D. Bradbury, “In blocks [security bitcoin],” *Engineering Technology*, vol. 10, pp. 68–71, Mar. 2015.

[69] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," in *25th IET Irish Signals Systems Conference and China-Ireland International Conference on Information and Communications Technologies (ISSC/CIICT)*, pp. 280–285, June 2014.

[70] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, p. 1, 2012.

[71] Nxt Community, "Nxt whitepaper," *Technical Report - Available at https://nxtdocs.jelurida.com/Nxt_ Whitepaper*, 2014.

[72] P. Vasin, "Blackcoins proof-of-stake protocol v2," *Technical Report - Available at https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf*, 2014.

[73] N. Houy, "It will cost you nothing to kill a proof-of-stake crypto-currency," *Available at SSRN 2393940*, 2014.

[74] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, pp. 34–37, Dec. 2014.

[75] M. Walport, "Distributed ledger technology: Beyond blockchain," *UK Government Office for Science*, 2016.

[76] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[77] C. Cachin, S. Schubert, and M. Vukolić, "Non-determinism in byzantine fault-tolerant replication," *arXiv preprint arXiv:1603.07351*, 2016.

[78] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, "Sawtooth: an introduction," *The Linux Foundation*, 2018.

[79] V. Costan and S. Devadas, “Intel sgx explained.,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 86, 2016.

[80] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, “Blockchain for iot security and privacy: The case study of a smart home,” in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 618–623, 2017.

[81] J. Eberhardt, M. Peise, D. H. Kim, and S. Tai, “Privacy-preserving netting in local energy grids,” in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–9, 2020.

[82] G. Grigoras, N. Bizon, F. M. Enescu, J. M. Lopez Guede, G. F. Salado, R. Brennan, C. O’Driscoll, M. O. Dinka, and M. G. Alalm, “Ict based smart management solution to realize water and energy savings through energy efficiency measures in water distribution systems,” in *Proc. of the 10th International Conference on Electronics, Computers and Artificial Intelligence*, pp. 1–4, 2018.

[83] C. Rottondi and G. Verticale, “A privacy-friendly gaming framework in smart electricity and water grids,” *IEEE Access*, vol. 5, pp. 14221–14233, 2017.

[84] J. Ellehauge, “Blockchain in geospatial applications,” *GIM International - Magazine for Geomatics*, vol. 31, no. 5, pp. 43–45, 2017.

[85] M. Kempe, “The land registry in the blockchain–testbed,” *A development project with Lantmäteriet, Landshypotek Bank, SBAB, Telia company, ChromaWay and Kairos Future*, 2017.

[86] N. Szabo, “Formalizing and securing relationships on public networks,” *First Monday*, vol. 2, no. 9, 1997.

[87] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World.* Portfolio, 2016.

[88] J. Fairfield, “Smart contracts, bitcoin bots, and consumer protection,” *Washington and Lee Law Review Online*, vol. 71, no. 2, pp. 35–50, 2014.

[89] M. Kondo, G. A. Oliva, Z. M. J. Jiang, A. E. Hassan, and O. Mizuno, “Code cloning in smart contracts: a case study on verified contracts from the ethereum blockchain platform,” *Empirical Software Engineering*, vol. 25, no. 6, pp. 4617–4675, 2020.

[90] S. Seebacher and R. Schüritz, “Blockchain technology as an enabler of service systems: A structured literature review,” in *Exploring Services Science*, pp. 12–23, Springer International Publishing, 2017.

[91] J. L. Zhao, S. Fan, and J. Yan, “Overview of business innovations and research opportunities in blockchain and introduction to the special issue,” *Financial Innovation*, vol. 2, no. 1, p. 28, 2016.

[92] N. Kshetri, “Can blockchain strengthen the internet of things?,” *IT professional*, vol. 19, no. 4, pp. 68–72, 2017.

[93] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[94] R. Perrey and M. Lycett, “Service-oriented architecture,” in *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.*, pp. 116–119, 2003.

[95] M. L. Das, "Privacy and security challenges in internet of things," in *Distributed Computing and Internet Technology*, pp. 33–48, Springer International Publishing, 2015.

[96] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[97] S. R. Niya, S. S. Jha, T. Bocek, and B. Stiller, "Design and implementation of an automated and decentralized pollution monitoring system with blockchains, smart contracts, and lorawan," in *Proc. of the IEEE/IFIP Network Operations and Management Symposium*, pp. 1–4, 2018.

[98] M. Samaniego and R. Deters, "Internet of smart things-iost: Using blockchain and clips to make things autonomous," in *2017 IEEE international conference on cognitive computing (ICCC)*, pp. 9–16, IEEE, 2017.

[99] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[100] M. Torky and A. E. Hassanein, "Integrating blockchain and the internet of things in precision agriculture: Analysis, opportunities, and challenges," *Computers and Electronics in Agriculture*, vol. 178, p. 105476, 2020.

[101] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[102] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based iot: Challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, 2017.

[103] M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 94–101, 2018.

[104] V. Gazis, "A survey of standards for machine-to-machine and the internet of things," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 482–511, 2017.

[105] W. Zheng, Z. Zheng, X. Chen, K. Dai, P. Li, and R. Chen, "Nutbaas: A blockchain-as-a-service platform," *IEEE Access*, vol. 7, pp. 134422–134433, 2019.

[106] Z. Shi, H. Zhou, J. Surbiryala, Y. Hu, C. de Laat, and Z. Zhao, "An automated customization and performance profiling framework for permissioned blockchains in a virtualized environment," in *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 404–410, 2019.

[107] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for iot," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, IoTDI '17, (New York, NY, USA), pp. 173–178, ACM, 2017.

[108] S.-C. Cha, J.-F. Chen, C. Su, and K.-H. Yeh, "A blockchain connected gateway for ble-based devices in the internet of things," *IEEE Access*, 2018.

[109] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, "Smart contracts vulnerabilities: a call for blockchain soft-

ware engineering?," in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pp. 19–25, 2018.

[110] S. Huckle, R. Bhattacharya, M. White, and N. Beloff, "Internet of Things, Blockchain and Shared Economy Applications," in *(ICTH-2016)* (Shakshuki, E, ed.), vol. 98 of *Procedia Computer Science*, pp. 461–466, 2016.

[111] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, "Blockchain-oriented software engineering: Challenges and new directions," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pp. 169–171, 5 2017.

[112] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts," in *Principles of Security and Trust*, pp. 164–186, Springer, 2017.

[113] F. Wessling, C. Ehmke, O. Meyer, and V. Gruhn, "Towards blockchain tactics: Building hybrid decentralized software architectures," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 234–237, 2019.

[114] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, and U. Zdun, "Foundational oracle patterns: Connecting blockchain to the off-chain world," in *Business Process Management: Blockchain and Robotic Process Automation Forum*, pp. 35–51, Springer International Publishing, 2020.

[115] K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial internet-of-things: Research challenges and opportunities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3567–3569, 2018.

[116] A. M. Antonopoulos and G. Wood, *Mastering ethereum: building smart contracts and dapps.* O'Reilly Media, 2018.

[117] C. Verdouw, H. Sundmaeker, F. Meyer, J. Wolfert, and J. Verhoosel, "Smart agri-food logistics: requirements for the future internet," *Dynamics in Logistics*, pp. 247–257, 2013.

[118] F. TongKe, "Smart agriculture based on cloud computing and iot," *Journal of Convergence Information Technology*, vol. 8, no. 2, pp. 210–216, 2013.

[119] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 468–477, IEEE, 2017.

[120] A. Ramachandran, D. Kantarcioglu, *et al.*, "Using blockchain and smart contracts for secure data provenance management," *arXiv preprint arXiv:1709.10000*, 2017.

[121] D. McFarlane and Y. Sheffi, *The impact of automatic identification on supply chain operations.* University of Cambridge, Department of Engineering, 2003.

[122] C. Sun, "Application of rfid technology for logistics on internet of things," *AASRI Procedia*, vol. 1, pp. 106–111, 2012.

[123] S. Srinivasan, D. Sorna Shanthi, and A. V. Anand, "Inventory transparency for agricultural produce through iot," *MS&E*, vol. 211, no. 1, p. 012009, 2017.

[124] F. Tian, "An agri-food supply chain traceability system for china based on rfid & blockchain technology," in *2016 13th international conference*

on service systems and service management (ICSSSM), pp. 1–6, IEEE, 2016.

[125] F. Tian, "A supply chain traceability system for food safety based on haccp, blockchain & internet of things," in *2017 International conference on service systems and service management*, pp. 1–6, IEEE, 2017.

[126] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "Iot in agriculture: Designing a europe-wide large-scale pilot," *IEEE communications magazine*, vol. 55, no. 9, pp. 26–33, 2017.

[127] J. Helmbrecht, J. Pastor, and C. Moya, "Smart solution to improve water-energy nexus for water supply systems," *Procedia Eng.*, vol. 186, pp. 101–109, 2017.

[128] E. Wang, S. Attard, A. Linton, M. McGlinchey, W. Xiang, B. Philippa, and Y. Everingham, "Development of a closed-loop irrigation system for sugarcane farms using the internet of things," *Computers and Electronics in Agriculture*, vol. 172, p. 105376, 2020.

[129] M. Á. Pardo Picazo, J. M. Juárez, and D. García-Márquez, "Energy consumption optimization in irrigation networks supplied by a standalone direct pumping photovoltaic system," *Sustainability*, vol. 10, no. 11, p. 4203, 2018.

[130] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of lpwan technologies for large-scale iot deployment," *ICT express*, vol. 5, no. 1, pp. 1–7, 2019.

[131] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in agri-food supply chain management: A practical implementation," in *Proc. of the IoT Vertical and Topical Summit on Agriculture*, pp. 1–4, 2018.

[132] E. A. Abioye, M. S. Z. Abidin, M. S. A. Mahmud, S. Buyamin, M. H. I. Ishak, M. K. I. A. Rahman, A. O. Otuoze, P. Onotu, and M. S. A. Ramli, "A review on monitoring and advanced control strategies for precision irrigation," *Computers and Electronics in Agriculture*, vol. 173, p. 105441, 2020.

[133] B. Bordel, D. Martin, R. Alcarria, and T. Robles, "A blockchain-based water control system for the automatic management of irrigation communities," in *Proc. of the IEEE International Conference on Consumer Electronics*, pp. 1–2, 2019.

[134] B. Ortuani, A. Facchi, A. Mayer, D. Bianchi, A. Bianchi, and L. Brancadoro, "Assessing the effectiveness of variable-rate drip irrigation on water use efficiency in a vineyard in northern italy," *Water*, vol. 11, no. 10, p. 1964, 2019.

[135] J. Huan, H. Li, F. Wu, and W. Cao, "Design of water quality monitoring system for aquaculture ponds based on nb-iot," *Aquacultural Engineering*, vol. 90, p. 102088, 2020.

[136] A. Khanna and S. Kaur, "Evolution of internet of things (iot) and its significant impact in the field of precision agriculture," *Computers and Electronics in Agriculture*, vol. 157, pp. 218 – 231, 2019.

[137] S. Cheung, "Reinventing the bazaar: A natural history of markets," *The Economic Journal*, vol. 113, 2003.

[138] H. Subramanian, "Decentralized blockchain-based electronic marketplaces," *Commun. ACM*, vol. 61, no. 1, pp. 78–84, 2017.

[139] V. P. Ranganthan, R. Dantu, A. Paul, P. Mears, and K. Morozov, "A decentralized marketplace application on the ethereum blockchain,"

in *Proc. of the IEEE 4th Int. Conf. on Collaboration and Internet Computing*, pp. 90–97, 2018.

[140] G. S. Ramachandran, R. Radhakrishnan, and B. Krishnamachari, "Towards a decentralized data marketplace for smart cities," in *Proc. of the IEEE Int. Conf. on Smart Cities*, pp. 1–8, 2018.

[141] M. L. Di Silvestre, P. Gallo, M. G. Ippolito, E. R. Sanseverino, G. Sciumè, and G. Zizzo, "An energy blockchain, a use case on tendermint," in *Proc. of the IEEE Int. Conf. on Environment and Electrical Engineering*, pp. 1–5, 2018.

[142] A. Hahn, R. Singh, C. Liu, and S. Chen, "Smart contract-based campus demonstration of decentralized transactive energy auctions," in *Proc. of the IEEE Conf. on Power Energy Society Innovative Smart Grid Technologies*, pp. 1–5, 2017.

[143] B. Pittl, W. Mach, and E. Schikuta, "Bazaar-blockchain: A blockchain for bazaar-based cloud markets," in *Proc. of the IEEE Int. Conf. on Services Computing*, pp. 89–96, 2018.

[144] L. Mikkelsen, K. Mortensen, H. Rasmussen, H. Schwefel, and T. Madsen, "Realization and evaluation of marketplace functionalities using ethereum blockchain," in *Proc. of the Int. Conf. on Internet of Things, Embedded Systems and Communications*, pp. 47–52, 2018.

[145] V. Scoca, R. B. Uriarte, and R. D. Nicola, "Smart contract negotiation in cloud computing," in *Proc. of the IEEE 10th Int. Conf. on Cloud Computing*, pp. 592–599, 2017.

[146] J. A. F. Castellanos, D. Coll-Mayor, and J. A. Notholt, "Cryptocurrency as guarantees of origin: Simulating a green certificate market

with the ethereum blockchain," in *Proc. of the IEEE Int. Conf. on Smart Energy Grid Engineering*, pp. 367–372, 8 2017.

[147] A. Pinna, S. Ibba, G. Baralla, R. Tonelli, and M. Marchesi, "A massive analysis of ethereum smart contracts empirical study and code metrics," *IEEE Access*, vol. 7, pp. 78194–78213, 2019.

[148] M. Pincheira, M. Vecchio, and R. Giaffreda, "Rationale and practical assessment of a fully distributed blockchain-based marketplace of fog/edge computing resources," in *2020 Seventh International Conference on Software Defined Systems (SDS)*, pp. 165–170, 2020.

[149] M. Pincheira, M. Vecchio, R. Giaffreda, and S. S. Kanhere, "Exploiting constrained iot devices in a trustless blockchain-based water management system," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–7, 2020.

[150] M. Spain, S. Foley, and V. Gramoli, "The impact of ethereum throughput and fees on transaction latency during icos," in *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[151] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: towards sustainable local energy markets," *Computer Science-Research and Development*, vol. 33, no. 1-2, pp. 207–214, 2018.

[152] European Space Agency, "Blockchain and earth observation," in *Whitepaper*, 5 2019.

[153] M. J. Molesky, E. A. Cameron, J. Jones, M. Esposito, L. Cohen, and C. Beauregard, "Blockchain network for space object location gathering," in *2018 IEEE 9th Annual Information Technology, Electronics*

and Mobile Communication Conference (IEMCON), pp. 1226–1232, 11 2018.

[154] Y.-P. Lin, J. Petway, J. Anthony, H. Mukhtar, S.-W. Liao, C.-F. Chou, and Y.-F. Ho, "Blockchain: The evolutionary next step for ict e-agriculture," *Environments*, vol. 4, no. 3, p. 50, 2017.

[155] E. Leka, L. Lamani, B. Selimi, and E. Deçolli, "Design and implementation of smart contract: A use case for geo-spatial data sharing," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1565–1570, IEEE, 2019.

[156] Kiran Nagaraja, S. Rollins, and M. Khambatti, "From the editors: peer-to-peer community: looking beyond the legacy of napster and gnutella," *IEEE Distributed Systems Online*, vol. 7, no. 3, pp. 5–5, 2006.

[157] E. Nyaletey, R. M. Parizi, Q. Zhang, and K. R. Choo, "Block-ipfs - blockchain-enabled interplanetary file system for forensic and trusted data traceability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019.

[158] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," *Technical Report - Available at http://storj.io/storj.pdf*, 2014.

[159] D. Vorick and L. Champine, "Sia: Simple decentralized storage," *Technical Report - Available at https://sia.tech/sia.pdf*, 2014.

[160] S. Williams, V. Diordiiev, L. Berman, and I. Uemlianin, "Arweave: A protocol for economically sustainable information permanence," *Technical Report - Available at https://www.arweave.org/yellow-paper.pdf*, 2019.

[161] H. Huang, J. Lin, B. Zheng, Z. Zheng, and J. Bian, "When blockchain meets distributed file systems: An overview, challenges, and open issues," *IEEE Access*, vol. 8, pp. 50574–50586, 2020.

[162] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[163] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*, pp. 369–378, Springer, 1987.

[164] A. Galletta, J. Taheri, and M. Villari, "On the applicability of secret share algorithms for saving data on iot, edge and cloud devices," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 14–21, 2019.

List of Abbreviations

ASC Advertisement Smart Contract.

BLE Bluetooth Low Energy.

CPU Central Processing Unit.

DAO Decentralized Autonomouse Organization.

DDos Distributed Denial of Service.

DFS Decentralized File System.

DLT Distributed Ledger Technology.

DSR Design Science Research.

ECDSA Ellipctic Curve Digital Signing Algorithm.

ETH Ethereum cryptocurrency ETHER.

FBFT Federated Byzantine Fault Tolerance.

FECR Fog-Edge computing resources.

IoT Internet of Things.

IPFS InterPlanetary File System.

IT Information Technologies.

LCA Life Cycle Assessment.

LPWAN Low Power Wide Area Network.

MCU MicroController Unit.

MSC Market Smart Contract.

OTAA Over-The-Air-Activation.

PBFT Practical Byzantine Fault Tolerance.

PoA Proof of Authority.

PoC Proof of Concept.

PoET Proof of Ellapsed Time.

PoS Proof of Stake.

PoW Proof of Work.

RLP Recursive Length Prefix.

RSC Reservation Smart Contract.

SF Spreading Factor.

SLA Service Level Agreements.

ST Smart Twin Smart Contract.

STA Smart Twin Application Smart Contract.

TEE Trusted Execution Environment.

UML Unified Modeling Language.

UNL Unique Node List.