



JOÃO GOULÃO

BSc in Electrical and Computer Engineering

Electronic Notary Environment for Collaborative Ecosystems

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon

September, 2023

Electronic Notary Environment for Collaborative Ecosystems

JOÃO GOULÃO

BSc in Electrical and Computer Engineering

Adviser: Ana Inês Oliveira

Assistant Professor, NOVA University Lisbon, School of Science and Technology | FCT NOVA

Examination Committee:

Chair: Bruno João Nogueira Guerreiro,
Assistant Professor, NOVA University Lisbon

Rapporteurs: Ana Inês da Silva Oliveira,
Assistant Professor, NOVA University Lisbon
Filipa Alexandra Moreira Ferrada,
Assistant Professor, NOVA University Lisbon

Adviser: Ana Inês da Silva Oliveira,
Assistant Professor, NOVA University Lisbon

Electronic Notary Environment for Collaborative Ecosystems

Copyright © João Goulão, NOVA School of Science and Technology, NOVA University Lisbon.

The NOVA School of Science and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

To my Family and Friends.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor, Ana Oliveira, who offered her continuous guidance and support throughout the last year of my masters degree. Her initiative and active commitment to helping and motivate me even at the more challenging points in my personal life were absolutely crucial for this work. Without her support not only derived from her professional knowledge but also her personal character this work wouldn't be possible.

I want to thank the NOVA School of Science and Technology for providing me with wonderful and valuable educational experiences and the professors which helped me shape into a better person.

Biggest thanks to all my family that have supported me throughout all my life, specially to my parents and my sister that have nurtured me to follow the fields of science and engineering since a young age and supporting my educational ambitious. My uncle, João Marcelo, and my cousin, Gonçalo Marcelo for always being positive influences throughout my life providing me with inspiration and suggestions. My grandparents which deeply assisted my parents in raising me and my sister and cared for us.

I would also want to show appreciation to my hometown friends, some whom I have known for most of my life as we watched ourselves grow and change throughout the years and helped me overcome many life challenges. A special thanks to João David, Pedro Peres, Mykyta Deyneha, João Castilho, João Ribeiro, Martim Nogueira and Márcia Vital. Also, my college friends that have I formed bonds with and endured many challenges together, specially to Lourenço Soares, Daniel Corte-Real, Rodrigo Teixeira, João Santos, Fábio Pires and Eduardo Teixeira. Finally, a special thanks to my cousin and most trusted friend Alexandre Goulão.

ABSTRACT

The rise of new industrial and consumer standards along with a new emerging paradigm where globalization and information technology make the world more interconnected and co-dependent. At the same time, we experience more economic disruptions and market turbulence that have made the necessity for new ways for organizations to collaborate and overcome these problems. Since collaboration ecosystems require collaboration agreements and documentation to regulate their behavior and exchange necessary information among its members, one major challenge that the science of collaborative networks faces is how the lack of trust between members and organizations hinders the process of collaboration, contrary to the benefits of being a part of collaborative communities. This challenge has brought up new research and efforts to mitigate the lack of trust and create new tools for collaborative communities to forge new bonds of trust among the members. One of the tools that virtual communities always use is notaries to store and vote on agreement documents. Until now, most electronic notaries have been regular centralized notaries where members put trust in a third-party company to safely store the document. This thesis aims at creating a new type of electronic notary using the technology of blockchain networks that can increase the trust and security of electronic notaries. Blockchain technology has come a long way. New tools have been created that enable an easier-than-ever way of creating these networks and using them for purposes other than financial and monetary uses. This thesis shows all the steps taken to create a private blockchain network that can work as a decentralized notary for collaborative network members to store hash files of contracts while granting permission for other members to vote on them. The project concludes that not only is it feasible and relatively easy to implement such a solution but it also makes it easier to detect any malicious editing in the stored information.

Keywords: Collaborative Networks, Blockchain technology, Electronic Notary.

RESUMO

O surgimento de novos padrões industriais e de consumo, juntamente com um novo paradigma emergente em que a globalização e a tecnologia da informação tornam o mundo mais interligado e co dependente, enquanto se verificam mais perturbações económicas e turbulências no mercado, tornaram necessárias novas formas de colaboração entre as organizações para ultrapassar estes problemas. Um dos principais desafios que a ciência das redes de colaboração enfrenta é o facto de a falta de confiança entre membros dificultar o processo de colaboração e os benefícios da colaboração. Este desafio tem suscitado novas investigações e esforços para mitigar a falta de confiança e criar ferramentas para que as comunidades colaborativas forjem novos laços de confiança entre os membros. Uma das ferramentas que as comunidades virtuais utilizam são os notários para armazenar e votar em documentos contratuais. Até agora, a maioria dos notários têm sido notários centralizados regulares, em que os membros confiam num terceiro para armazenar o documento em segurança. Esta tese tem como objetivo criar um tipo de notário utilizando a nova tecnologia de blockchain que pode aumentar a confiança e a segurança dos notários. A tecnologia blockchain tem ganho popularidade e foram desenvolvidas ferramentas que permitem criar estas redes de forma mais fácil e utilizá-las para outros fins que não os financeiros e monetários. Este projeto mostra todos os passos dados para criar uma rede privada de blockchain que pode funcionar como um notário descentralizado para os membros da rede colaborativa armazenarem ficheiros de contratos e darem permissão a outros membros para votarem nos mesmos. O estudo conclui que não só é viável e relativamente fácil implementar uma solução deste tipo, como também facilita a deteção de qualquer edição maliciosa na informação armazenada.

Palavras chave: Redes Colaborativas, Tecnologia Blockchain, Notários Eletrónicos.

CONTENTS

1	INTRODUCTION.....	1
1.1	Context and Motivation	1
1.2	Problem Statement.....	3
1.3	Document Structure	3
2	BACKGROUND AND RELATED WORK.....	5
2.1	Collaborative Networks.....	5
2.1.1	Virtual Organization Breeding Environments	10
2.1.2	Virtual Organizations.....	13
2.2	Electronic Notary	17
2.2.1	Digital Signatures	18
2.2.2	Notary Management.....	20
2.3	Blockchain Technology.....	21
2.3.1	Blockchain General Structure.....	22
2.3.2	Consensus Models	24
2.3.3	Types of Blockchains	27
2.3.4	Blockchain Frameworks.....	29
2.3.5	Blockchain Applications	30
2.4	Relation to Current Work	33
3	SYSTEM DESIGN.....	35
3.1	Context Identification	35

3.2	Design Requirements.....	36
3.2.1	User Requirements.....	36
3.2.2	Network Requirements.....	36
3.2.3	Use Cases.....	38
4	SYSTEM DEVELOPMENT.....	41
4.1	Framework for e-notary blockchain network.....	41
4.2	Ethereum Client.....	44
4.3	Network Creation	46
4.3.1	Genesis File Configuration	46
4.3.2	Launching The Network.....	49
4.3.3	Consensus Algorithm.....	54
4.4	E-Notary Smart-Contract.....	56
4.5	Application Interface	59
4.6	Cloud Deployment.....	62
5	RESULTS & TESTING.....	65
5.1	Blockchain Network Results	65
5.2	Contract Interaction.....	66
5.3	Application Testing	68
6	CONCLUSION	71
6.1	Application Testing	71
6.2	Limitations & Future Work.....	72
	BIBLIOGRAPHY	75
	ANNEX A: SOURCE CODE	81

LIST OF FIGURES

Figure 2.1: Different definitions and how they correlate in terms of interaction and integration.	6
Figure 2.2: Classification of Collaborative Networks, adapted from (Camarinha 2008).	8
Figure 2.3: Different kinds of topologies adapted from (Durugbo, 2016).	9
Figure 2.4: Stages of VO lifecycle.	13
Figure 2.5: VO creation in a VBE context. From (Camarinha-Matos & Afsarmanesh 2007).	15
Figure 2.6: Process of digital signature creation and verification.	19
Figure 2.7: Overview of notary functions. From (Oliveira, 2016).	21
Figure 2.8: Example of Merkle tree root calculation.	23
Figure 2.9: Representation of how blocks interlink. From (Yaga et al., 2018).	23
Figure 2.10: Flow chart of the proof-of-work algorithm.	25
Figure 2.11: Different types of blockchain networks.	28
Figure 2.12: Representation of the blockchain trilemma.	29
Figure 2.13: Overview of the healthcare process. From (Shahnaz et al., 2019).	32
Figure 2.14: Representation of how IoT devices interact with the blockchain. From (Novo, 2018).	32
Figure 3.1: Overview of the proposed solution.	37
Figure 3.2: Use case diagram of the notary usage.	38
Figure 3.3: Sequence diagram of the interactions.	39
Figure 3.4: System conceptual architecture.	39
Figure 4.1: Screenshot of the list of most used Ethereum clients, taken from https://www.ethernodes.org/ .	44
Figure 4.2: Architecture of a EVM for Hyperledger Besu.	46
Figure 4.3: Genesis file configuration.	47

Figure 4.4: Account creation and management screen.	50
Figure 4.5: Directory organization of the nodes	51
Figure 4.6: Bootnode launching setup in Hyperledger Besu	51
Figure 4.7: Node URL of the bootnode.....	53
Figure 4.8: Configuration command to launch a node and connect to the bootnode.	53
Figure 4.9: Command to verify the validator nodes.....	53
Figure 4.10: Result of the command, displaying the addresses of the validator nodes.....	54
Figure 4.11: Illustration of the consensus algorithm.....	55
Figure 4.12: Command to vote on adding a new validator node to the network.	56
Figure 4.13: List of validators displaying the newly added node into the network.....	56
Figure 4.14: a) Network selection screen. b) Contract function's screen.....	58
Figure 4.15: a) Metamask connected to Remix IDE. b) Metmask connected.....	59
Figure 4.16: Layout of the dapp interaction with the e-notary blockchain network.....	60
Figure 4.17: Base code to verify if the user is using the Metamask extension.....	61
Figure 4.18: Front-end function to call the smart contract function when one of the interface buttons is pressed.	62
Figure 4.19: Configuration command used to launch the bootnode inside the cloud virtual machine.....	63
Figure 4.20: High-level architecture of the solution with the cloud deployment.	64
Figure 5.1: Screenshot of console monitoring the process of a node creating and importing blocks.	66
Figure 5.2: Metamask activity log of the contract deployment.	66
Figure 5.3: a) Storing a file through Remix IDE b) Metamask activity log of the interaction. ...	67
Figure 5.4: Notary application interface.....	68
Figure 5.5: a) When a file is successfully stored. b) When a vote is cast by a valid user with permission from the owner.	69
Figure 5.6: a) When a user tries to store the same file. b) Message when a user who is not the original uploader of a file.	69

LIST OF TABLES

Table 2.1: Table of VBE characteristics.....	11
Table 2.2: Different types of roles inside a VBE.....	12
Table 2.3: List of different VBE functionalities.	13
Table 2.4: Table comparing different frameworks.	30
Table 4.1: Generic commands in Ethereum blockchain networks.....	42
Table 4.2: Smart Contract Functions.....	57

ACRONYMS

ABI	Application binary interface
CN	Collaborative Network.
CNO	Collaborative Networked Organizations.
CURL	Client URL
Dapp	Decentralized Application
DSA	Digital Signature Algorithm.
ECDSA	Elliptic Curve Digital Signature Algorithm.
ECOLEAD	European Collaborative Networked Organizations leadership initiative.
E-Notary	Electronic Notary.
EVM	Ethereum Virtual Machine
IBFT	Istanbul Byzantine Fault Tolerance
ICT	Information and Communication Technologies.
IoT	Internet of Things.
JSON-RPC	JSON remote procedure call
PoW	Proof-of-Work.
PoA	Proof-of-Authority.
RSA	Rivest-Shamir-Adleman.
SHA	Secure Hash Algorithm.

SME	Small and medium-sized enterprises.
VBE	Virtual Organization Breeding Environment.
VO	Virtual Organization.
WizAN	Agreement Negotiation Wizard.

INTRODUCTION

This chapter introduces the emergence of collaborative networks, and some history regarding blockchain networks. The objectives of the thesis and general summary for each chapter are also summarized.

1.1 Context and Motivation

Industry 4.0, also known as the fourth industrial revolution, is the current trend of automation and data exchange. The need for Industry 4.0 arose from several factors, including the desire for greater efficiency, customization, and automation in manufacturing processes, as well as the ability to collect and analyze large amounts of data. The advancements in technology, such as the Internet of Things (IoT), cloud computing, and big data analytics, made possible the connection of physical and digital systems in a way that was not possible before considering the vision of Industry 4.0. Moreover, more precise control of production processes, results in increased productivity and improved product quality (J. H. Kim, 2017).

As more and more companies adopt advanced technologies and automation, those who do not risk are being left behind. These new technologies make it possible for companies to adapt to changing market conditions and customer needs, leading to greater flexibility and responsiveness to demands.

However, there are several challenges these new technological trends bring to companies, such as the high costs to implement new technologies, compatibility issues, lack of flexibility and adaptation to changing markets, and lack of standardization among tools and operating methods that make it difficult for small and medium-sized enterprises (SME) or companies to compete with larger organizations (Afsarmanesh & Camarinha-Matos, 2005;

Ericson et al., 2020). All these challenges can many times create barriers, as small companies don't have either the budget to continuously change, nor the proper infrastructure to deal with new demands from customers. Furthermore, global economic instabilities and ever-changing political landscapes create huge disruptions in supply chains and national markets, products that were once outcompeted by a few national companies quickly find themselves no longer operating in some markets, and the lack of certain goods leads to new products being sought after or entirely new options to solve certain needs.

With these challenges it has become clear that small and medium-sized companies need to collaborate and change the way they perform business to better adapt to the new problems and challenges. Collaboration can lead to the creation of standards, and knowledge sharing that enables greater specialization, and most importantly reducing costs while increasing flexibility. Collaboration and communication have become the tool that enables companies to increase their competitiveness when faced with larger organizations (Camarinha-Matos & Afsarmanesh, 2007).

A key component that enables collaboration between people and organizations is trust, and since collaboration needs companies to share costs, risks, information, and decision-making power (Camarinha-Matos et al., 2013; Folke et al., 2005). Without trust, companies will become reluctant to share these resources and ultimately hinder their chances of striving and achieving the full potential that collaboration could bring.

In today's increasingly digital landscape, characterized by the ubiquity of cyber communication, the imperative role of contracts and agreements in bolstering trust among organizations cannot be overstated (de Oliveira Albuquerque et al., 2016). As the pace of forming contractual relationships accelerates in response to the demands of our rapidly evolving business environment, the need for a reliable mechanism to securely store and authenticate the identities of the parties involved has grown exponentially in significance. The absence of such a mechanism leaves organizations vulnerable to doubt and hesitation when faced with requests to share resources. This vulnerability, in turn, can precipitate a cascade of inefficiencies and palpable erosion of trust, rendering it paramount for organizations to have robust contracts and agreements that encompass robust authentication methods.

Blockchain technology has come a long way since its first inception, more recently a lot of research has been conducted to use its main attributes as a decentralized network for other projects beyond digital currency (Bhutta et al., 2021). With its recent developments and newer upgrades in recent years having not only new features but also frameworks that can greatly facilitate the development of new dynamic networks, blockchain networks have never been

easier to implement. The importance blockchain networks bring into the context of collaboration increasing trust among companies is based on the fact that they offer a transparent and secure way to store and validate contracts with immutability and validate the identity of their participants (Fiorentino et al., 2020).

1.2 Problem Statement

Considering the context and motivation mentioned before, this thesis aims to research and develop a test prototype blockchain network that serves as an electronic notary (e-notary) for companies and entities to collaboratively store documents in the context of the subject of collaborative networks as a way to have a decentralized database, that enables trust, and a dynamic environment that can secure documents and manage who can have access to them and cast vote if allowed by the original owner. Some of the main objectives are:

- Research and test the different available frameworks and clients for blockchain development, and what options and tools they provide to create a blockchain that suits the needs previously mentioned.
- Careful study of the needs and attributes that will be needed for the blockchain to have. Which ones are already available or can be developed, and how difficult is their implementation.
- Create the blockchain network, simulate and test it. Study and evaluate its strengths and weaknesses as well as the difficulty of implementation. How well does it perform and how much information can it process depending on the hardware it's being used.
- Can the current blockchain frameworks provide a blockchain network for the purpose of an electronic notary for collaborative ecosystems, and which prerequisites will this community requires to have.

1.3 Document Structure

Introduction: The first chapter presents the context that leads to blockchain research, the problem of new technologies and trends in the economy, and the new pressures and

challenges these have for small and medium-sized companies which leads to an increase in collaboration and thus trust. How blockchain networks can help companies increase their trust when signing agreements online and rapidly.

Background and Related Work: Chapter two presents the subject of collaborative networks as a scientific discipline where the research development of this thesis is based. It introduces the related concepts that support organizational collaboration and contextualize the need for an electronic notary. It also makes a small overview of projects that developed different collaboration environments and their electronic notaries, stressing how documents are signed and functionalities are provided. Lastly it is analyzed and showcased an overview of how blockchain networks operate, their different types, consensus algorithms, frameworks, and trends in recent applications.

System Design: In the third chapter the network designing process of the blockchain network is presented. The different requirements for such an e-Notary are identified, also the required infrastructure, and the different use cases are defined.

System Development: In the fourth chapter all the sequential steps taken to build the network and deploy it to the goals set in the previous chapter are presented, starting from the choice of the framework, the client, the different tools that helped facilitate the development of this thesis work. Furthermore, the configuration of the genesis file, the nodes in the network and the deployment of the functionalities to adjust to the requirements of the electronic notary are also described. Finally, the launching in a cloud infrastructure is described.

Results & Testing: This chapter showcases the network working highlighting the nodes proposing and validating different blocks. The results are shown through the built application that permits the interaction of the network with the different e-notary functionalities.

Conclusions: The final chapter consists of conclusions and discusses the results of the previous chapters, what problems and flaws the network can face, what possible solutions could have, and any possible future work that could improve the blockchain network.

BACKGROUND AND RELATED WORK

The development of a blockchain e-notary solution for collaborative networks involves many different subjects. In this chapter, there is a comprehensive description of the discipline of collaborative networks and the terminologies used in this field. Secondly, a very brief description of the concept of electronic notaries and how public-private key authentication works is presented. Lastly, there is an overview of the core concepts of blockchain networks, the different characteristics they can have, and recent projects that tried to utilize it for other means than the traditional monetary one.

2.1 Collaborative Networks

Collaborative networks (CNs) is a scientific discipline that deals with and studies the development of systems that facilitate collaboration and cooperation among companies, organizations, and people. The term was first coined by (Camarinha-Matos & Afsarmanesh, 2005), this field emerged because of the increasing importance of collaboration and interorganizational relationships and management practices that have been stressed in the last few decades with the emergence of a more global economy. It was suggested that traditional organizational structures and practices were not equipped to handle the new complex and dynamic nature of modern businesses and market instabilities, as well proved a major concern for places such as Europe where most of its economies are composed of small and medium-sized enterprises (SMEs). Furthermore, with the development of new information and communication technologies (ICT) new opportunities and possibilities were made available to face these problems and adapt to these new paradigms that include forms of organizing

business and different ways to enable collaboration to promote efficiency, innovation, and 'shockproof' to abrupt changes (Camarinha-Matos, 2014).

With all this in mind, a collaborative network can be defined in the simplest way as being: A type of network in which people and organizations work together and share resources to achieve a common goal with the help of ICT platforms or virtual networks to organize, coordinate and exchange information (Camarinha-Matos et al., 2008).

It is possible to make a clear-cut separation between what can be defined as collaboration in this sense and how collaboration requires a higher level of integration than other forms that need less interaction as shown in Figure 2.1.

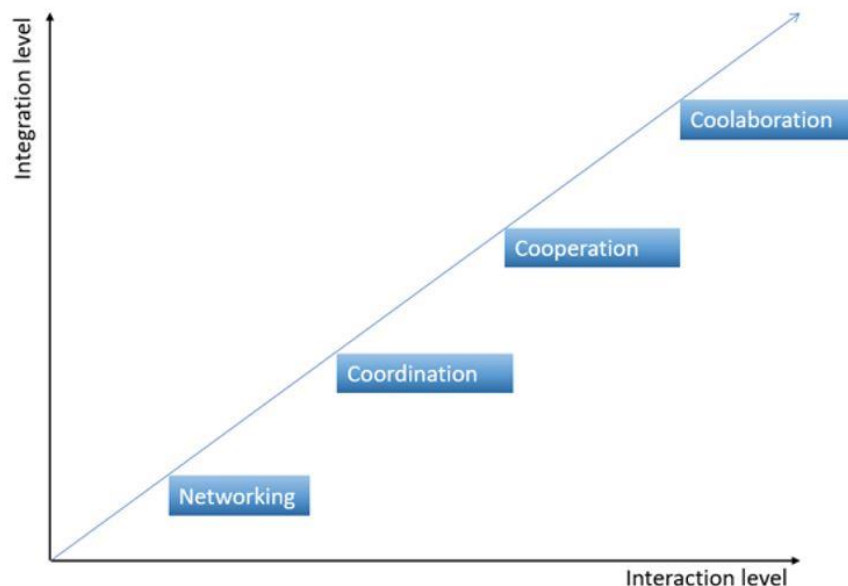


Figure 2.1: Different definitions and how they correlate in terms of interaction and integration.

An overview of what the definitions in figure imply is listed as follows:

- **Networking:** Exchange of information and resources.
- **Coordination:** Agents with different roles and objectives team up and coordinate their skills to achieve individual objectives, exchanging information and resources in the process.
- **Cooperation:** All involved agents share a common goal and divide tasks and activities to reach the objective. They share information and resources in the process, but their tasks remain mostly individual.

- **Collaboration:** Agents work together to make something new by sharing resources and skills, with multiple tasks and activities being shared along with information.

The study and fostering of CNs among enterprises, organizations and people can bring several advantages to economies such as increased competitiveness and access to new markets and customers due to being able to share resources and knowledge to tackle larger projects. This also leads to improved innovation and creativity with a bigger exchange of ideas between partners working in the same field or completely different fields as shown by different surveys on their effects on collaboration among SME enterprises (Mulyana & Wasitowati, 2021).

In the field of CNs, there is a vast array of taxonomy and network classification to categorize the different types of collaborative networks and their strengths and weaknesses, (Camarinha-Matos, 2014; Camarinha-Matos & Afsarmanesh, 2021). One way used to describe, as shown in Figure 2.2, can be through a first distinction of two main kinds, starting from "ad-hoc collaboration networks" and "collaborative networked organizations" (CNOs). Ad-hoc networks are characterized by being the standard, often informal networks of organizations created in a spontaneous way to address a specific task, with no agreement rules or communication protocols thus being not really effective at solving more complicated tasks that could involve more flexibility. On the opposite, collaborative networks induce favorable organizational structures for supporting collaboration, CNs can be further divided into two types which are distinguished by "goal-oriented networks" and "long-term strategic networks".

Long-term strategic networks are made with the objective to help their participants create goal-oriented networks by offering an environment tailored to incentivize more communication and tools for easier formation of groups and finding partners when a business opportunity arises when a member alone could not take. There are Professional virtual communities, which are an online community that joins individual professionals from a specific field to share knowledge and tackle projects. Examples of this can be any community of freelancers in a common platform. There are also virtual organization breeding environments (VBEs) which are a community composed of organizations, enterprises that use online platforms to come together to work on a business opportunity that alone couldn't be tackled. VBEs provide the necessary infrastructure and tools to effectively enable its measures to collaborate and form goal-oriented networks that can successfully operate. Such tools can go from common communication standards, service contract making tools for managing projects, with member and fluid roles, resources, and tasks they can change to. More specific manifestations of this kind of network could be business ecosystems and industry clusters.

Goal-oriented networks can be described as networks that are usually made through a long-term strategic network to reach or achieve a specific goal or objective. They tend to have a high degree of specialization, assign very distinct and clear roles for their participants, and tend to remain stable and fixed once formed, during the accomplishment of the business/project they were created to.

Goal-oriented networks can be divided into continuous production-driven networks which are organized around the continuous production and distribution of goods and services, they tend to focus on efficiency, reliability predictability, and standardized processes and routines. Examples can be supply chains and manufacturing networks. On the other hand, as grasping opportunity networks are organized around the pursuit of new opportunities and unexpected events. They are characterized by their flexibility and adaptability to quickly assemble and reorganize resources. Examples of these networks can be called virtual organizations (VOs).

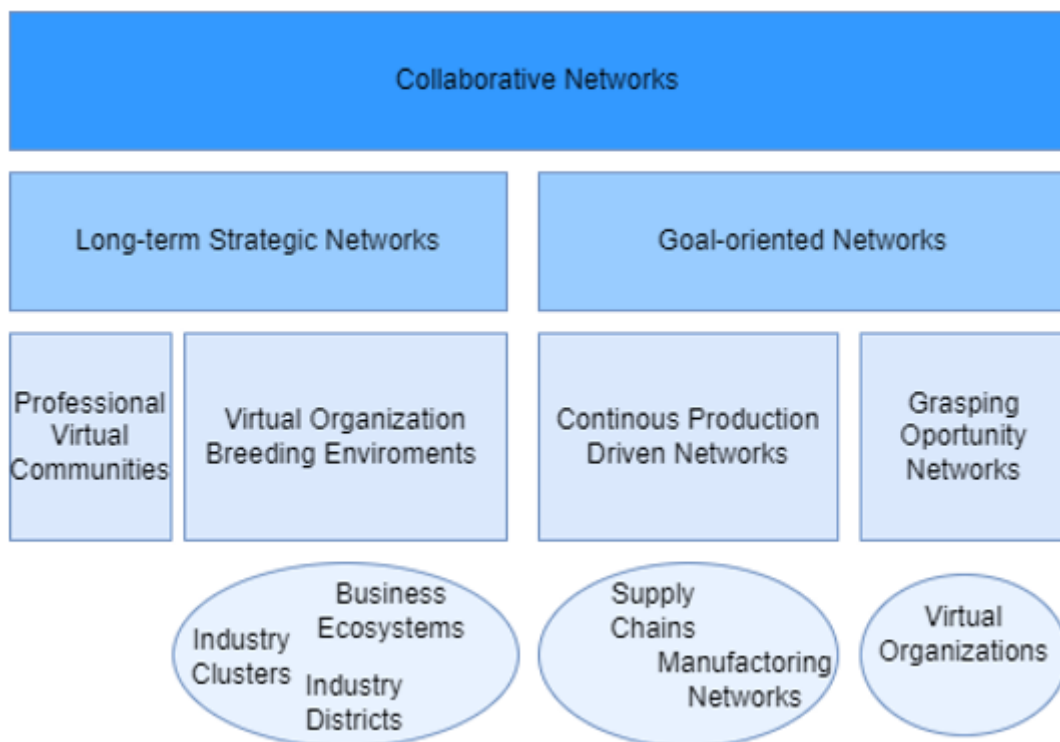


Figure 2.2: Classification of Collaborative Networks, adapted from (Camarinha 2008).

There are other authors that try to focus more on the classification of different types of collaborative networks through their arrangement and behavior (Durugbo, 2016), claiming that a classification of enterprise networks, like this, can better understand the underlying logic

behind the network, how prepared it is for any type of change and how its participants benefit and contribute is weighted and measured.

In terms of structure logic, a broad separation could be made on how horizontal or vertical the topology. The type of topology will depend vastly on the specific needs of the organization, the sizes of the enterprises, and complexity, as such, different topologies can be effective in different situations. In a more concrete example, we can have three major types of configurations for collaborative networks can be considered: star, tree, and torus topologies as illustrated in Figure 2.3. Star topologies have a central participant that takes a leading role and usually manages all the tasks, delegates work, and coordination. Usually, it is composed of a large enterprise that works with smaller ones.

Tree topology is with a bigger focus on a hierarchy and chain of command, usually fit for more specific tasks that need more coordination.

Torus topologies are characterized by organizations that can be connected in a ring shape or a "horizontal" organization where communication between different units is way more integrated as a partner needs to communicate with multiple entities at once and not just a higher up, these topologies while being more flexible require way more tools for communication and coordination.

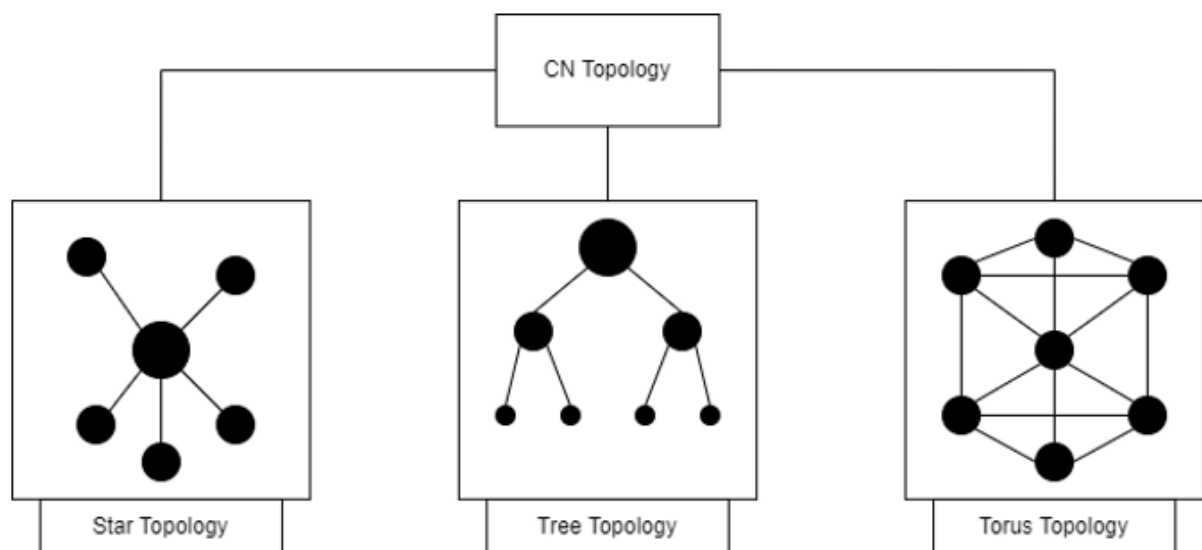


Figure 2.3: Different kinds of topologies adapted from (Durugbo, 2016).

For the behavioral logic, it has been noted that trust is one of the most important factors in the functioning of a CN as it has the most effect on the behavior of the participants. There are many forms of trust as described in related literature, trust could manifest in terms of contracts, management or socially. Trust has been identified as the key factor that allows

communities to share information and resources thus leading more openly to increased productivity and efficiency working towards their goals. On the other hand, if the trust between partners is low, they may be more hesitant to share information and resources, communicate less openly or simply doubt the authenticity of choices or requests from one another. Overall, as trust increases between partners it results in an Increase of exchanges between members resulting in an increase in more specialized domain knowledge and expertise.

2.1.1 Virtual Organization Breeding Environments

As mentioned before, the process of creation of virtual organizations (VOs) is assumed to take place within a long-term strategic collaborative network on which its members adopt long-term cooperation agreements and standardize the necessary communication tools with the sole goal of facilitating the VO formation process when a business opportunity arises. To put it another way, the main goal of VBEs is to change the way businesses and people do usual ad-hoc partnerships to proper goal-oriented collaboration networks.

According to (Afsarmanesh et al., 2009), networks such as IECOS, CeBeNetwork, Helicenetwork, NetworkA, Virtuelle Fabrik, and Torino Wireless are a few examples of networks that serve as early VBE examples. These networks, which may be referred to as first-generation VBEs, are mostly restricted to geographical locations, still relying on few ICT tools, and guided by a constrained conceptual framework. Observing and collecting a significant amount of empirical knowledge and requirements from these networks served as the foundation for the development of a new model and conceptual framework for VBEs as well as for the dynamic creation of VOs within VBEs and the corresponding advanced support ICT infrastructures and tools, giving rise to the second generation of VBEs.

Nevertheless, there are still some main problems and paradigms that VBEs need to answer and focus on, these can be summarized as follows:

- How to gather the information and standardize and create profiles for each member and participating organization?
- How to manage the trust among organizations, and what metrics can we use to evaluate their performance?
- What infrastructure is needed to establish communication and enable collaboration in an environment where multiple types of organizations can reside?
- What basic ontology can be established for the industry the VBE is targeting, minimizing the ambiguity that might arise, for instance, from different cultural heritages or different industry environments?

In (Gasparotto & Guerrini, 2013; Romero et al., 2008) a systematic study has been carried out to investigate what are the different topologies and classes of VBEs. For the classification several characteristics were analyzed and noted which forms they would take in studied VBEs. Some of the more noteworthy ones are summarized in Table 2.1.

Table 2.1: Table of VBE characteristics.

Characteristic	Possible Variations	
VBE Funding	Publicly funded	Member's fee
Location	Regional	Non-Regional
Diversity of products	Specialized	Diverse
Sector	Single-Sector	Multi-Sector
Size	Small	Large

After identifying the main types of criteria and characteristics present in VBEs, and although each VBE proved to be unique, further research carried by (Afsarmanesh & Camarinha-Matos, 2009) to see if a larger pattern could be seen that would lead to different topologies of VBEs. With this, a broader separation between the types of characteristics seen in Table 2.1 was made and tried to verify which ones were dominant categories, drivers for collaboration, or just values arriving at four different types of domain categories:

- The "stable products domain VBE type" is focused on regional clusters of organizations that are very business and social-oriented. Their function roles are already very entrenched, and their activities focus solely on cyclic tasks. These types of VBEs don't take use or tools for management, that enable trust or try to improve the VBE for new demands in the market.
- "One-of-a-kind domain VBE" is more focused on VOs of long duration. Its members have a high trust level between each other, make use of ICT tools and develop unique products.
- "Emerging domain VBE" is described as a VBE that was formed to respond to a new market need which causes the merging of organizations from very different domains with competences that often complement each other. These also make use of ICT tools to help increase the trust and rules for the environment.
- Lastly the "Innovation-driven domain VBE" is usually focused on making VOs of very short duration to design and develop new innovative products or conduct research. These

networks have at their disposal some methodologies and tools to assist in the VO creation process.

To properly understand VBE supporting systems, it is necessary to identify the different components and entities, what are the relations between each other, and the responsibilities they have and require. The number of roles and actors can vary for each VBE and its context. A lot of these roles can be performed by the same entity or be assigned to multiple entities (Afsarmanesh et al., 2008; Romero et al., 2008). Table 2.2 shows some of these roles with a brief description.

Table 2.2: Different types of roles inside a VBE

Role	Description
VBE Administrator	The VBE administrator role can be carried out by a person or an organization that oversees the VBE's operation, the promotion of cooperation and/or collaboration among its members, management of its general processes, and the proper management of the profiles and competencies of its members, the registration of members and their information into standard profiles
VBE Member	The role is meant to represent any organization inside the VBE and partake in the process of providing its competencies and skills as members of a VO.
Opportunity Broker	The role is given to a VBE member who is responsible for trying to find new business opportunities and possibly interacting with customers.
VO planner	The member of the VBE who finds himself a business opportunity with the help provided by the broker is tasked with the responsibility of creating a VO, finding partners, assigning the roles inside the VO along managing its lifecycle.

A modeled detailed inspection on how the VBE management can be made and broken down into smaller individual processes, their relations/dependencies with the different roles, and what sort of services are needed to be covered by the VBE, identified that the management system of a VBE needs to cover the following functionalities, as briefly described in Table 2.3 (Oliveira, 2016; Rabelo et al., 2006; Romero et al., 2008).

Table 2.3: List of different VBE functionalities.

Functionality	Description
Member Admission	The service of reception and integration of a contender for VBE membership is required to send information to build a profile to properly register, this information can vary depending on the VBE
Member Withdrawal	Can be triggered by either request of a VBE member or VBE administrator the information and profile data of the past member can be saved depending on the rules and agreement terms of the VBE
Management of Profiles	The continuous update of information on the profiles of the VBE members and the search for specific information regarding a group of members and accessing their information profile
Consulting Services	The process of displaying information for possible customers on what the available activities the VBE can provide based on its member composition

2.1.2 Virtual Organizations

In much of the literature focused on VOs there have been several different frameworks of research to develop and bring new ways to enable the success of VOs. A common trend in the study of VOs tends to divide its lifecycle into different stages that could be summarized as in Figure 2.4.

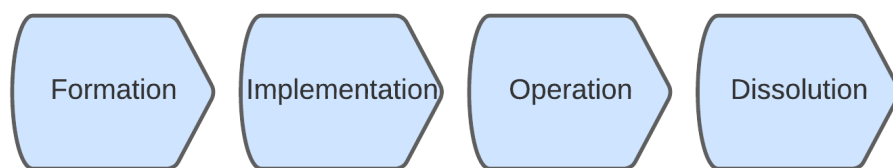


Figure 2.4: Stages of VO lifecycle.

The formation stage is the process of setting up, recruiting, and organizing members after finding a business opportunity. It consists of a process of finding potential members, afterwards there is the need to establish a governance mode and decision-making processes, as well as the establishment of common communication tools to work.

The implementation stage is characterized by defining roles and responsibilities for each member, by assigning specific tasks or outlining the areas of expertise that each participant is expected to contribute, defining deadlines, negotiating and the management of resources is

also part of this stage as making sure there is a way to keep track and evaluate the processes of each member.

The operation stage is focused on keeping the VO functioning effectively and efficiently, naturally following the constant monitoring of resources and their allocation as time goes on, and ensuring the communication channels are maintained to better respond to changes as the VO evolves and changes its structure and partners or challenges that may arise to reach goals or changes of strategies.

Finally, there is the dissolution phase in which the focus is usually on archiving and preserving records, reporting feedback, and disbursing assets among members and stakeholders. In this stage, there is a choice of the VO either dissolving itself and ending the collaboration network or being reorganized and repurposed for any other goal. Some authors merge the formation and implementation phases including all the necessary tasks in a creation phase (Afsarmanesh et al., 2009).

The formation and implementation or creation stage can be further divided into four major steps, being: Business opportunity finding, VO planning, Partner search and selection, and Negotiation. Usual literature on different frameworks and meta-models for VO creation takes into account the fact that VOs are being created in the context of a VBE or the pre-existence of another long-term association that can provide some tools and information for an efficient VO creation. A more general case of this process can be visualized in Figure 2.5.

With this, it can be assumed that a VO can only start when there is a business opportunity; and a business opportunity can arise in two different ways, either a customer makes an order that sparks the VO, or the participants compete to acquire one with a call of tenders examples for can be seen in the ECOLEAD project that provides methods and tools to quickly find new call for tenders (Camarinha-Matos et al., 2008; Polyantchikov et al., 2017).

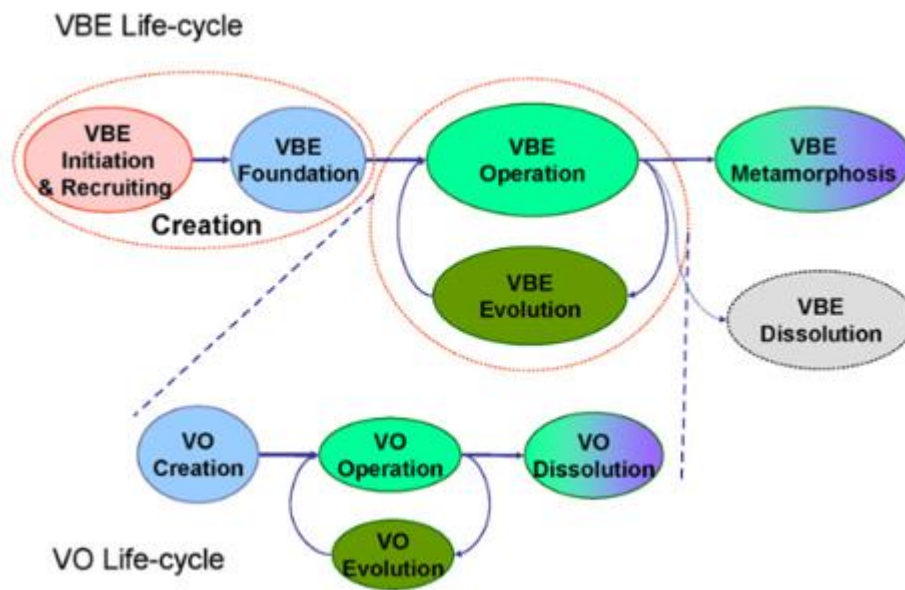


Figure 2.5: VO creation in a VBE context. From (Camarinha-Matos & Afsarmanesh 2007).

The task of VO planning is about finding the competencies that are needed, which conditions they need, what will be the organizational structure of the VO and the different roles each member will have as well as the contracting. (Camarinha-Matos et al., 2005), provides a case-by-case scenario on how the partnership contracts can be made. According to authors, the most common types of contract agreements are with either an "Explicit consortium" where the client knows all the members of the VO that will participate, or a "Internal consortium" where the client only establishes a contract or agreement with an organizer, usually the VO planner, who then ensures the use of internal contracts with the rest of the VO members, so the client doesn't necessarily need to care about who is involved.

During the partner selection, it is worth mentioning that there is also a vast field of different angles to find solutions that help VO creation or its lifecycle. Some of these can be based on multi-agent architectures to assist and better scale operations. Projects such as in (Arrais-Castro et al., 2018; ben Salah et al., 2019) use these multi-agent systems within the process of partner selection and identification with a multi-criteria decision model and their associated values over a temporal framework of past, present and future in the form of matrices, also data fusion methods and fuzzy logic can be used.

Works such as (Vaez-Alaei et al., 2022) conclude from their research that the most used criteria for partner selection are the similarity in objectives, expertise, cultural similarity, and the successful history of their past collaboration projects. These were then identified as key

factors for increasing trust among members and the exchanges and knowledge sharing. Another factor that influenced the criteria of partner selection in this study, is the complexity of the project as it shows to be one of the main reasons that leads to collaborative failure and how the different knowledge between partners can have a relation of complementarity or coverage. With this, they create a model that takes into consideration how trust changes based on history, and the complexity the type of knowledge regarding the relation they need to maintain.

(ben Salah et al., 2019), uses a two-stage operation to make a partner and role selection for VO creation, consisting of first selecting the criteria that through an, Analytical Network Process fuzzy technique are weighted based on the requirements, while in the second stage the weights are then used to develop a multi-objective programming model configured to get the best partner allocation order. This model has the advantage of being able to compare the different combinations and trade-offs between the different criteria, with a case test in the textile Industry.

(Shadi & Afsarmanesh, 2014) works on a tool called VOSAT that through an agent-based view of the different partners on a VO can monitor if the daily promises made by its members are fulfilled or not thus being able to control the level of trustiness of its agents. This tool is also used to help a VO coordinator to better replace partners during the VO operation while also calculating the risk of such replacement in the context of the tasks that the VO tends to pursue.

Other works such as (Hsieh, 2019) are more focused on certain types of VOs, namely negotiation for supply chains, where the focus is on optimization of the management of the workflows using an agent-based approach and contracts. The main objective is to try to achieve efficiency in the time cost and the demand of the supply chain, while strengthening the capability of the VO to handle internal disruptions. The system is prototyped using petri nets and agent platforms using FIPA specifications.

Negotiation between entities refers to the process by which two or more enterprises come to an agreement on the terms and conditions of a business relationship. The end goal of the negotiation process is to have a concise agreement that takes the form of a contract to protect in case a member doesn't fulfill their part of the deal. Previous work on the subject and other projects suggests that although there are tools that can be developed to help the creation of contracts and facilitate negotiation, most details and deals still require representatives to meet and discuss the details in person or through video conference before agreeing.

An early example can be seen in (Camarinha-Matos et al., 2009), where a platform called agreement negotiation wizard (WizAN) is developed to work through the inputs of other tools used during the VO creation process, such as using the tools to find collaboration opportunities, partners, and detailing the different processes for them as the input for the customer identity, logical structure and the processes that each member needs to attend to. The outcome of this tool is an agreement divided into several sections for the different details of the negotiation. This tool provides: a contract editor, so members can monitor the details and signatures of the final agreement; a virtual rooms so they can meet in private; an assisted contract creation system, that provides templates of contracts; and finally an electronic notary (e-Notary) that allows all the involved parties to prove their authenticity and provide a repository for the signed contracts.

In comparison to a more recent project by (Camarinha-Matos et al., 2015), there is also a tool developed to assist in the negotiation phase of the VO creation, making use of both the tools used for planning and partner selection, using a contract editor to monitor the process of agreements and responsibilities between each actor, virtual negotiation rooms for meetings and a collection of negotiation templates to apply for each negotiation topic. This tool adds some other different services, such as a risk assessment tool which through information provided by the VBE management can give a numerical level of trustworthiness of each pending member for a specific role. Finally, the platform also has an e-Notary, so clients and collaborators can make a digital signature on the agreement documents while also providing an archive for them.

2.2 Electronic Notary

From analyzing the previous examples in Section 2.1.2, the usage of electronic notaries (e-notaries) was ever present and necessary for the VO creation process. Electronic notaries are individuals or software managed by an organization that is authorized to witness and validate the authenticity of digital documents. They play a crucial role in the VO creation process, where many business agreements take place online and often involve the signing of contracts and other legal documents. One of the main ways in which e-notaries help to increase trust in the VO is using of digital signatures. A digital signature is a type of electronic signature that uses encryption to verify the authenticity of a document. When an electronic document is signed with a digital signature, it acts as a seal of authenticity, much like a traditional notarized

signature on a paper document. The use of digital signatures in conjunction with electronic notaries helps to ensure the integrity and authenticity of electronic documents within a VO. It allows enterprises to prove that they are who they say they are and that the documents they are signing are genuine. This is particularly important in the VO, where there is often a lack of face-to-face interaction, and it can be more challenging to verify the identity of the parties involved in a transaction. They help to ensure that business transactions are carried out smoothly and efficiently and that all parties involved can have confidence in the authenticity of the documents being signed, allowing a way to easily prove if a member does not comply with the agreed terms of the VO.

2.2.1 Digital Signatures

Although the purpose of digital signatures in this context isn't to legally bind enterprises to the agreements they conduct, but merely to improve the authenticity of their requests and agreements, it is important to note that these digital signatures aren't mixed with any legally binding ones. This characteristic is often referred to as "non-repudiation" which requires a legal status with the local governments to be granted.

The usage of digital signatures started with the development of the RSA algorithm (Rivest et al., 1978), and later on many other algorithms were developed with some being more popular for certain scenarios than others, such as DSA, RSA, SHA (Ferrag et al., 2018), some more specific algorithms such as blind RSA allow the protection of the signer identity, thus being useful for digital voting or online purchases anonymously (Pointcheval & Stern, 2000) or the case of ECDSA that provides increased security over DSA but also is particularly popular with cryptocurrency technologies (Hartwig, 2016; Johnson et al., 2001).

What these algorithms introduced was the concept of public-private key cryptography, or asymmetric key which enables the usage of two different keys to send messages and read them. Usually when sending messages, a mixed combination of session keys to provide protection for the messages and asymmetric keys to establish that link and make the signatures. The process of digital signatures can be exemplified in Figure 2.6.

Each member has assigned himself two different keys: a public one that is known to all other members; and a private key that is only known and used by the respective owner. Whenever a member wants to create his digital signature to sign a document, he uses one of the encryption algorithms explained above that have been standardized and pre-established for use for the specific e-notary since different algorithms require different sizes and key characteristics. The procedure can be summarized as follows:

- The document or message is hashed: The document or message being signed is processed through a hash function, which generates a fixed-size string of characters (the hash) that represents the contents of the document. The hash is unique to the document, meaning that even a small change to the document will result in a completely different hash.
- The hash is encrypted using the signer's private key: The signer's private key is a secret code that is used to encrypt the hash. Encrypting the hash with the private key produces the digital signature.
- The digital signature is attached to the document: The digital signature and the document are sent to the recipient together.
- The recipient verifies the signature using the signer's public key: The recipient uses the signer's public key, which is publicly available, to decrypt the digital signature and compare it to a new hash of the document. If the decrypted signature matches the new hash, the document is authenticated, and the recipient can be confident that it has not been altered and that it was indeed signed by the claimed signer.

Since the usage of electronic notaries isn't about legally binding the organization into the agreements, the private and public keys used can be assigned to the VBE members freely when they register along with their profile.

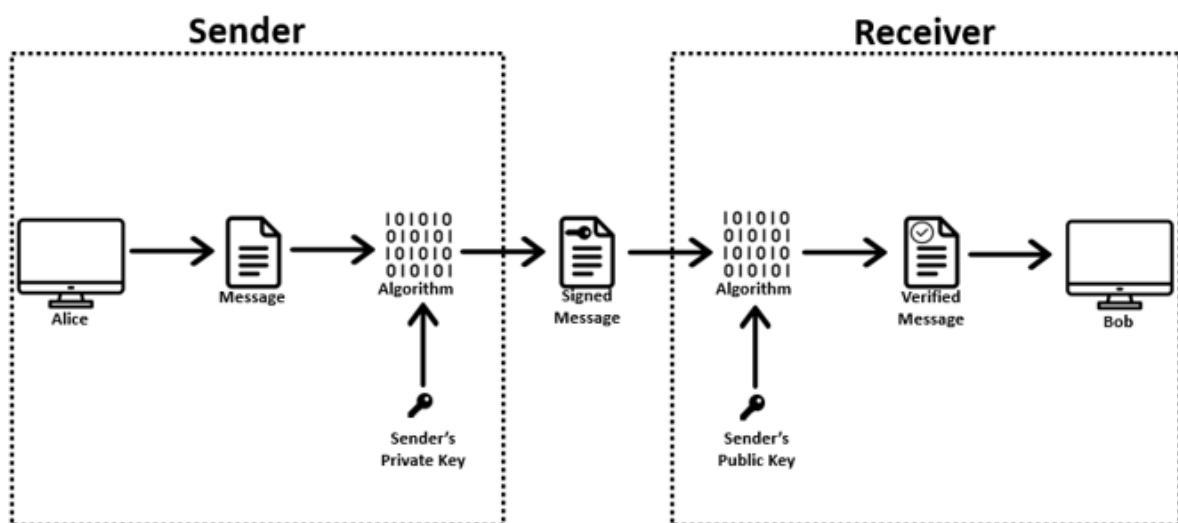


Figure 2.6: Process of digital signature creation and verification.

2.2.2 Notary Management

The e-notaries need to offer a whole set of different functionalities aside from digital signatures to do their intended work, these functionalities and services can largely depend on their intended purpose and working context.

Examples such as those provided by (Ruotsalainen & Manning, 2007) show that a federated approach could be made, where a central e-notary communicates with several distributed systems that send information about a registered member. The central e-notary system then supports a variety of different types of data objects as long as they have the required format and information. This e-notary system provides each user with an associated personal folder for all his archived information, being composed of the different collected data objects. The sent data objects all have a digital signature as well as a security parameter that specify who can then have access to it. It also includes another signature and timestamp provided by the central archive to be able to provide a secure audit-log system.

In the context of collaborative networks, there have been two projects where e-notaries were implemented. Namely in the ECOLEAD project (Oliveira & Camarinha-Matos, 2008) where the e-notary is used in the VO creation process, providing its users the capacity to register separately from the VBE environment and get their private keys for digital signatures. The e-notary has the functionalities of storing agreement documents, and validating documents with digital signatures to prove the authenticity of the signers as well as the option to read the documents they are allowed to access. In (Oliveira, 2016) an e-notary was developed for the GoNet project, including the notion of a "Negotiation Dossier" that acts as a folder for storing all the negotiation documents of a particular VO negotiation process, the member that can manage this specific folder is usually the VO planner, aside from this particular aspect most of the functionalities are the same as the ones in the previous project (with a register function so each member can have his private key for document signing, a database for all the stored users and their permissions on the documents and dossiers that they can access). A more complete overview of the functionalities can be depicted in Figure 2.7.

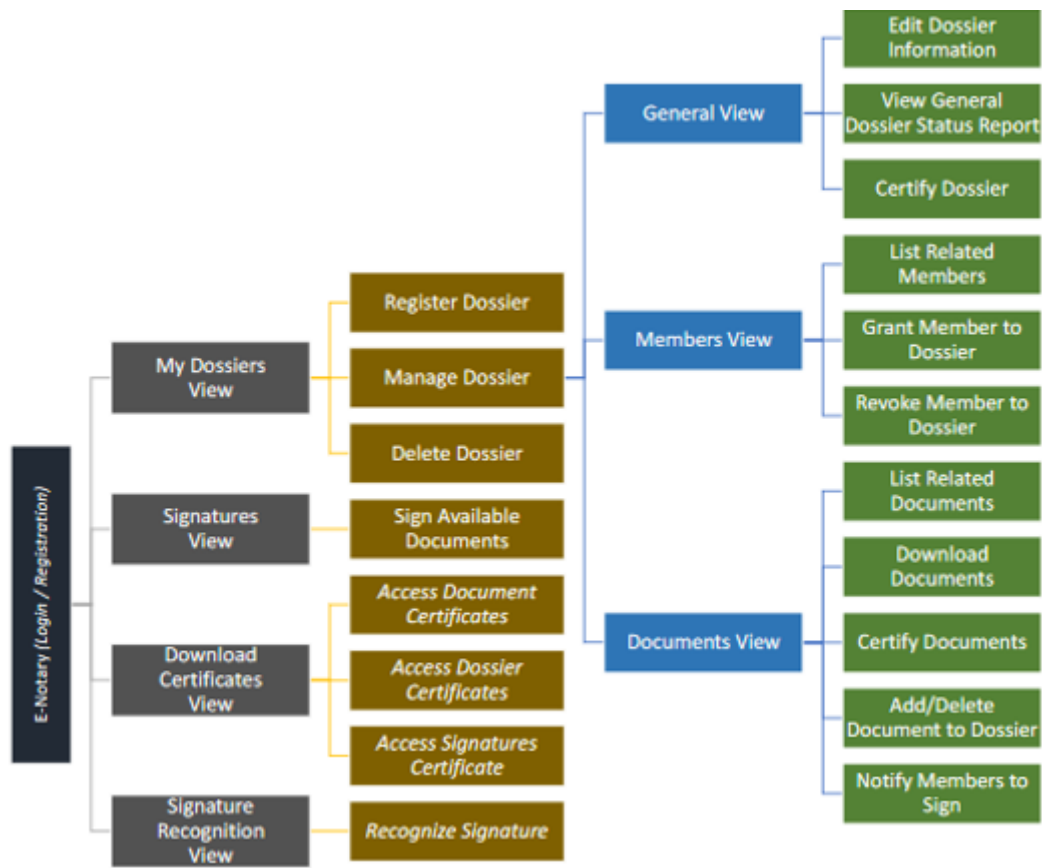


Figure 2.7: Overview of notary functions. From (Oliveira, 2016).

2.3 Blockchain Technology

Blockchain is a network technology that became popularized in 2008 as a new way to make payments and money transactions without the need for a trusted third-party institution and the core concept behind cryptocurrency, namely popularized by Bitcoin (Nakamoto, 2008). The main feature of this technology is to have a distributed, non-centralized database that provides a secure and transparent way to transfer digital assets without the need for central authority.

The technology evolved with the passing years and depending on the context and literature there is an understanding of three major different "generations" of blockchain technology. Namely what is referred to as the standard basic blockchain that is present in Bitcoin is labeled as "Blockchain 1.0" (Aggarwal & Kumar, 2021). Blockchain 2.0 saw the introduction of "smart contracts" with platforms such as Ethereum which are composed of self-executing contracts with the terms of agreement between buyer and seller being written into direct lines of code, expanding the capabilities beyond simple transactions, and enabling the

development of more decentralized applications (S.-G. Kim, 2018). Blockchain 3.0 is about focusing less on cryptocurrency usage and more on applications in other industries and different paradigms, such as supply chains, manufacturing, the internet of things, healthcare, digital voting, and identity management systems. In short, it is about the change of focus into integrating different technologies with it (di Francesco Maesa & Mori, 2020).

2.3.1 Blockchain General Structure

To understand how these networks evolved and the later applications that came to dominate recent literature on the subject it is necessary to understand the core concepts of blockchain. One of the main problems for which blockchain was designed to solve, was the issue with double-spending which is a flaw related to digital cash protocols. Where a coin is spent more than once (Chohan, 2018). To tackle this problem, blockchain was created as a peer-to-peer network where transactions are stored into blocks, these blocks have a specific architecture and content that can vary a bit depending on each design, but many use the following structure:

- Header fields
 - Block number, which is a sequential number assigned to each block, the genesis block is referred to as the first block in a chain and assigned the number 0, each next block has a greater value than the previous. Used to specify and locate specific blocks.
 - The previous block header's hash works as a unique fingerprint that is calculated using the information of the previous block and helps to ensure the integrity and security of the blockchain, every block contains this hash except for the genesis block.
 - Merkle tree root of the data stored in the block, is composed of the hash of the data in the block which allows an easy verification of the data without the need to examine each individual set of data. A more detailed visual example can be found in Figure 2.8.
 - The size of the block, measured in bytes is useful for limiting the amount of data included in the block and can be used for determining fees the users have to pay for their transactions and monitoring the size of the blockchain.
 - A timestamp, usually in Unix time format, that is the number of seconds elapsed since January 1st, 1970. It helps provide a chronological order for each transaction and ensures the integrity of the chain.

- Nonce value, an arbitrary number that is present in blockchains that use proof-of-work consensus algorithms which will be discussed in a later section.
- Data
 - List of transactions or events recorded.
 - Any additional information.

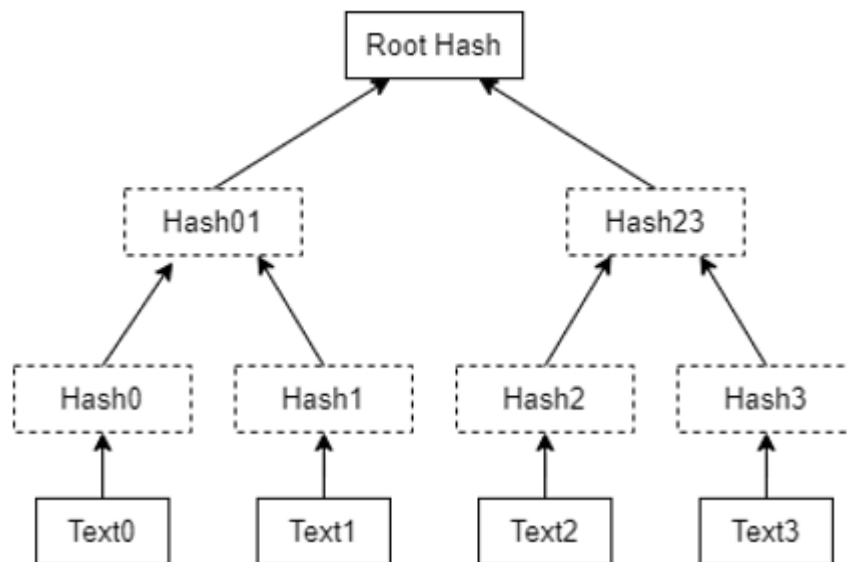


Figure 2.8: Example of Merkle tree root calculation.

With this information about the general structure of each block, it is possible to see how the chaining effect of the blockchain works, with each block containing the hash of the previous block. So, if a previous block change it would result in a different hash and thus make a chain effect on all following blocks, making it possible to detect any change in the values. Figure 2.9 shows a general view of a chain of blocks.

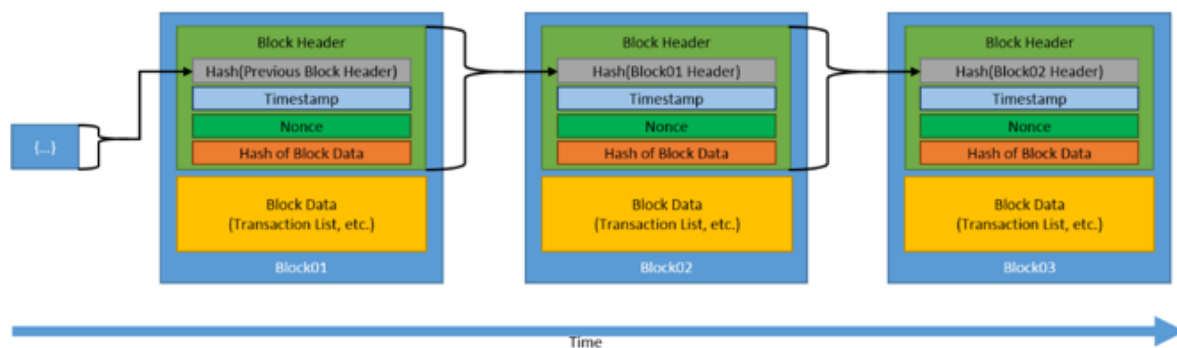


Figure 2.9: Representation of how blocks interlink. From (Yaga et al., 2018).

To have an even more general overview of how the whole system works, when the user makes a transaction using an application such as a virtual wallet or API he sends a broadcast to the network of nodes. These nodes then run consensus algorithms, also known as mining, to select which one publishes the next block, and once the transaction is done a new block is added. This new block addition is also broadcasted to the other nodes to verify the block. As a reward for publishing the new block, the node is rewarded depending on the consensus algorithm.

It is also worth noting that there are two types of nodes in blockchain networks: the so-called lightweight nodes and the regular nodes (Antonopoulou, 2014). Lightweight nodes do not store the entirety of the blockchain and are typically used in mobile wallets, to be useful for their users in storing only parts of the blockchain that are relevant for the transactions being performed. These nodes don't have much computational power and cannot perform the functions as normal nodes do and get rewards. Regular nodes are computers or servers that run the necessary software to perform the node functions, they store the entirety of the blockchain and are responsible for validating transactions and propagating them to other nodes.

2.3.2 Consensus Models

As it has been noted in the previous sections, to know which node is responsible for publishing a block the network uses a consensus algorithm, also called the consensus model. These are used for solving any other conflicts when two nodes try to publish the same block or the reason why a node would not try to prevent another block from being published for his benefit (Yaga et al., 2018).

According to (Yaga et al., 2018), for a potential node to join the network, they must agree on the initial state of the aforementioned network that is inscribed in the genesis block, and they can independently validate every single block and come to an agreement on the state of the blockchain. However, if two valid chains are presented to a node, it discards the shortest by default.

Following this, the next sub-sections briefly describes the most common consensus models used in blockchains according to (Baliga, 2017; Bamakan et al., 2020; Yaga et al., 2018).

2.3.2.1 Proof-of-Work Consensus

Proof-of-Work (PoW) consensus model works by allowing the first node to solve a computationally intensive problem to publish the next block. The solution to this mathematical problem can be easily confirmed after achieving its solution. The process is what is often referred to as mining.

The problem or puzzle revolves around finding a random number, the previously mentioned nonce, that once inserted with the hash of the previous block and the current block header makes a final hash that fulfills certain criteria, such as having a certain number of zeroes. The process is illustrated in Figure 2.10.

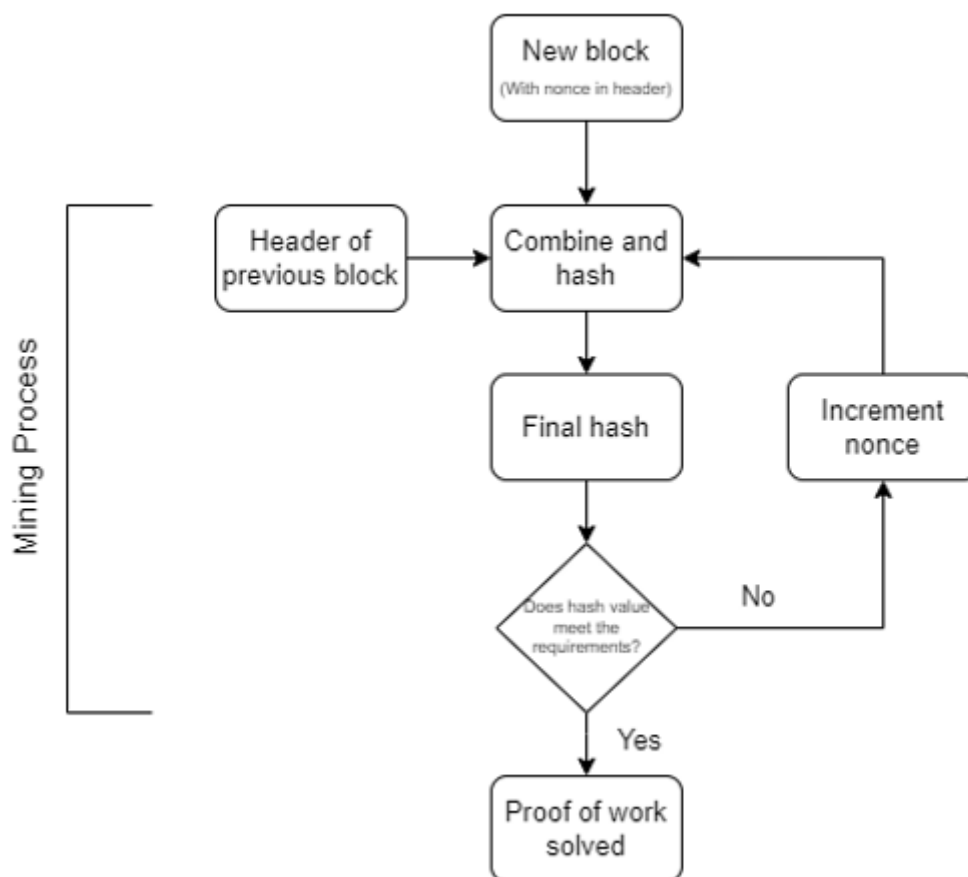


Figure 2.10: Flow chart of the proof-of-work algorithm.

This problem or challenge can then be tweaked and increased the difficulty of the challenge to control the security and the rate at which blocks are published. This is done by altering the number of needed zeros. An increase in the required number of zeros for the final hash will result in a more difficult problem and more computational resource consumption. By

the nature of this challenge, the amount of work put into solving it doesn't necessarily mean one is the first to find the solution nor does it have any implication for future challenges.

After a node finds the correct nonce number it publishes the block to all other nodes, that verify and validate the block and start immediately working on trying to publish the next block as the other nodes are also doing it. The publishing node is rewarded by the system in a financial incentive to keep the network working.

In some cases, nodes can aggregate themselves into groups that try to divide the workload, either equally or unequally, among themselves to work on finding the correct nonce number, in the case they correctly guess it and publish the block, the incentive is divided proportionally among the members and their respective work assigned. Since this method uses computationally intensive problems to post a block, any attacks that try to emulate more nodes can't influence or manipulate the network. However, if an entity manages to gather 51% or more of the total computational power (hash rate) in the network it can disrupt the network by manipulating blocks or withholding blocks. This problem isn't a direct threat to many of the public blockchain networks since they have enough participants and decentralization to make any attempt at this attack unprofitable.

2.3.2.2 Proof-of-Stake Consensus

The philosophy behind the proof-of-stake consensus is that the more invested a node in the network, the more motivated it is to help it and not disrupt. This works by having the participating node invest a certain amount of its currency into a special wallet that no longer allows it to be spent. And this amount of currency is used as the determinant for how likely the node is to be selected, among the other nodes, to publish a new block. With these types of models instead of new currency being "created" each time a block is created as a reward, the publisher simply collects a transaction fee paid by a user that makes a transaction.

There are various methods to select the node that publishes the block and how his stake can influence it. If it uses a random election, then the network will attribute a certain probability for each node based on the percentage of currency, that each node holds within the network.

This method has the advantage of not requiring the nodes to do heavy computing processes and wasting too much energy when using the network.

2.3.2.3 Proof-of-Authority Consensus

In this consensus model, the authority to publish and write new blocks lies on specific pre-selected nodes that are operated by trusted parties. These nodes must have their identity available to be verified in the blockchain using the stored signatures, being therefore easily validated. These types of models can use "reputation" scores that can increase or decrease by ratings given by its users. This score influences the capacity a node has to publish the next block. These types of models are only used on certain types of blockchains that are private and permissioned and presuppose a pre-established level of trust among the nodes. These types of blockchain networks will be further analyzed.

2.3.2.4 Byzantine Fault Tolerance

Inside these models there are various sub-types. However, the one that is more used and relevant for this work is the Partial Byzantine Fault Tolerance algorithm. This method in particular uses a voting system to achieve a consensus on the block that will be published. The minimal number of votes needed for a certain block to be published cannot be less than two-thirds of the nodes present in the network.

Nodes here are divided by having a primary node and secondary nodes. The primary one is elected first at random and then the nodes perform periodic elections. Nevertheless, if the primary node hasn't made any broadcast requests in a certain period, the other nodes elect the one with the highest legitimacy.

In this method, the user sends his request to the primary node. This node broadcasts the request to all other nodes in the network, these nodes then process the request. Each node sends an individual reply to the user, and if the user receives at least two-thirds of the requests he has the confirmation that there are not enough faulty nodes trying to disrupt the request and the process is concluded.

2.3.3 Types of Blockchains

According to literature, there are various ways to categorize the different types of blockchain networks. One simple way these can be divided is between public and private.

Public blockchains, also referred to as permissionless blockchains (Lee Kuo Chuen & Deng, 2017), are defined as being open for anyone to partake in them without requiring to provide any identification as they can enter and leave as nodes and operate anonymously. Many of the usages of these types of blockchains use Proof-of-Work and Proof-of-Stake

consensus algorithms and by their nature, the more participants and nodes the more secure and trusted it becomes. As previously mentioned in, these models, the more nodes exist, the harder is to disrupt the network (Bernal Bernabe et al., 2019). Most of these networks are generally used only for the purpose of cryptocurrencies, with the main examples being Bitcoin and Ethereum.

Private blockchains can be further divided between permissioned or permissionless (Politou et al., 2021; Yang et al., 2020). Private blockchains are networks where a select number of users are allowed to partake in, and information is validated by trusted nodes. Instead of anonymously, they tend to use different consensus algorithms than public ones, such as byzantine fault tolerance or proof-of-authority. Since there is a general trust among the nodes, proof-of-work is inefficient as it wastes energy and makes a long competition when it shouldn't be the focus of the private blockchain. The focus of private blockchains is their usage on businesses with a decentralized network.

Permissioned private blockchains are the standard use for most businesses that want to implement their own decentralized network. Since any user, or node, that wants to join needs to be authorized by a central authority or group, they are the most used for internal use cases inside organizations. Permissionless private blockchains can serve as a hybrid between the two and have their own usage (Helliard et al., 2020). These networks have a pre-selected number of nodes that are trusted and allow anyone to join if they meet a certain pre-defined criterion, allowing users to join in a read-only authorization. Permissionless private blockchains can be used for some inter-organizational collaboration such as supply-chain management. It is important to note that not all literature recognizes the existence of permissionless private blockchains and usually refers to public blockchains as the ones that are always permissionless.

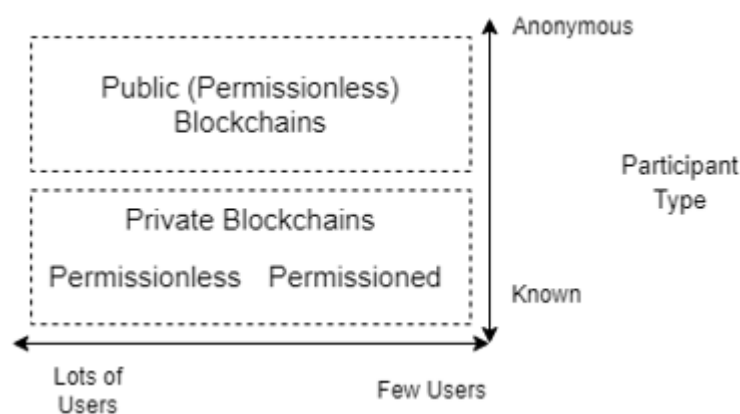


Figure 2.11: Different types of blockchain networks

Overall, the different types of blockchains and their different properties illustrated in Figure 2.11.

Some other literature tends to categorize differently of the properties a blockchain network, and how a network needs to balance or prioritize some of these properties in exchange for others. Most famously, there is the "Blockchain Trilemma" coined by Vitalik Buterin (Hafid et al., 2020; Steiu, 2020; Zhou et al., 2020), where he describes the three most important properties of Blockchain as being its decentralized nature, the security it offers, and the scalability as a real problem. The trilemma's problem is that it is only possible to focus on two of these properties while leaving the other. The trilemma can be exemplified in Figure 2.12.

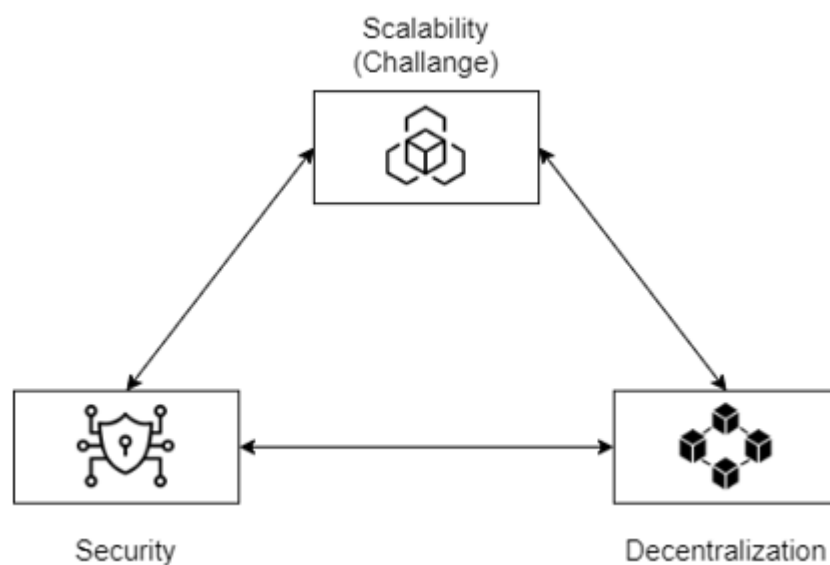


Figure 2.12: Representation of the blockchain trilemma.

From the scalability point of view, we can only choose between more decentralization as a trade-in for having less security or vice versa, this happens for example as more data is added to the network and more flows of information it becomes harder to move the data around and keep it updated. The possible solutions for this would be to either limit the network and only let more powerful nodes partake in which would make the network less secure from 51% attacks, or only allow more trusted authorities to partake in it which would remove the decentralization as an example.

2.3.4 Blockchain Frameworks

Many of the famous crypto-currencies and blockchain networks were created using frameworks like Ether being created in the Ethereum platform. These frameworks are a set of

protocols and rules and provide the infrastructure for building a blockchain network or a blockchain-based application. They already include some consensus models, privacy and security features, and the usage of smart contracts.

Some of the more famously known platforms are Ethereum, Hyperledger Fabric, and R3 Corda. Hyperledger Fabric is one of the more famously known since it was developed by the Linux Foundation and is Open-Source which allow for a very modular creation of blockchain networks without any clear standard. Moreover, it features its own type of smart contracts known, as Chaincode, that allows for a better exchange between applications and the blockchain ledger. A detailed description of the more famous frameworks can be found in Table 2.4.

Table 2.4: Table comparing different frameworks.

Framework	Industry Focus	Consensus	Open Source	Cryptocurrency
Hyperledger Fabric	Cross-Industry	Voting-based or pluggable consensus	Yes	No
Ethereum	Cross-Industry	Proof-of-work or proof-of-authority	Yes	Yes
Quorum	Financial-Industry	Byzantine Fault Tolerance or Proof-of-authority	Yes	No
R3 Corda	Financial-Industry	Voting-based	Yes	No

2.3.5 Blockchain Applications

The decentralized and secure nature of blockchain technology makes it well-suited for a wide range of use cases, and many organizations and individuals are exploring ways to leverage it for their specific needs. In this section some of the most promising blockchain implementations and applications that are currently in development or in use, that are relevant, to some extent, to the first research question of this thesis, are further expanded.

In (Hyla & Pejaś, 2020), it was developed a private permissioned blockchain network for the purpose of being able to validate digital signatures on documents without the need for a third party to perform maintenance and timestamping. The issue arises with the problem that,

user public keys used to perform these signatures, are only valid for a relatively short time of two years. So, a trusted third party needs to create a hash of the digitally signed document and the current time and restamp it periodically.

To solve this, the notary takes all the signed documents are inserted into a single block which generates a hash based on the time and also generates a special object from the reduced hash tree that is given to every single user that took place in a document signing. So, if the user wants to verify the validity of a signed document, he uses the object as input to a function that researches the blockchain network and returns a true/false response. This works by not only using the time in the created hash but also because every single block in the network is created within a defined period with the same size independently of the input. And thus, this helps increase the scalability of the network and verify the time of the signature. Finally, the proposed scheme is very adaptable and can be integrated into already existing public blockchains as well, making it easy to change the consensus model.

Another use case proposed and studied is using blockchain networks to create a voting system (Hjalmarsson et al., 2018). In this case, it is developed a permissioned blockchain with a proof-of-authority consensus where despite people still needing to access specialized voting booths, the objective is to make a secure and way to vote without the risk of fraud and reducing the costs and time of any election. The network is divided into two types of nodes: the "district nodes" that are responsible for serving their assigned district; and the "bootnodes" that serve only coordination. Bootnodes initiate an election by deploying a smart contract to each district node, seeing the smart contract as the ballot for each respective voter. After the votes are cast the district nodes will be validated by all nodes on the network and added to the blockchain. The smart contracts will be prepared to only let a predetermined set of people vote and so each individual voter has an identity assigned to them to partake in the voting process as well as a transaction ID that can be used after voting to verify their casted vote.

In the health sector, there has also been a large amount of research and development of blockchain applications that help store personal health records securely while also maintaining a dynamic set of access rules and file management, as proposed in the framework of (Shahnaz et al., 2019). Here a special hash is used to secure the health records that are stored in the blockchain network, and through the usage of smart contracts roles and access options can be achieved, enabling a patient to allow medical staff only to access and edit certain records and use create, read, update and delete operations with the usage of functions to add and remove roles to other people. The latter performs an evaluation by measuring response times or any

breaches in security by trying to detect document changes in the hash codes. Figure 2.13 shows a basic illustration of the application.

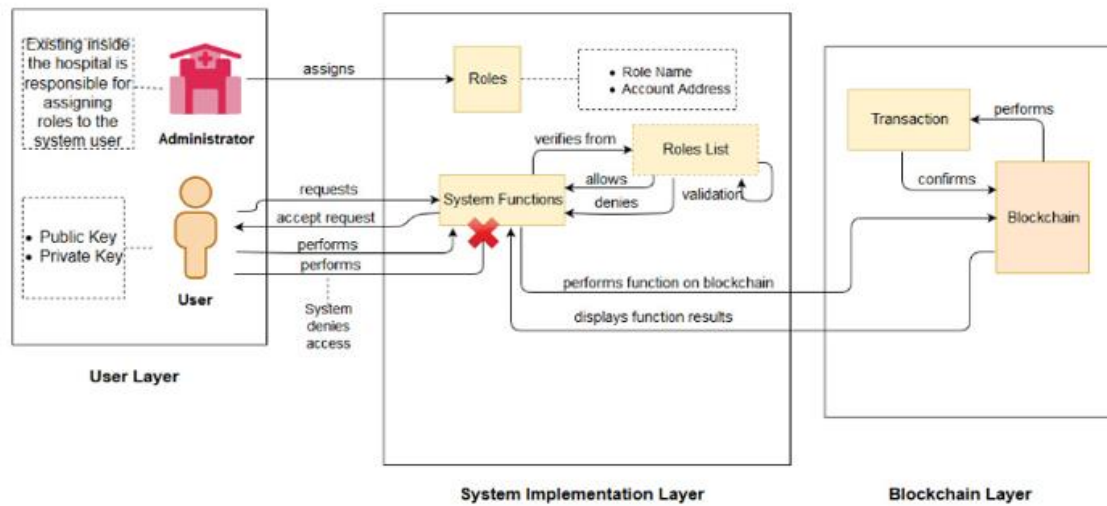


Figure 2.13: Overview of the healthcare process. From (Shahnaz et al., 2019).

Finally, there is also the usage of blockchain networks for the context of the internet-of-things (IoT) with implementations such as those shown in (Novo, 2018). Here the main challenge lies in the fact that IoT devices can't handle having the whole blockchain stored inside their own hardware. To solve, this IoT devices first communicate their requests to a special interface hub that is connected to an actual blockchain network node to process the information. The request actions and access would be defined in the smart contracts on the blockchain that are triggered for every transaction. The only entities with the ability to change the content and permissions

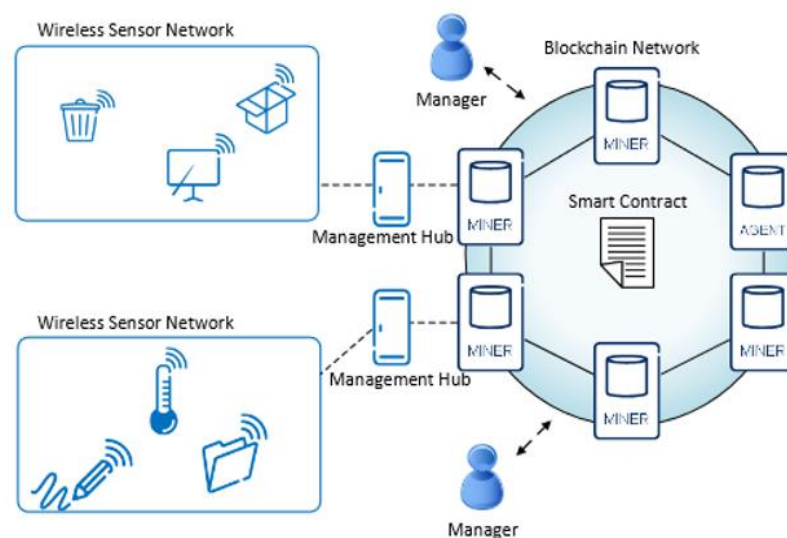


Figure 2.14: Representation of how IoT devices interact with the blockchain. From (Novo, 2018).

of the contract would be the managers that register themselves and also the IoT devices. The representation of the network can be seen in Figure 2.14.

2.4 Relation to Current Work

In summary, this chapter explained the three different technologies that are involved in the development of an e-notary blockchain service for collaborative networks. The context of collaborative networks raises the problem of enabling trust among organizations and people while providing a conceptual field that helps integrate ICT tools to create collaboration ecosystems with a variety of different characteristics. Starting from the concept of the VBE that creates a large community of organizations or people as members. These collaborative networks aim to set standard ICT tools for their members to share resources and find partners when a business collaboration arises which results in the creation of a VO. One tool that VBEs offer to their members is an electronic notary where the agreements of any VO participants can reliably store cast votes. In related VBE projects, electronic notaries are provided by a central trusted third-party member that stores the information, manages permissions, and validates the identities of the members.

Blockchain technology offers a new form of storing and validating data in a decentralized way through the usage of chains of blocks and consensus algorithms. Furthermore, this grants it greater resistance to infrastructure problems and security over any malicious act of forging fake information inside its data storage. Depending on the consensus algorithm and accessibility of the blockchain network it is possible to create a private blockchain network where only VBE members have access to and interact by either becoming additional nodes in the data storage and validation or mere clients for its service. This blockchain network thus can function as an electronic notary with a smart contract deployed that has all the necessary back-end functionalities coded. A blockchain notary decentralized network can work as a substitute for regular electronic notaries in VBE communities as it bolsters more shock-proof, security and trust among the members.

SYSTEM DESIGN

In this chapter, presents the solution developed for creating an electronic notary for a VBE community using a blockchain network. It starts by explaining the context in which the service is applied. The chapter then follows by listing the application requirements either for the network or for its users. Finally, it presents all the steps taken to develop the aforementioned solution.

3.1 Context Identification

As previously mentioned, namely in the first chapter, this work aims at developing an electronic notary to support a VBE community. This e-notary also aims to leverage the level of trust in this community by assisting the members in a VO creation process. When a business opportunity arises, and a VO is created the VO planner will use this e-notary to store the hash of the agreement he has made with the rest of the VO members. When this agreement is stored in the e-notary the original uploader will be able to allow the other users to have access to it and vote on the proposed agreement and check how many of the allowed users have already cast their vote.

With this context in mind, it is expected that some necessary features for the usage of this e-notary service are already provided by the VBE community. Furthermore, the costs and required infrastructure should also be supported and maintained by this community, whether by the manager of the community or the community member via fees or some other hybrid approach. Some of these features are, for example, the tools for creating hash files of documents, necessary communication tools that allows its members to share information, and other tools that will be mentioned later in this chapter along the development section.

3.2 Design Requirements

This system is designed to meet the needs and context mentioned in the previous section 3.1. This section intends to clearly explain what the blockchain network and user requirements are, to understand how the system works and how it serves the user.

3.2.1 User Requirements

This section specifies the user requirements for the usage of the e-notary service. The list is as follows:

- Be a VBE member.
- Account manager that allows the creation of accounts specific to the blockchain framework that was chosen and authentication when using the application.
- A tool to create hash strings of files.
- A communication link with other VBE members so they can share their account IDs.
- Access to the cloud service
- If a VBE member wants to join as a node, it requires to have the same client framework as the blockchain network.

3.2.2 Network Requirements

Considering the previous chapters description of the requirements, it's important to select the necessary characteristics for the blockchain network to have. Proceeding on a private blockchain network, means that only allowed users can connect and participate, either working as nodes to validate and publish blocks, or mere clients that only interact with the network for the e-notary services. These users are naturally VBE members, it is intended to have means to only allow the VBE members to connect to this blockchain network and interact with it via an application interface. For this, an appropriate solution is to deploy the minimum functional requirements of the network (some nodes and the application interface) in a cloud service that is shared by the entire VBE community. An overview of how the solution is provided can be seen in Figure 3.1.

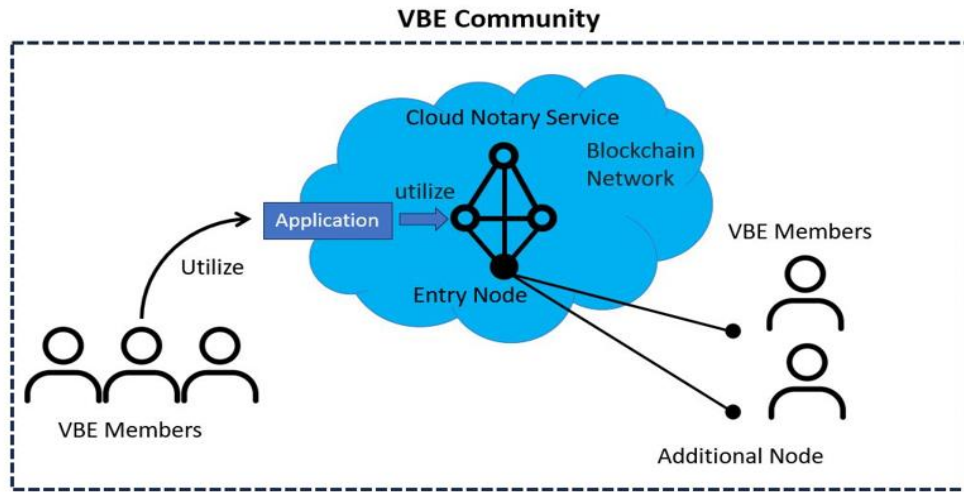


Figure 3.1: Overview of the proposed solution.

Considering this solution it is possible to subdivide network requirements into two further sections: the blockchain network requirements and the cloud service requirements.

The blockchain network functional requirements can be listed as follows:

- Proof-of-Authority Consensus algorithm.
- Only allowed members to join as nodes.
- Only allow new nodes after accepted by votes from previously allowed nodes.
- Have at least one node that can communicate with applications.
- Have a specific smart contract deployed in the network that will provide all the functionalities for e-notary service.
- Remove nodes from the network.
- Allow nodes to specify which users can accept requests.
- Metrics and means to verify which nodes validated and who published a certain block.

The cloud service functional requirements need to be the following:

- Firewall restrictions to allow the ports of the deployed nodes to communicate with outside users and between themselves.
- Role assignments and role management mechanisms to specify who can connect to the virtual machine.
- Specific region of the deployed cloud service.
- Controlled by the VBE administrator which grants VBE members access to the cloud.
- Interaction with the blockchain e-notary service preloaded with the deployed smart contract.

3.2.3 Use Cases

Use cases represent methodologies that aid in comprehending the system's workflow. Through the utilization of UML use cases, it becomes feasible to scrutinize and arrange the prerequisites alongside the functionalities within a system and discern how they ought to be executed. Within these use cases, the individuals who interact with the system are referred to as actors. The forthcoming system is anticipated to accommodate two distinct user categories: VO members and VO planners. The system's use cases from the viewpoint of the user are shown in Figure 3.2.

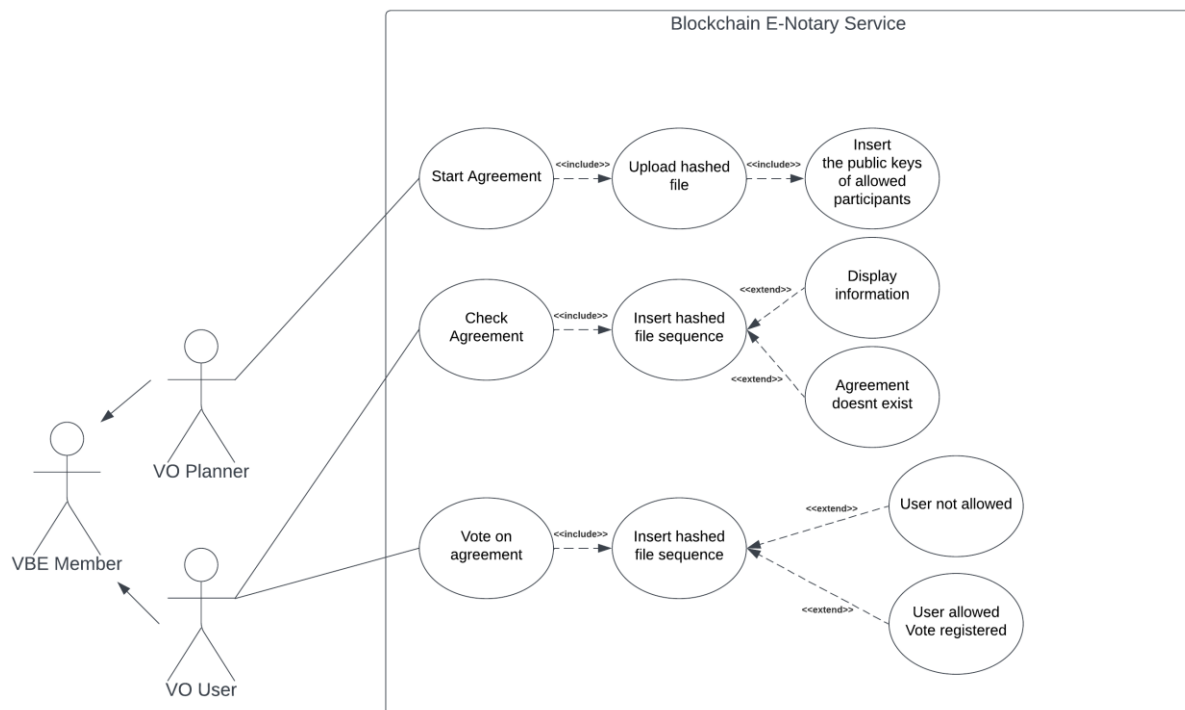


Figure 3.2: Use case diagram of the notary usage.

Moreover, sequence diagrams are graphical representations in software engineering that illustrate the interactions and order of messages exchanged between different components or objects within a system over a specific period. They are commonly used during the design and analysis phases of software development. A sequence diagram representing the main interactions of the users with the designed system is illustrated in Figure 3.3.

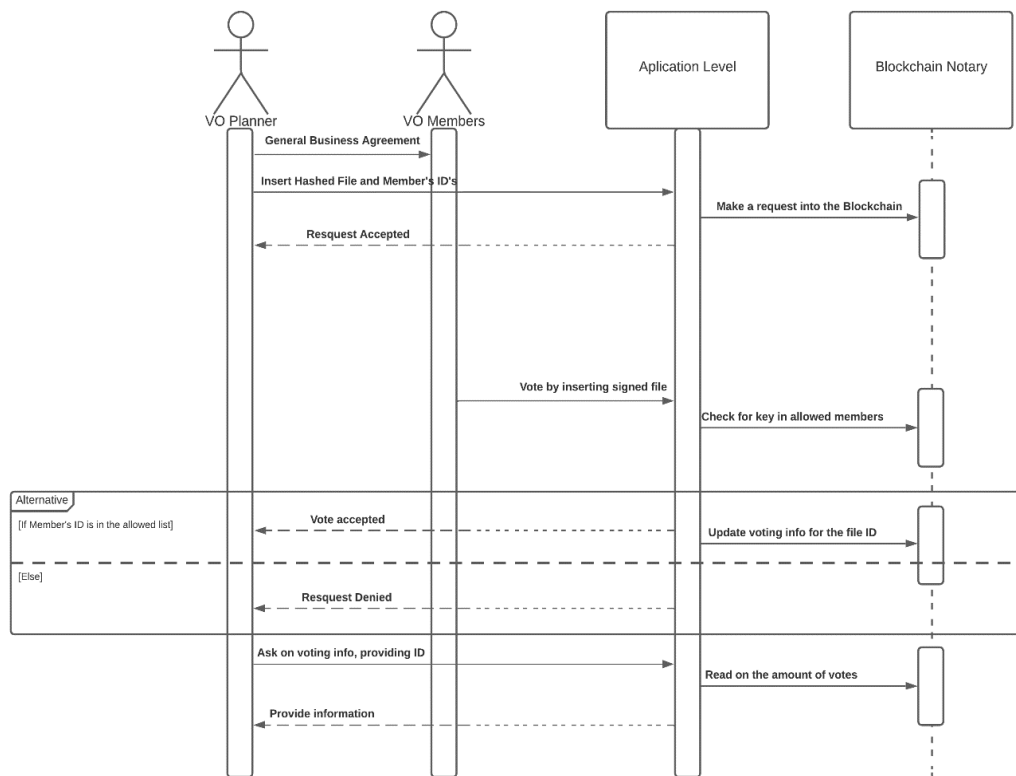


Figure 3.3: Sequence diagram of the interactions.

A system conceptual architecture diagram illustrated in Figure 3.4 helps describe, in broad terms, the different concepts and abstractions between the different conceptual layers in the system.

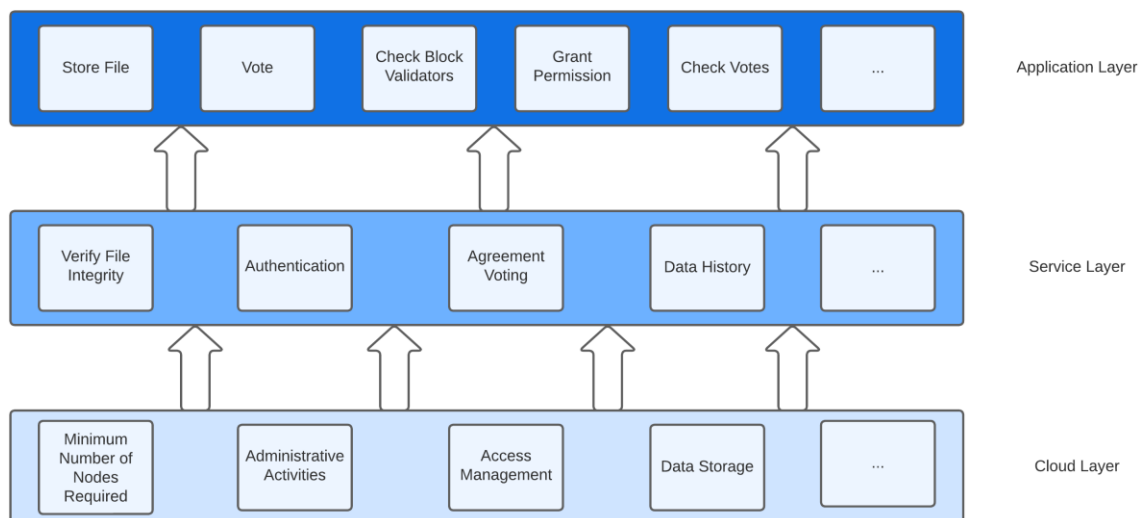


Figure 3.4: System conceptual architecture.

SYSTEM DEVELOPMENT

This chapter presents all the steps taken to develop the system solution that can fulfill the requirements and architecture presented in the previous chapter. The main features, optional characteristics, and workarounds to mitigate some of the challenges to develop this electronic notary are explained. Moreover, a description of the process for choosing the framework and client to use to build the blockchain network e-notary with the smart contract deployment, and launching it in a cloud service is also provided.

4.1 Framework for e-notary blockchain network

The first step to take when building a blockchain network is to choose a framework and client to use. In the background and related work chapter, an analysis of the most well-known and commonly used frameworks available and what differs between them.

For this thesis work, it was chosen the Ethereum framework for a variety of reasons. First of all, it has all the required characteristics to develop the project, starting from its availability to use other consensus algorithms and the ability of usage for other means than monetary-related subjects. Secondly, it's the most well-established framework that offers these characteristics and as a consequence, it has much wider community support and a variety of different clients working and developing tools for this framework providing much broader community support and consistent updates. Thirdly, it's the most recommended framework, among developers, for starting with blockchain development and learning (Li et al., 2017; Vacca et al., 2021), its communities are the most engaged in providing assistance for people initiating in this field and as a result do more frequent updates and compatibility testing between

different tools and approaches. The last reason is that as a result of having a much wider community than other frameworks, it offers a larger variety of secondary tools that are more intuitive and better suited for understanding the functionalities of blockchain development. While some of these tools serve for "hiding" away from the developer the more complex steps and setups to facilitate the learning process for much simpler solutions at the cost of taking away some options and variety, other tools offer opportunities for saving time and resources when developing with this framework.

Another noteworthy characteristic that needs to be pointed out from the Ethereum framework, is that it is based around tokens, sometimes referred to as "ether". So, independently of what it's being used for it is always working around this core functionality even if it is stated that interactions won't cost any tokens from the user. Every Ethereum client implements a common JSON remote procedure call (JSON-RPC) protocol, so regardless of the client and network at work, there is a list of common commands that can be used to read and write information on the blockchain by making requests with tools like, for example, Client URL (CURL) to nodes.

These methods can be separated into three categories: Gossip, State, and History. A very brief overview of some of these methods can be seen in Table 4.1. Gossip methods are responsible for sending data to the network and knowing which current state or block number a certain node is in. State methods serve to get information about the current state of certain stored data information that include methods to know how many tokens a user has, and what information he can retrieve from a contract or transactions sent by a certain address. Finally, the history-type methods fetch information written in all blocks starting from the genesis.

Table 4.1: Generic commands in Ethereum blockchain networks.

Method	Method Type	Description
eth_blockNumber	Gossip	Returns the number of the latest block
eth_sendRawTransaction	Gossip	Sends signed data of a transaction or creation of a contract, returns the transaction hash if successful

eth_getBalance	State	Returns the number of tokens a given address has either at a certain point in time (block) or in the earliest or latest recorded history
eth_getStorageAt	State	Returns the information of a certain storage data, needs the address of the storage along with its position within that storage
eth_estimateGas	State	Returns the number of tokens necessary for doing a specified transaction in the parameters
eth_getBlockTransactionCountByNumber	History	Returns the number of transactions of a block specified by its number
eth_getBlockByNumber	History	Gives all the information of a given block number, from headers, roots of information, and address of the publisher
eth_getTransactionByHash	History	Returns all the information regarding a transaction given the hash, from the block where it resides, the index, headers, and other input information

The Ethereum framework also establishes its architecture and nomenclature, starting from the term Ethereum Virtual Machine (EVM). The EVM is contained in every client software that runs as a node. The Ethereum nodes use the EVM as the computation engine that manages the state of the network by processing the data and updating the distributed ledger, it thus partakes in the process of block creation and transaction execution (function calls to a smart contract).

In terms of developer architecture for the Ethereum framework there is a distinction between two layers that are managed by different client software in an Ethereum node, namely the execution layer and the consensus layer. The description of each corresponding client is:

- **Execution Client:** An execution client is responsible for processing transactions, executing smart contracts, and maintaining the state of the blockchain (by processing in the EVM). It

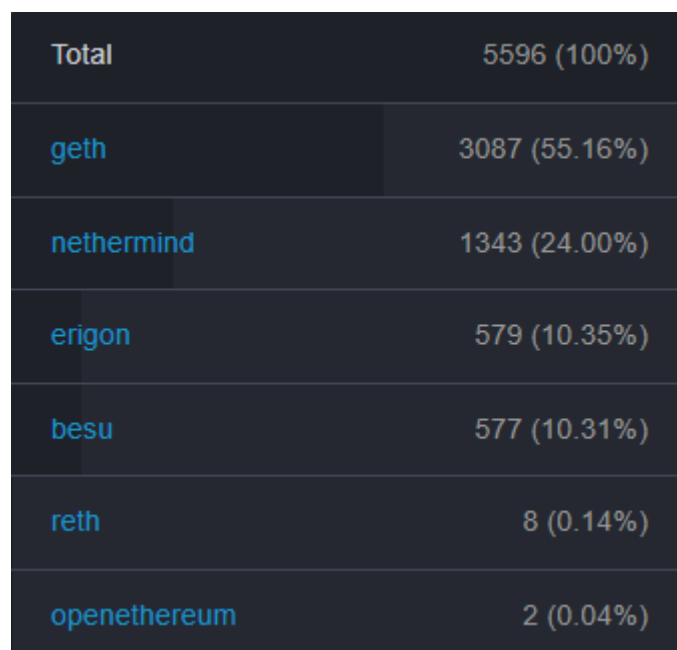
interprets and executes the code within smart contracts and updates the blockchain's state based on the outcomes of those executions.

- **Consensus Client:** Responsible for reaching agreement among the network participants on the order and validity of transactions. It implements the consensus algorithm that ensures all nodes in the network have a consistent and valid view of the blockchain.

While execution clients handle the processing of transactions and smart contracts, consensus clients ensure that all nodes agree on the state of the blockchain. In many blockchain systems, these two components are separate to achieve modularity and flexibility to reduce its dependability on a single codebase and single point of failure allowing also for more specification.

4.2 Ethereum Client

After choosing the framework to use for the development of the e-notary blockchain network it is important to select which Ethereum client software we want to develop the network with. There are a variety of different clients to choose from and some are more popular than others as Figure 4.1 shows.



Total	5596 (100%)
geth	3087 (55.16%)
nethermind	1343 (24.00%)
erigon	579 (10.35%)
besu	577 (10.31%)
reth	8 (0.14%)
openethereum	2 (0.04%)

Figure 4.1: Screenshot of the list of most used Ethereum clients, taken from <https://www.ethernodes.org/>.

For this thesis work, it was chosen to work with Hyperledger Besu client. This client offers a variety of different features that other clients do not and fits more into the usage for private networks that require some degree of centralization and trust among its users. A more comprehensive list of reasons that lead to choosing this client over others is as follows:

- **Enterprise Focus:** Being part of the Hyperledger project, Hyperledger Besu is focused on making a client that will be used for enterprises thus offering more forms of providing privacy, security, and performance for private blockchain networks.
- **Permissions and Privacy:** Hyperledger Besu offers tools that can control who is part of the network and allow users to set up permission lists that can dictate who they will accept information from and validate it. Thus, allowing for greater security and segmentation inside a network.
- **Pluggable Consensus:** For private networks, Hyperledger Besu offers not only an execution client but also a consensus client which many other clients do not offer together. Not only does this allow for fewer compatibility issues and time to set up and develop the network but also offers a wide range of different kinds of proof-of-authority consensus algorithms.

A simplified architecture of the client can be found in Figure 4.2. From that viewpoint, we can highlight some of the key characteristics of that come with this client. Namely the networking protocols for discovering peers and communicating with TCP transport protocols, an execution core where the EVM runs all the calls and transactions into valid blocks while also validating incoming blocks; the storage where all the world states reside with all the accounts and smart contract codes along with all the published blocks. Finally, there is an interface layer where any user or application can send RPC commands to manipulate the client running. The different architectural blocks listed can be manipulated to varying degrees by JSON calls either using generic commands from the Ethereum framework or commands exclusive to the client architecture being used. The illustration also displays the pluggable consensus that brings a higher degree of modularity which other clients do not offer.

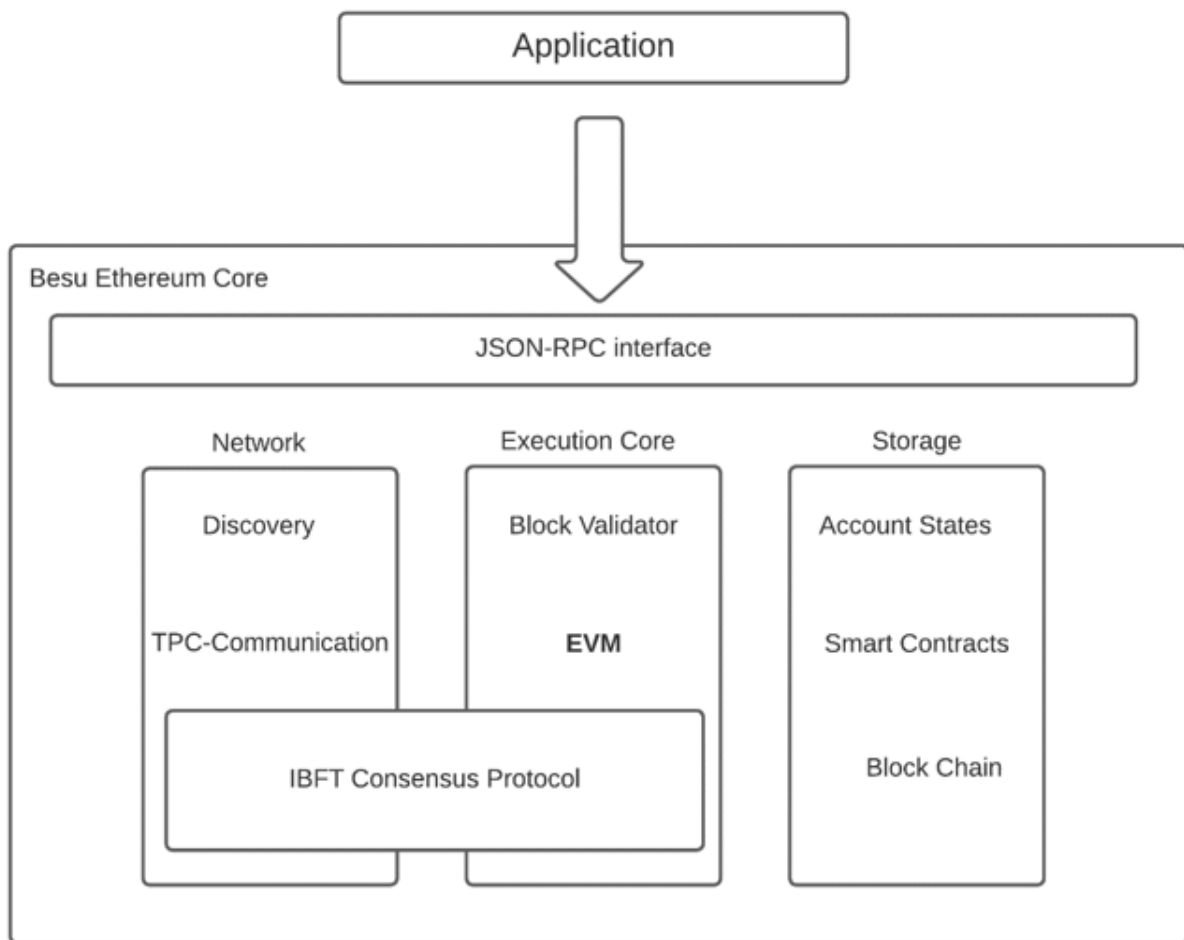


Figure 4.2: Architecture of a EVM for Hyperledger Besu.

4.3 Network Creation

With the framework and client tools selected the next phase of development focuses on using the aforementioned tools to create the blockchain network with the necessary characteristics primary functioning rules to work as a e-notary. Starting from the configuration of the genesis file that sets the basic rules for of the network to the necessary commands for launching the individual nodes to give the network it's private properties.

4.3.1 Genesis File Configuration

The first step for creating the network is to configure the genesis file. When a node wants to create a new block it needs to be able to have a full copy of the blockchain and thus it needs to know the instructions on how to create the first block and validate/create any subsequent

blocks in the network. The genesis file is a JSON file that defines the first block and how the network is going to operate, Figure 4.3 shows the genesis JSON file configured in this setup.

```
{
  "config": {
    "chainId": 2023,
    "shanghaiBlock": 0,
    "ibft2": {
      "blockperiodseconds": 3,
      "epochlength": 4000,
      "requesttimeoutseconds": 4
    }
  },
  "nonce": "0x0",
  "timestamp": "0x0",
  "extraData":
    "0xf85494df4523ecbeb5930fcc85341644ab3e20245213539445b4b33608e7b1b5dc5335
    63cb04177b9555201494438821c42b812fecdcea7fe8235806a412712fc094e96825c5ab8
    d145b9eeca1aba7ea3695e034911a",
  "gasLimit": "0xffffffffffffffff",
  "difficulty": "0x1",
  "mixHash":
    "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
  "coinbase": "0x000000000000000000000000000000000000000000000000",
  "alloc": {
    "45b4B33608e7B1B5dC533563cB04177b95552014": {
      "balance": "9999999999999999"
    }
  }
}
```

Figure 4.3: Genesis file configuration

The parameters that are present in the genesis JSON file are as follows:

- **chainId**: One of the two identifiers for the network, the other one being the network ID. For private networks, it is irrelevant the chosen ID and Besu automatically sets the network ID to be the same as chain ID.
- **shanghaiBlock**: Refers to the latest milestone block, which defines the latest protocol version of the network starting at block number zero. This can be used to create further updates on the network by specifying at which block number the nodes will agree to update their protocols.
- **ibft2**: The name of the proof-of-authority consensus protocol that will be used in this network and will be further explained later. Besu also gives access to other protocols for different purposes such as proof-of-stake or proof-of-work.

- **blockperiodseconds**: specifies at which block number the network will discard the votes cast by other nodes on who to remove or add to the network as a validator/node. If any existing proposals remain when this happens the nodes will recast their votes accordingly.
- **epochlength**: The number of blocks that mark the end of each epoch in the consensus algorithm. After each epoch all the votes casted to add a new validator or block are discarded. It is used as a safety mechanism for the network to staying in a deadlock for too long without human intervention.
- **requesttimeoutseconds**: The number of seconds for each round of votes for block publishing.
- **nonce**: The nonce value as previously mentioned in a previous chapter. It is a hash value consisting of all the information stored in the block.
- **timestamp**: The hexadecimal value representing the creation date of the block, can be any date that is in the past
- **extradata**: A recursive length prefix encoded string containing the public addresses of the initial validators for the network. Hyperledger Besu offers the functionality to create this encoded string from a JSON file containing the list of addresses. This string has four encoded addresses, the three minimum number of nodes required for the network to function, and one to work as a bootnode as will be explained later. This also puts a limit on the maximum size a block can have.
- **gasLimit**: Indicates the total value of tokens that can be spent for each block. Every interaction with the network has a token 'fee' value that is relative to the computational power required to make the interaction, since this private network is without token fees it will still be useful for ensuring that no client can make such a high computational power request that slows down the network performance because, despite not requiring any tokens the node will only accept interactions that do not exceed this value.
- **difficulty**: Value set to 1 for the consensus protocol to properly work. Otherwise, useless as it is only used for proof-of-work consensus.
- **mixHash**: unique identifier for the 'Istanbul' block identification required to run the IBFT protocol.
- **coinbase**: Address where the 'mining' reward tokens go to, set by default to address '0', this is of no use to our network.
- **alloc**: Sets the specified address with the decimal or hexadecimal number of tokens in the network, used here for debugging purposes.

Using this file multiple nodes on the same machine to simulate the network locally can be launched. This local simulation is identical to cloud deployment with four nodes running on a Linux virtual machine. The only difference is that this simulation case does not encapsulate the firewall and network setups to properly launch the network.

4.3.2 Launching The Network

To launch the network, it is necessary to be able to create nodes and Ethereum accounts to work as validators that will make it work. As later explained, for the consensus algorithm to work and the network to publish new blocks it is necessary to have, at minimum, three connected nodes.

In order to launch a node, it is required that the node has an identity, for this case, the node needs to have an associated Ethereum account comprising of a public key and a private key. The process for creating accounts and respective keys is a standard protocol in the Ethereum framework, meaning that any Ethereum account is viable to any blockchain network running on the same framework as the validity of the key pairs is checked and created using the same mechanisms. This also means that any program that creates Ethereum accounts and manages these accounts can be used to create valid addresses for this network.

Hyperledger Besu doesn't have the ability to create Ethereum accounts, so for this reason, the accounts were created using an open-source tool named Metamask. Metamask offers a wide range of different functionalities, namely the ability to create and manage keys, and tokens and connect to different applications and blockchain networks while displaying information about transactions and tokens in the network. As it is an Internet browser extension it does not store the user keys on a dedicated remote server. It leaves the questions regarding security up to the user as he can export the created account keys into files or import from those files into the browser extension. Figure 4.4 displays the basic panel for the account creation screen and subsequent management main screen. On the left of the figure, it displays the number of tokens the selected account has inside the connected network (in this case it is a pre-selected public test network). The activity regarding any recent interactions the account made along with the token balance change. On the top of the screen, it displays the public address of the account to facilitate sharing it with other users. On the right side of Figure 4.4, there is the account creation screen which generates an Ethereum account based on the account name.

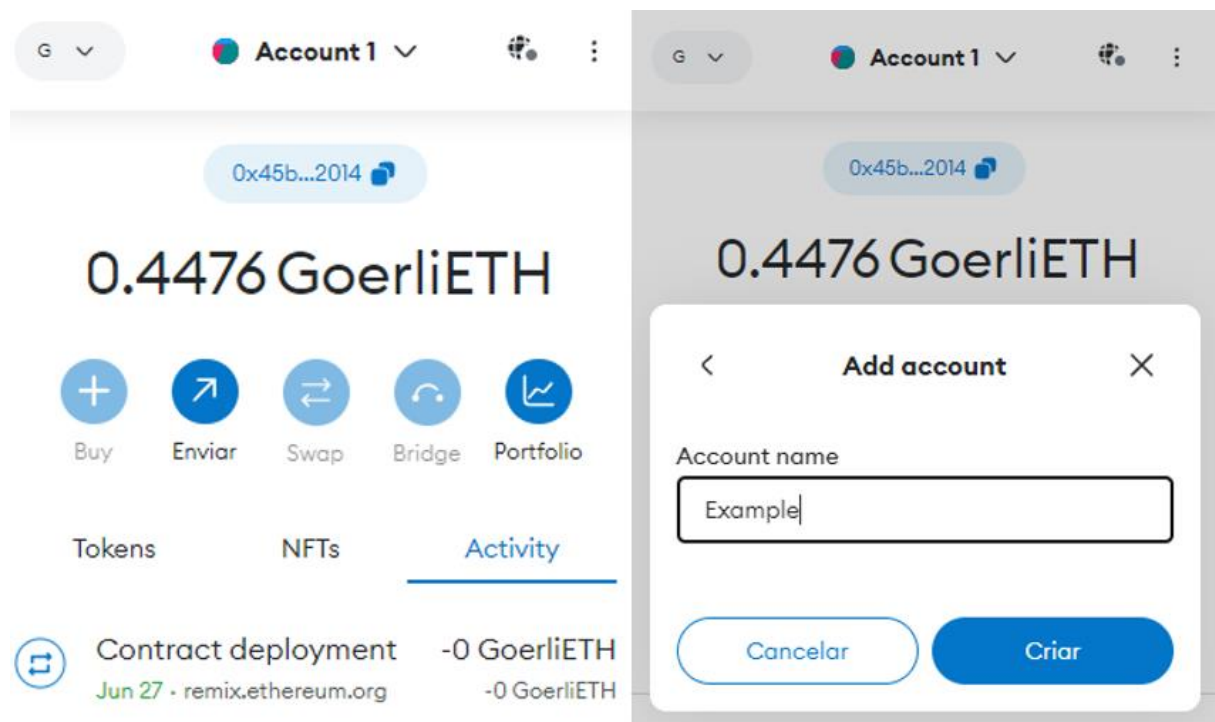


Figure 4.4: Account creation and management screen.

After creating four different accounts, it is possible to export their public and private keys from Metamask into the local machine. These keys will be in different files, namely the **PubKey.txt** containing the public key and the **PriKey.pub** with the private key of the respective account. Since nodes require a directory to work in and the directory also requires the keys of the node's account to take an identity inside the network, Figure 4.5 describes how the directories were organized for each node.

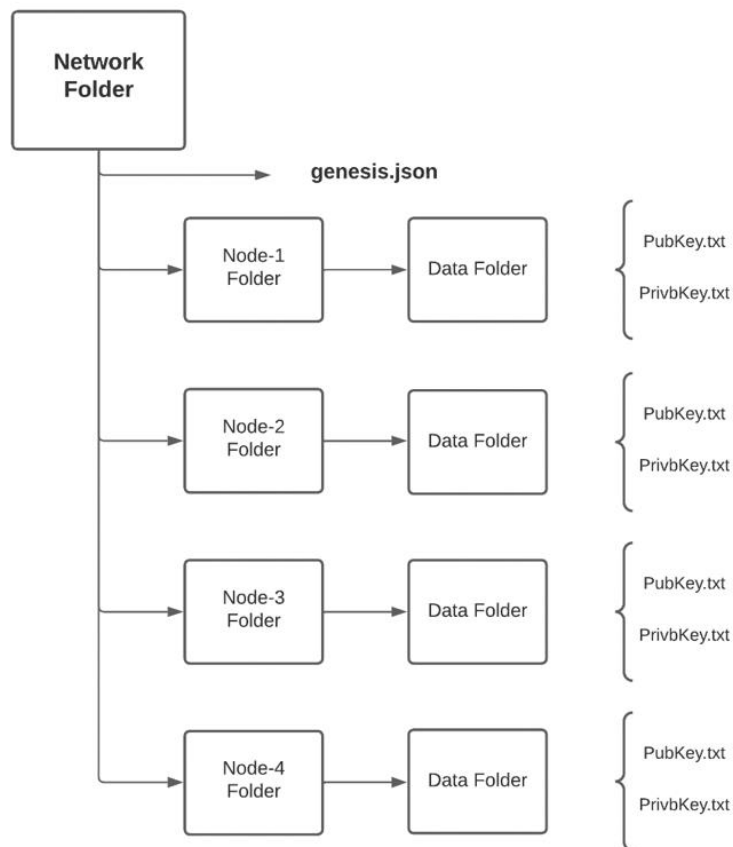


Figure 4.5: Directory organization of the nodes

When a node wants to join a network, it needs to connect to available nodes already inside the network and let them discover new peers. A bootnode is referred to as a node that works as an entry point for new nodes to join the network and despite being connected to the network it can be configured to not take part in the process of publishing blocks. The key feature of a bootnode is how publicly available it is, so it is not behind the Network Address Translation firewall. Therefore, it can be accessible to users outside the local network. To launch the first node as a bootnode, it is necessary to use the command displayed in Figure 4.6 using Hyperledger Besu.

```

besu --data-path=Node_data --genesis-file=./genesis.json --rpc-http-
enabled --rpc-http-api=ETH,NET,IBFT --host-allowlist="*" --p2p-
host=0.0.0.0 --rpc-http-host=0.0.0.0 --p2p-port=30303 --rpc-http-
port=8545 --rpc-http-cors-origins="all"

```

Figure 4.6: Bootnode launching setup in Hyperledger Besu

To better comprehend the command a description follows:

- **--data-path:** points to a folder for the data folder of the node, which will contain the information of the blockchain.
- **--genesis-file:** specifies the genesis file, the first block in the chain, and all the instructions of how the network is going to function.
- **--rpc-http-enabled:** tells the node to have RPC enabled
- **--rpc-http-api:** Will enable the ETH, NET, and IBFT APIs which will allow for the use of the listed API commands over the interface. These are the specific commands that are Ethereum-related (ETH), the network-related commands that provide information about the network and the peers and finally the IBFT API which gives the commands that allow manipulating the validators and consensus related information.
- **--host-allowlist:** list of hostnames that can the JSON-RPC API, if left with "*" we allow to accept requests from all host names.
- **--p2p-host:** Specifies the IP address the peer-to-peer network interface of the node will listen to. For testing proposes set to "0.0.0.0" meaning it will listen to all available network interfaces, which can pose a security problem without firewall protection. If this command is not called the default will be '127.0.0.1'.
- **--rpc-http-host:** The IP address that the JSON-RPC interface will listen to. Similar to the p2p command it poses the same potential security issues and will just be set to '0.0.0.0' for easier simulation. If also left unset the default will be '127.0.0.1'.
- **--p2p-port:** The listening ports for peer-to-peer communication for TPC and UDP, default is 3030, but other nodes need different ports.
- **--rpc-http-port:** Same as the peer-to-peer the default, the TPC port which JSON-RPC listens to.
- **--rpc-http-cors-origins:** Specifies the list of origins that are permitted to make HTTP requests to the node's JSON-RPC interface, this can be set so only specified web applications can interact with the node interface. For this example it was set to "*" meaning to accept all origins, by default, it is set to none.

After launching the first node as bootnode the terminal will display a enode URL which is a way to specify the node in the form of a URL that contains the node's public key followed by a "@" and the IP address (0.0.0.0) and the peer-to-peer port (30303) as Figure 4.7 shows.

```
Enode URL
enode://73d0b311d141d876689f52927b1547d790b3a3f5a44eaea470b08
ab7967033cd0a48f39fb06d4d812d494440abd748315b8a7ab5626974c3d4
5c7f7b7ff9916c@0.0.0.0:30303
```

Figure 4.7: Node URL of the bootnode.

When the bootnode is up, the other three nodes will need to also be launched and specified to connect to the first bootnode via the enode URL. They will also need to have their ports for peer-to-peer and JSON-RPC communication, the rest of the specifics will be similar to the first node as with the data folder and genesis file. Figure 4.8 shows the example to launch node 2, all subsequent nodes are launched in the same way but using different ports.

```
besu --data-path=Node_data --genesis-file=../genesis.json --
bootnodes=enode://73d0b311d141d876689f52927b1547d790b3a3f5a44eaea470b08ab
7967033cd0a48f39fb06d4d812d494440abd748315b8a7ab5626974c3d45c7f7b7ff9916c
@0.0.0.0:30303 --p2p-port=30304 --rpc-http-enabled --rpc-http-
api=ETH,NET,IBFT --host-allowlist="*" --rpc-http-cors-origins="all" --
rpc-http-port=8546
```

Figure 4.8: Configuration command to launch a node and connect to the bootnode.

When all four nodes are launched the network begins publishing blocks according to the rules specified in the genesis.json. Further tests to verify if the network is working properly can be done by making an HTTP request to the bootnode for information regarding the validators of the network. Figure 4.9 displays the command "ibft_getValidatorsByBlockNumber" of the latest published block by sending the request to the address of the bootnode used for HTTP calls.

```
curl -X POST --data
'{"jsonrpc":"2.0","method":"ibft_getValidatorsByBlockNumber","params":["l
atest"], "id":1}' 0.0.0.0:8545
```

Figure 4.9: Command to verify the validator nodes.

The following result displaying the addresses of the four nodes launched and listed as validators in the network is shown in Figure 4.10.

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": [
    "0x45b4B33608e7B1B5dC533563cB04177b95552014",
    "0xB65a61610257c13706300C0C5CB3aa965F554937",
    "0xB50e21d2806EE7B0e522ff87ff29c4dF9afA1169",
    "0x57Fcdb0F7417c551231D01BDcA1Db4D71EF4D0c4"
  ]
}
```

Figure 4.10: Result of the command, displaying the addresses of the validator nodes.

4.3.3 Consensus Algorithm

As previously mentioned, the network will be running a proof-of-authority (PoA) consensus algorithm, called Istanbul Byzantine Fault Tolerance (IBFT). This mechanism is divided into three major phases and always requires a supermajority of votes to publish the next block, thus needing three nodes at minimum to work. The network has **N** number of nodes where it can tolerate **F** faulty nodes where $N = 3F + 1$.

When a round starts a node will be selected to propose the next block through a round-robin selection. This node will broadcast to all other nodes the 'Proposal' with the information of the block. The nodes that receive the message will check if it is from a valid node. If the information of the block headers is also valid and if they are in the same round sequence, if all of these are correct then the nodes will broadcast a 'Prepare' message while waiting to receive $2F + 1$ valid 'Prepared' messages from the other nodes.

To be regarded as a valid message it will need to also have the matched sequence round, the block hash and be part of the known validator's node list. When it reaches this required number of valid messages from other broadcasts it will enter the 'Commit' phase where the same process will be repeated. Otherwise, if at any phase the node does not receive the number of required messages, it will timeout and a new round will begin.

When the necessary number of 'Commit' messages is reached, the node will add the block to the chain with the 'extraData' field containing the $2F + 1$ signatures of the validators. Finally, a diagram illustrating the sequence of states of this algorithm can be seen in Figure 4.11.

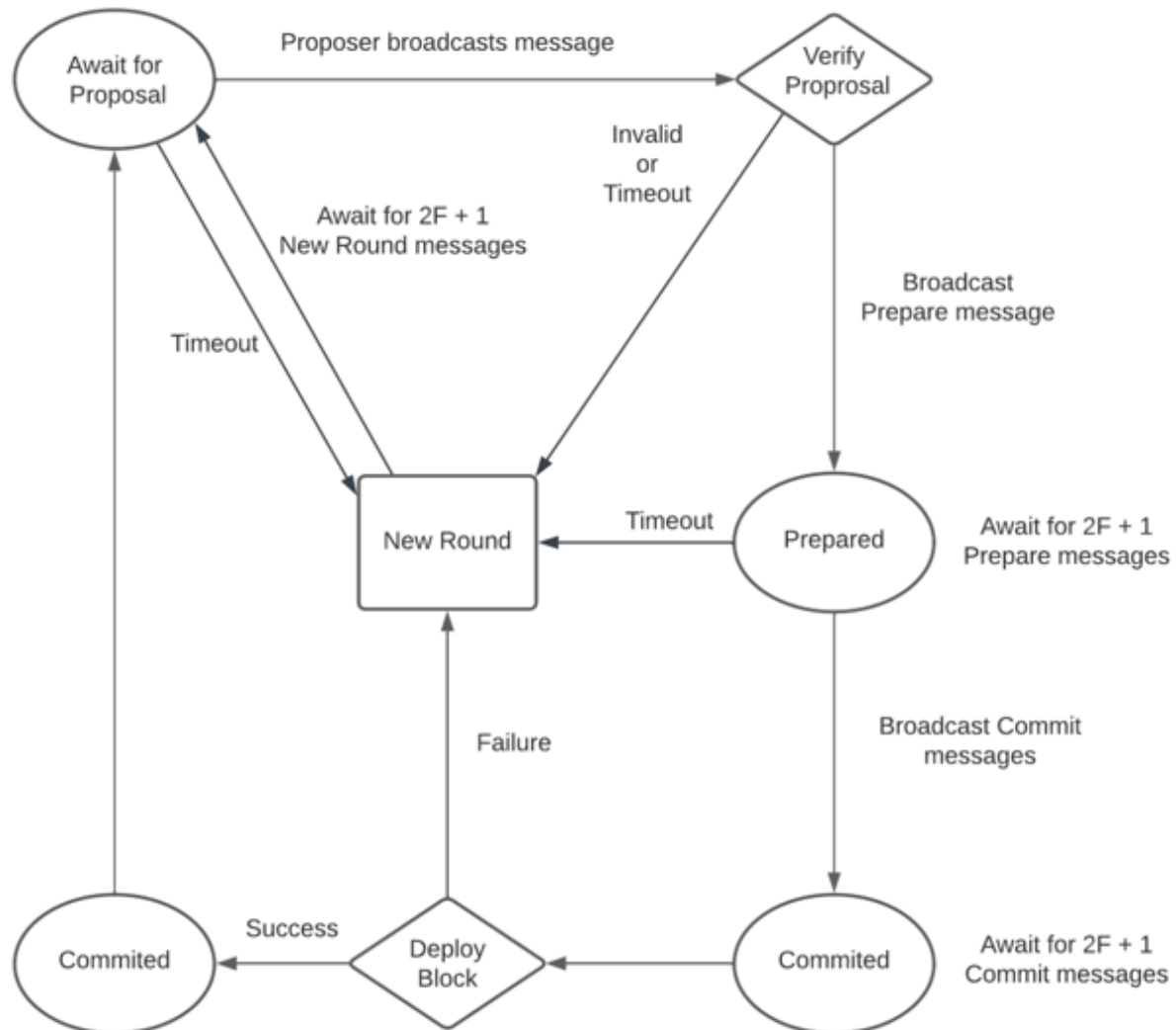


Figure 4.11: Illustration of the consensus algorithm.

Compared to other relatively simple PoA algorithms, IBFT 2.0 offers more security checks through rounds of votes rather than being a simple validator round robin or other types of Byzantine fault tolerance. One of these problems is that a Byzantine proposer can reach a consensus on two different blocks with two different sets of honest validators and thus lock the chain. With IBFT 2.0, adding the different broadcasts of messages and the requirement for the supermajority of votes, there is no such problem. This comes at the cost of a rather slower network performance since more messages and votes will need to be collected for the protocol to work and some scalability issues in the long run. This algorithm then makes a fine adjustment between more security at the cost of less performance when compared with standard PoA.

Since the network needs to have the ability to add new nodes to extend the list of validators it is necessary to be able to add and remove validators from the network. This is done using the command in Figure 4.12

```
curl -X POST --data
'{"jsonrpc":"2.0","method":"ibft_proposeValidatorVote","params":["0xAd880
1F2EcFd0965f8f9507e755C4C571CBa0CC5", true], "id":1}' http://0.0.0.0:8545
```

Figure 4.12: Command to vote on adding a new validator node to the network.

By using the command to propose a new validator along with the address of the new node account, the other validators can allow a new address to validate blocks. This needs to be repeated in the super majority of nodes to successfully add the new address. To verify the end result, it is possible to recast the previous command in Figure 4.9 to get the output displayed in Figure 4.13.

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": [
    "0x45b4B33608e7B1B5dC533563cB04177b95552014",
    "0xB65a61610257c13706300C0C5CB3aa965F554937",
    "0xB50e21d2806EE7B0e522ff87ff29c4dF9afA1169",
    "0x57Fcdb0F7417c551231D01BDcA1Db4D71EF4D0c4",
    "0xAd8801F2EcFd0965f8f9507e755C4C571CBa0CC5",
  ]
}
```

Figure 4.13: List of validators displaying the newly added node into the network.

4.4 E-Notary Smart-Contract

Through the development of the smart contract, it is possible to encapsulate the core principles of a traditional electronic notary to achieve greater security and trustworthiness. The smart contract requires some core principles in order to make it behave the way that is desired. For a blockchain network, a smart contract is the same as an Ethereum account, this means that other users/accounts can make transactions to execute the functions inside the contract.

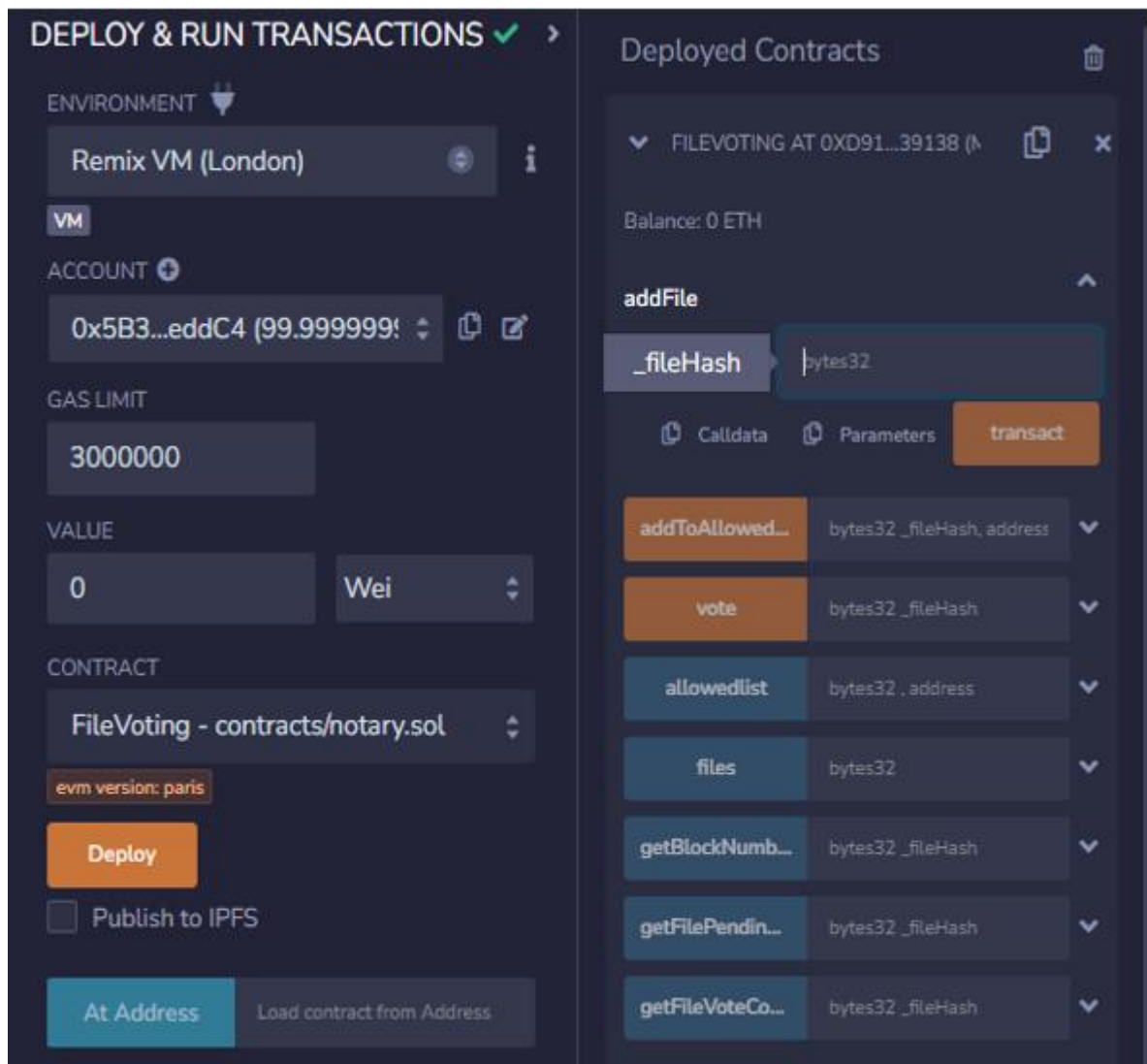
In Ethereum, smart contracts are usually coded in high-level, developer-friendly programming languages like Solidity or Vyper. The smart contract code must be compiled into bytecode so the EVM inside the node can understand it. The tool that was used to help develop the smart contract code and test it was Remix IDE Ethereum, an open-source web browser application that allows developers to code smart contracts in Solidity, compile them, and deploy them to debug the functions using dedicated test blockchain networks. Alternatively, Remix IDE can also deploy and interact with smart contracts on networks that are connected via Metamask.

A brief overview of the main developed smart contract functions can be found in Table 4.2, and a complete overview of the source code in Annex A.

Table 4.2: Smart Contract Functions

Function	Description
addFile	Receives a string of the hash of a contract file and converts it to a byte32 array, creating a structure of the file information such as the number of votes, pending votes, and the address of the original uploader.
addToAllowedList	Uses the input string of hash and public address to put the address in an 'Allowed List' to be able to vote on the contract. Only the owner of the original file can give this permission.
vote	With the input of the same hash string the contract checks if the user is in the allowed list or not and adds his vote. Users can only vote once per file and cannot undo the vote once cast.
getFileVoteCount	Returns the number of people that have voted in the file.
getFilePendingVotes	Returns the number of users that have not voted yet but were allowed to vote by the owner.

After the solidity code is done, it can be compiled in Remix so it turns the solidity code into bytecode instructions for the EVM core to read, we can choose where we want to deploy the smart contract, since Remix IDE offers test networks to debug the contract functions. Figure 4.14 shows the test network selected to deploy the contract, with Remix IDE offering various addresses to switch from easily when interacting with the contract, it also has the list of functions that were developed and easily called.



(a)

(b)

Figure 4.14: a) Network selection screen. b) Contract function's screen

After debugging the code, it can be deployed in the e-notary blockchain network. This can be done in two possible ways. First could be to create a Java script file that would compile and translate the contract into byte code, which would be used in the command "eth_sendRawTransaction" with the byte code in the data field and "null" on the receiver parameter. The second method would be to use Remix IDE to deploy via Metamask. Here could be necessary to connect Metamask to the network that is deployed and running on the local machine. This could be done by adding a new network via the menu with the criteria shown in Figure 4.15 (b). The Chain ID must match the one in the genesis file, and the RPC URL must be the same as the setup in the first node along with the same specified port. After connecting to

the network via Metamask it is possible to interact with the blockchain network via Remix IDE by compiling the code and deploying via Injected provider - Metamask as in Figure 4.15 (a).

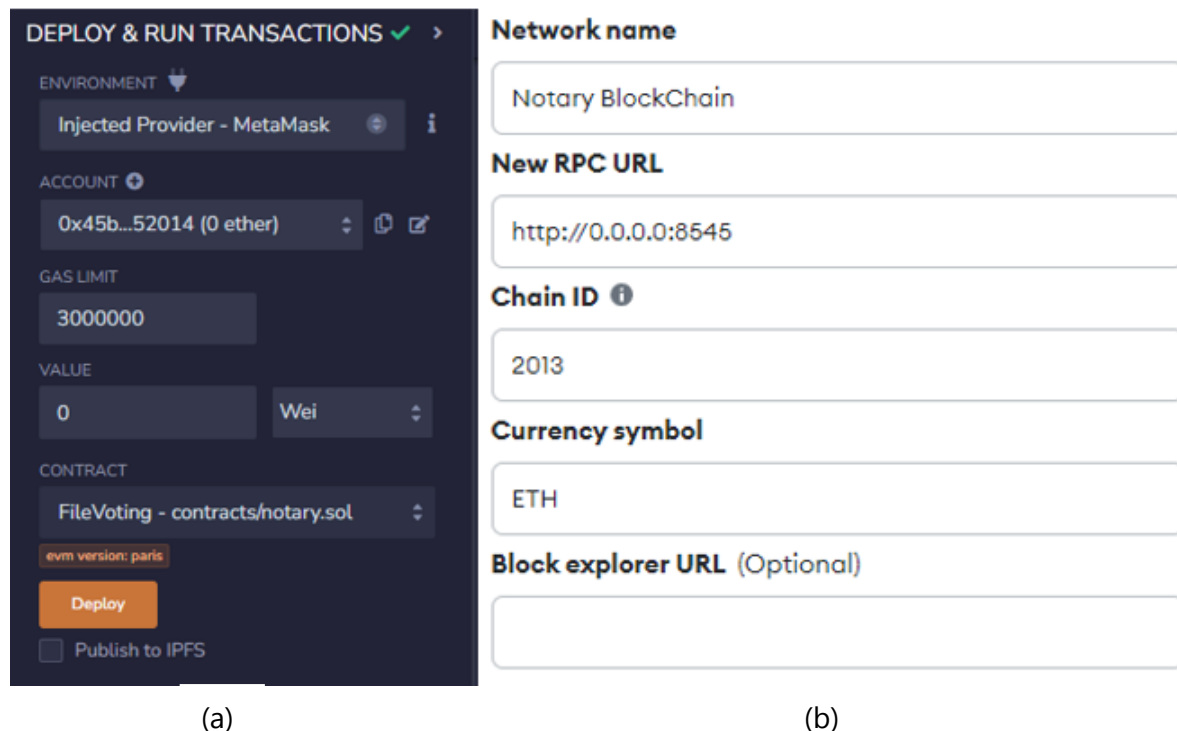


Figure 4.15: a) Metamask connected to Remix IDE. b) Metmask connected

4.5 Application Interface

For the blockchain networks, applications that interact with these networks are referred to as decentralized applications (dapp). They are usually the frontend user interface that is built to interact with a smart contract deployed on a blockchain network. In contrast to ordinary applications that have a backend code in a centralized server. The premise of a dapp lies in having its entire backend "living" in the decentralized server that comprises the entire blockchain network.

Dapps makes use of client libraries to forward JSON-RPC requests to Ethereum blockchain networks, and some of the more famous libraries are Ethers.js and Web3.js. These libraries provide a higher-level interface for interacting and composing calls for the blockchain network. For applications to properly know what library functions they need to call to interact with a smart contract deployed on a blockchain, they need to have the contract application binary interface (Contract ABI); The contract ABI will have the contract's functions in a JSON format and allows a developer to use as a regular function. So, in abstract terms, the ABI along

with the client libraries will work as the interface between the application and its regular data and the bytecode transactions that the EVM of a node inside a blockchain network can read and execute.

To further facilitate the development of these dapps, tools like Metamask behave as "Web3 providers" and become the interface that exchanges information between the application and the blockchain network. In this work, Metamask is be used as this interface to facilitate the interaction with the application interact with the network, while also providing the methods that allow the app to authenticate the account thus separating the provider from the signer. Figure 4.16 illustrates an overview layout of the relationship between the application and the network.

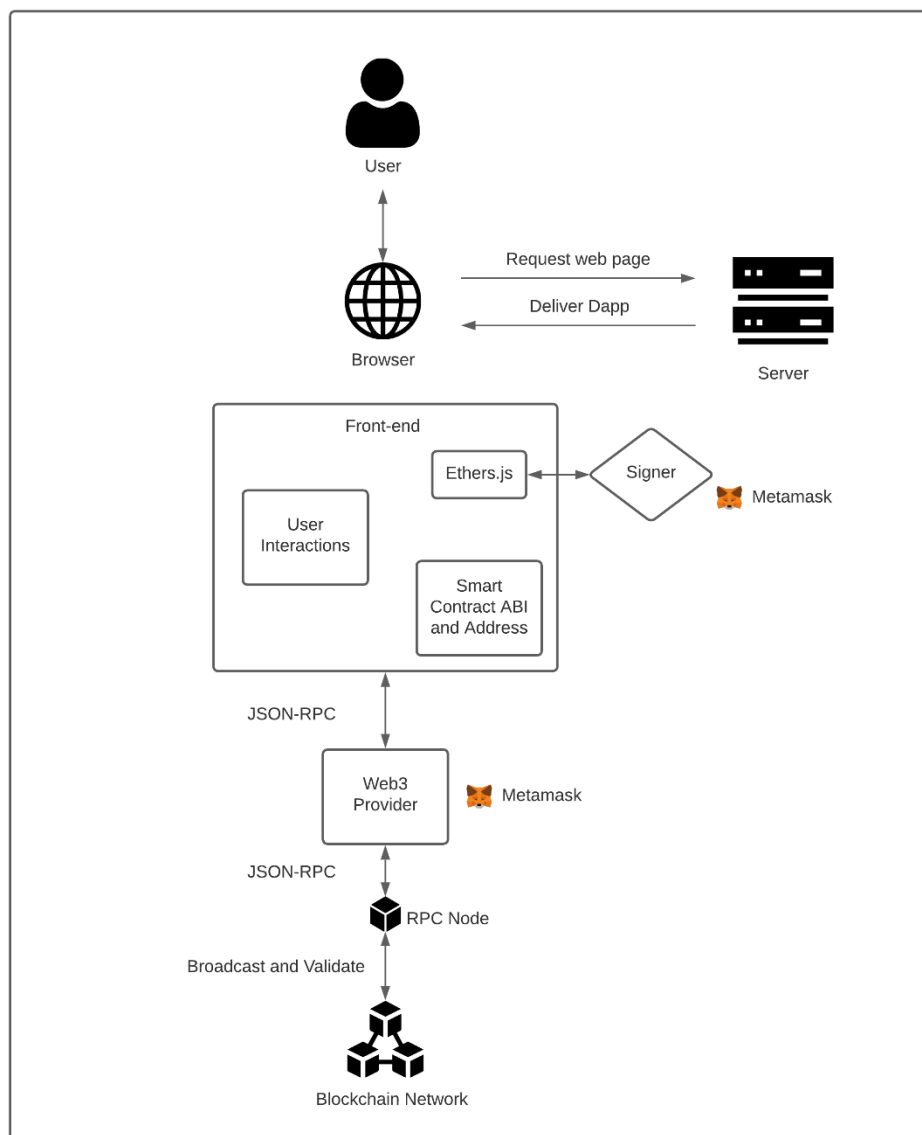


Figure 4.16: Layout of the dapp interaction with the e-notary blockchain network.

During the development of the front-end, the JavaScript library Ethers.js was used to enable the creation of Ethereum objects that authenticate and call smart contract functions with Metamask via the provider object "window.ethereum". Figure 4.17 displays how the front-end detects the account of the user and if metamask is installed or not.

```
import React, { useEffect, useState } from "react";
import { ethers } from "ethers";
import "./Application.css";
import File_ABI from "./notary.json";

const getEthereumObject = () => window.ethereum;

const findMetaMaskAccount = async () => {
  const ethereum = getEthereumObject();
  //check if metamask is installed
  if (!ethereum) {
    return null;
  }

  console.log("Metamask is installed", ethereum);
  const accounts = await ethereum.request({ method: "eth_accounts" });

  if (accounts.length !== 0) {
    return accounts[0];
  } else {
    //No accounts found

    console.log("No account detected");

    return null;
  }
};
```

Figure 4.17: Base code to verify if the user is using the Metamask extension.

To call smart contracts functions from the front-end it is necessary to have the contract's ABI file imported and the retrieved contract address, as Figure 4.18 displays.

```

//...
const contractAddress = "0x39Bc8345F9714F17e02f215130F6bF5D827465e6";
const contractABI = abi.abi;
//...
if (ethereum) {
  const Web3_provider = new ethers.providers.Web3Provider(ethereum);
  const signer = provider.getSigner();
  const NotaryContract = new ethers.Contract(
    contractAddress,
    contractABI,
    signer
  );
  callContractFunction("store_file", Input1);
}

```

Figure 4.18: Front-end function to call the smart contract function when one of the interface buttons is pressed.

4.6 Cloud Deployment

The final development step, was to find a solution for a e-notary that can fit within the context of collaborative networks. After having the blockchain network and the application front-end for the e-notary, there is the need to have an infrastructure that can hold some of the blockchain network nodes while providing the service to everyone inside a VBE. Also, this infrastructure needs to be able to allow VBE members to assist in expanding the blockchain network and thus increase the trust and decentralized nature of the network. This goes in line with the topic of the "blockchain trilemma" (mentioned in Chapter 2), where a balance and compromise between the decentralization, scalability, and security of the network is required. From the range of possible solutions, this work uses a cloud deployment as the base infrastructure for the e-notary blockchain network.

The reasons that lead to Cloud Deployment are the bigger concern on the security aspect of the network and the fact that more complex solutions that would involve, for example, virtual private networks would require much more complex resource management and the complex architecture that would be out of the scope of this project work. Moreover, also specifics of the VBE can be considered such as centralization, around its administrator and what characteristics the community in specific has ranging from the sector and amount of

members, to the geographical distribution of its members. So, the cloud deployment works as a very general solution with some aspects left blank on purpose because they would entirely depend on the VBE in specific.

The chosen platform to make a cloud deployment was Microsoft Azure. The high-level architecture of the e-notary blockchain network inside the cloud virtual machine is displayed in Figure 4.20. Overall, the same steps previously mentioned in the network creation section were made when launching the nodes inside the virtual machine. Thus, four nodes were launched with the specificity that the bootnode is the only point of entry that receives calls from outside applications and specifically from the front-end application. This node is also the only one that allows authorized users to connect their local machines as additional nodes of the blockchain network. Figure 4.19 displays the structure of the command that is used when launching the bootnode inside the cloud virtual machine.

```
besu --data-path=Node_data --genesis-file=../genesis.json --rpc-http-  
enabled --rpc-http-api=ETH,NET,IBFT --host-allowlist="*" --p2p-  
host=<VM_PUBLIC_IP> --rpc-http-host=<VM_PUBLIC_IP> --p2p-port=30303 --  
rpc-http-port=8545 --nat-method=NONE --rpc-http-cors-origins=chrome-  
extension://nkbihfbeogaeaoehlefnkodbefgpgknn,http://remix.ethereum.org,  
<front-end_URL>
```

Figure 4.19: Configuration command used to launch the bootnode inside the cloud virtual machine.

Some of the changes of launching the node compared to the local network production in previous subsections are: the p2p and RPC host being replaced with the public IP of the cloud so the node can be accessed from outside either to expand the network or to receive JSON calls from the application; the nat-method is set to 'NONE' so it does not overwrite these values, lastly, the 'cor-origins' instead of receiving calls from any domain, is restricted to the domain of the metamask chrome extension URL, the remix IDE tool used to deploy the contract, and the front-end URL that will be hosted on Replit for faster implementation.

Lastly, there are some considerations when setting up the cloud, namely the firewall rules need to allow port forwarding, inbound and outbound calls to the RPC and p2p ports of the first node both in TCP and UDP protocols. This enables so this node to be connected from outside machines (VBE members) to extend the blockchain network and only accept calls from the front-end browser page with Metamask. Other firewall rules will largely depend on the level of access the VBE administrator that owns the cloud service, will give to other VBE members to give private access to the network and not the whole public.

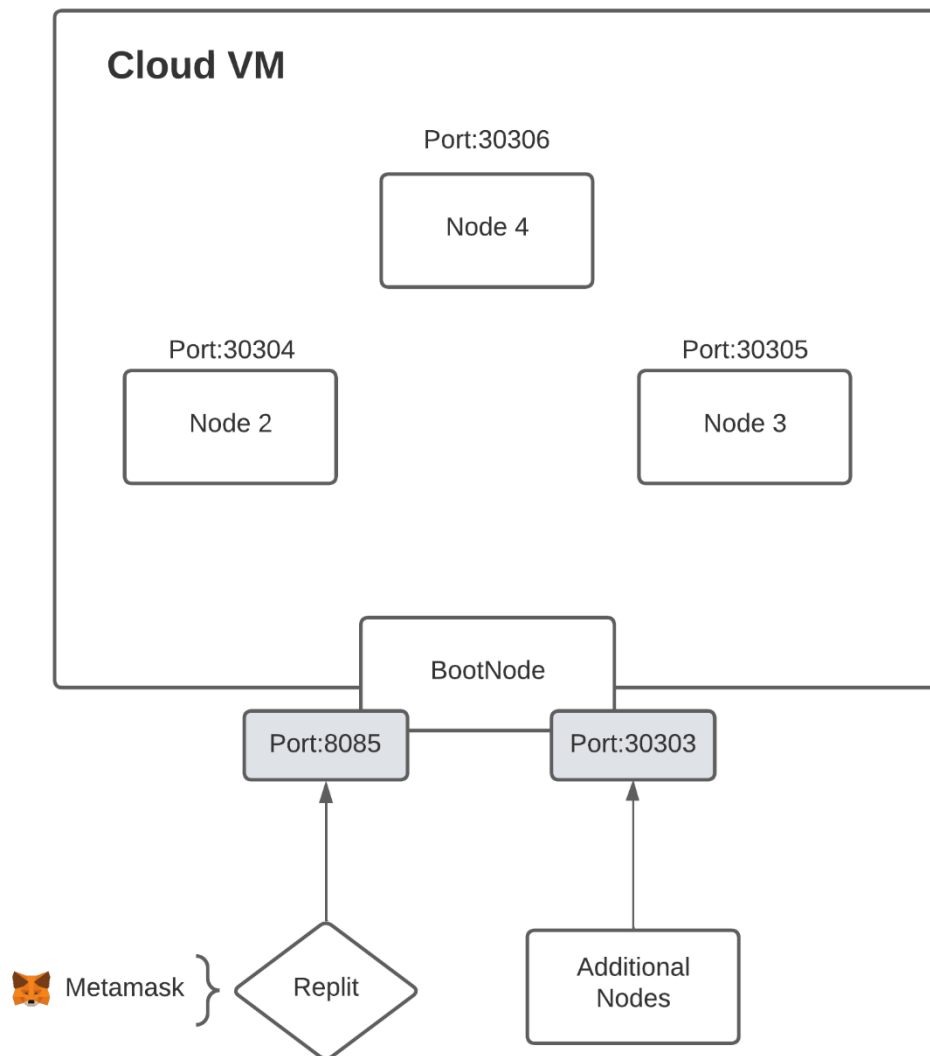


Figure 4.20: High-level architecture of the solution with the cloud deployment.

RESULTS & TESTING

The previous chapter displayed all the steps taken to build and deploy the network starting from the blockchain network to smart contract code and finally the application and cloud deployment. This chapter showcases the results of interacting with the network while deployed in its final state by: interacting with the smart contract for storing files; adding permission to other users to vote on the contract; displaying the application interface; and adding new nodes to the network with permission from the cloud administrator and any other additional nodes in the network.

5.1 Blockchain Network Results

While there are some metrics to measure the performance of a blockchain network, these metrics can often be misleading. Some of these measures such as Transactions Per Second or Time to Finality won't tell much about the performance of this specific use case network. Transactions per second, as the name implies, is a measure of how many transactions per second the network can handle, which while a useful metric for public blockchain networks using this measure for this specific case which is a community with a rather limited size of participants and transactions it won't be a problem. The same also applies to the Time to Finality measure which measures the amount of time to confirm a transaction. These measures won't be useful for the case of private networks and can be misleading since the performance score on these metrics can be high at the detriment of having a network that is not too secure or scalable.

This leaves the only solution being to test the functionalities of the private network by using it and note if it meets the demands listed in the application design. Starting from the same sequence of development steps in Figure 5.1 displays a screenshot of a terminal running a node of the private blockchain network, displaying the number of the block it added to the

chain by importing from another proposer node or by proposing the block and broadcasting to the other nodes.

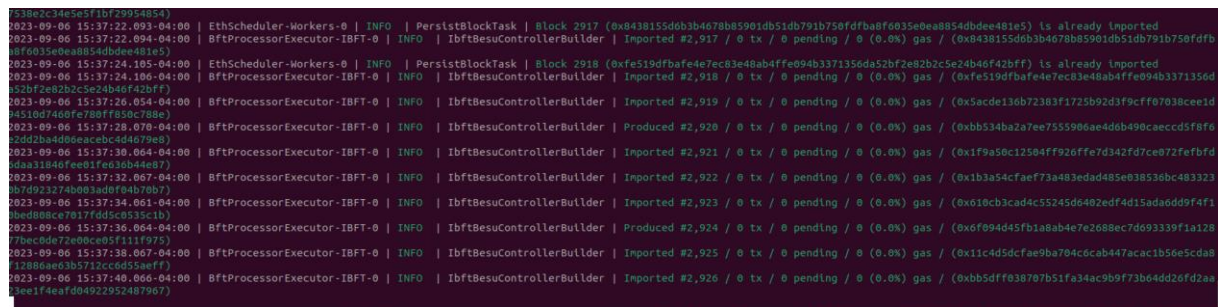


Figure 5.1: Screenshot of console monitoring the process of a node creating and importing blocks.

It is possible to add new validators to this network deployed in the cloud by creating a node on any local machine and by remote control over cloud to the virtual machine. This is done using the command to propose a new validator by sending the proposal with the new node address to three of the cloud nodes, like mentioned on the previous chapter. This way the VBE administrator has some degree of authority over the e-notary and can grant cloud access through the firewall settings to any of the VBE members who wish to assist in expanding the network by granting access through the firewall to access the p2p port and RPC HTTP port of the bootnode.

5.2 Contract Interaction

The deployment of the notary smart contract is done using Remix IDE through one of the node's addresses inside the cloud. Figure 5.2 displays the deployment transaction certificate.

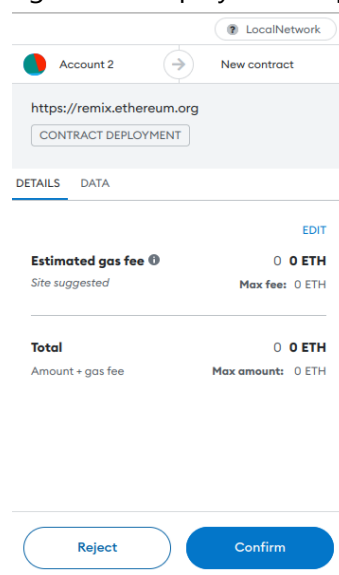


Figure 5.2: Metamask activity log of the contract deployment.

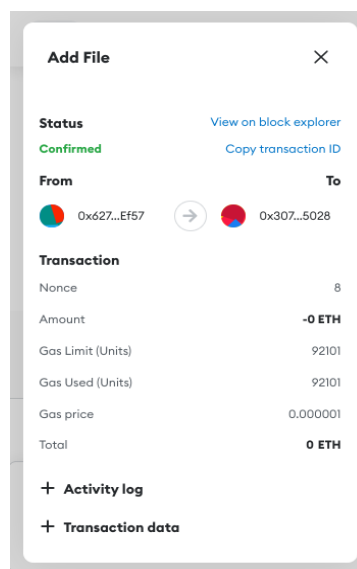
From an outside local machine, it is possible to begin adding a file to verify if the smart contract is working inside the blockchain network. Starting by using any online tool to calculate the hash of a simple PDF file and get the value then in Metamask. Then the connection to the RPC port of the cloud and the connection to Remix IDE to Metamask are established, with the address of the contract to interact with it and store the string hash file inside the blockchain. Remix IDE displays the outcome from that interaction in Figure 5.3 (a) and the transaction log inside Metamask in Figure 5.3 (b).

```

[block:3384 txIndex:0] from: 0x627...BEf57 to: Storage.addFile(string) 0x307...95028 value: 0 wei data: 0xeb9...43764 logs: 1 hash: 0xd82...5aef4
status true Transaction mined and execution succeed
transaction hash 0x200efb8eccde361aa3ab32184741e3aee96a34f47d729302b7dada7ca6960e8
block hash 0xd82fda2327253dde158c54b59a67209bb693f9f875a857bc458a82de9b5aef4
block number 3384
from 0x627306990aba83A6e1400e9345bC60c78a8BEf57
to Storage.addFile(string) 0x30753E4A8aad7F8597332E813735Def5d0395028
gas 92101 gas
transaction cost 92101 gas
input 0xeb9...43764
decoded input {
  "string file": "d6be96a875d81ealae27d03a273b47d"
}
decoded output -
logs [
  {
    "from": "0x30753E4A8aad7F8597332E813735Def5d0395028",
    "topic": "0xb957499a7658fda76a577d75c1d04db0f5231823087af49c0da0691520dd9027",
    "event": "fileAdded",
    "args": {
      "0": "d6be96a875d81ealae27d03a273b47d",
      "file": "d6be96a875d81ealae27d03a273b47d"
    }
  }
]
val 0 wei

```

(a)



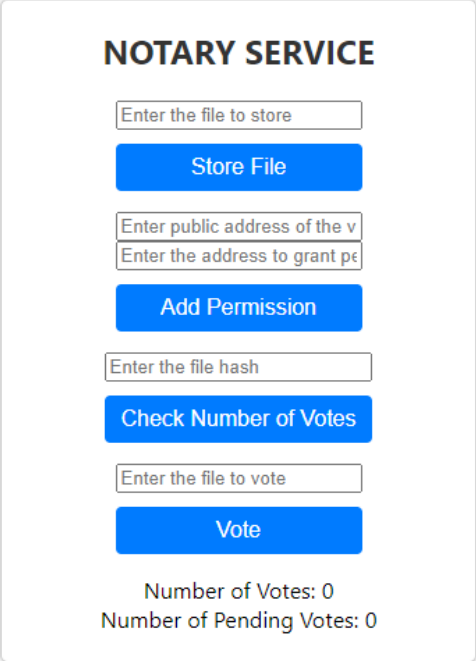
(b)

Figure 5.3: a) Storing a file through Remix IDE b) Metamask activity log of the interaction.

From the two previous figures, it can be depicted that the correct function was called and successfully accepted, meaning that the code in the smart contract function did store the file. It is also possible to verify that the transaction did not cost any tokens and the block number where it is published. With this information, if there is any intruder node validating false information any user can query the validators of the block number and get the list of addresses that were involved in proposing and validating the block.

5.3 Application Testing

After launching the front-end web application and creating an application prepared to interact with the smart contract that was deployed, any authorized user that can connect to the exposed IP and port of the cloud with MetaMask can interact with the application and make usage of the services. The application interface is shown in Figure 5.4.



NOTARY SERVICE

Enter the file to store

Store File

Enter public address of the v
Enter the address to grant pe

Add Permission

Enter the file hash

Check Number of Votes

Enter the file to vote

Vote

Number of Votes: 0
Number of Pending Votes: 0

Figure 5.4: Notary application interface.

Figure 5.5 displays the different screens showcasing the functions being called successfully, with voting rights being granted to another account, thus increasing the number of pending votes and consequently, the number of votes increasing when a vote is cast by an authorized user.

NOTARY SERVICE

Enter the file to store

Store File

0x45b4B33608e7B1B5dC53

d6be96a875d81ea1aee27d0

Add Permission

Enter the file hash

Check Number of Votes

Enter the file to vote

Vote

Number of Votes: 0
Number of Pending Votes: 1
Permission Granted

(a)

NOTARY SERVICE

Enter the file to store

Store File

Enter public address of the v

Enter the address to grant pe

Add Permission

Enter the file hash

Check Number of Votes

d6be96a875d81ea1aee27d0

Vote

Number of Votes: 1
Number of Pending Votes: 0
Vote Casted

(b)

Figure 5.5: a) When a file is successfully stored. b) When a vote is cast by a valid user with permission from the owner.

Lastly, in Figure 5.6 the display screens when the smart contract functions are called but the inputs do not match the function requirements, not changing the blockchain state in any way, are illustrated.

NOTARY SERVICE

d6be96a875d81ea1aee27d0

Store File

Enter public address of the v

Enter the address to grant pe

Add Permission

Enter the file hash

Check Number of Votes

Enter the file to vote

Vote

Number of Votes: 0
Number of Pending Votes: 0
File Already Exists

(a)

NOTARY SERVICE

Enter the file to store

Store File

d6be96a875d81ea1aee27d0

0x45b4B33608e7B1B5dC53

Add Permission

Enter the file hash

Check Number of Votes

Enter the file to vote

Vote

Number of Votes: 1
Number of Pending Votes: 1
You are not the owner of the File

(b)

Figure 5.6: a) When a user tries to store the same file. b) Message when a user who is not the original uploader of a file.

CONCLUSION

This chapter summarizes the development process of the e-notary blockchain network creation, how it benefits collaborative networks, and how feasible it is to create one and make use of its advantages as a notary to increase trust inside the community. Lastly, it presents some of the limitations in the development and challenges in the measurement of the network along with suggestions for future work regarding the unused features that were not studied work and other use cases for blockchain networks to collaborative networks.

6.1 Application Testing

With the broad necessities provided in the introduction and the specific subject of collaborative networks as the response to the new paradigms of organizational collaboration, trust reveals itself as one of the most important aspects for enabling more efficient collaboration. Notaries have long remained unchanged over the years with very basic frameworks that could face security and scalability problems from their centralized nature and have been one of the necessary tools for VBEs to store agreement documents and signatures. This thesis aims to present a proof of concept for a new practical way of creating electronic notaries for collaborative networks that could mitigate some of the challenges that the subject faces.

By using blockchain technology to create a decentralized network that has some degree of central authority and where nodes are all trusted entities, it is possible to provide a new way of storing and voting on contracts when VBE members want to create a virtual organization. Due to the nature of how data is stored in the network by blocks that are validated and proposed by nodes every time data is changed the interaction is recorded on the blockchain, making it easy to know in which exact block it resides and which node was responsible for its

validation, thus offering a quick and efficient way to trace any node trying to publish false information into the chain.

To create this network, a study on the different frameworks, network types, and client tools that could be used to achieve the goal was made. As an outcome, it was decided to use the Ethereum framework with the Hyperledger Besu client. By using these tools, it was possible to create an Ethereum blockchain network that runs an improved, proof-of-authority consensus algorithm where any new node that wishes to partake in the block publishing and validation consensus needs to be approved by the already established participant nodes. This way, only entities trusted by most of the network community are allowed to use their local machines to store and validate the data of the blockchain, which could stimulate the members to create more trust among the rest of the community where they belong.

After the network was created with the desired characteristics for the subject of collaborative networks, a smart contract was deployed in the network so that it could support the functionalities of the e-notary. To facilitate user interaction with the contract a very basic web browser interface was created to help users interact with these smart contract functions. The interface comes pre-loaded with all the tools to interact with the specific instance of the smart contract deployed on the network at the exposed node of the cloud deployment. This way the VBE administrator can allow users to interact with the e-notary services by managing the cloud's firewall settings so users can connect to the node that the application communicates to. This gives ultimate authority over the blockchain services to the administrator while the specific information stored inside the network keeps its decentralized nature as no one can validate false information without most nodes agreeing to do so.

6.2 Limitations & Future Work

This thesis proposed a different solution for one of the tools used inside a VBE, with the main objective to prove how feasible it is to create a network of this kind and the advantages it has over the regular centralized e-notary tools that can bring to the VBE community. However, there are some limitations in this system and suggestions for future work regarding the topic of blockchain applications in the context of collaborative networks.

Due to the limited scope and time frame to develop this project, there were some network features that were not subject to development in this work, such as the questions regarding private interactions within the network. Usually in blockchain networks, such as the one developed in this work, every interaction with the network is publicly visible on the

blockchain network. However, there are some extension tools that Hyperledger Besu implements that can make the transaction information private and only visible to certain participants in the network.

Another feature that was not explored regarding the chosen Ethereum client is related to the permission features that can better handle which addresses can interact with the network, either as a whole or specific per node. This allows for greater scalability of the security for the network as local or global rules could be applied limiting the exposure to potentially malicious users.

The creation of a blockchain network requires some level of infrastructure maintenance among the VBE community, and the more participants inside the network the more secure it gets. As was referred in previous chapters, there are no feasible ways to measure the quality and performance of this network as the purpose of serving as a e-notary for a private community puts very little stress on the network since there will be a low number of transactions per time frame. This leaves the network "open" to serve as other tools for collaborative networks without increasing infrastructure costs. However, there are certain limitations to smart contracts as there is a limit in size the file can get, that is 24KB which is the equivalent of roughly 500 lines of code or one-fifth of the code developed for this work.

BIBLIOGRAPHY

- Afsarmanesh, H., & Camarinha-Matos, L. M. (2005). A framework for management of virtual organization breeding environments. *IFIP Advances in Information and Communication Technology*, 186, 35–48. https://doi.org/10.1007/0-387-29360-4_4
- Afsarmanesh, H., & Camarinha-Matos, L. M. (2009). On the classification and management of Virtual organisation Breeding Environments. *International Journal of Information Technology and Management*, 8(3). <https://doi.org/10.1504/IJITM.2009.024604>
- Afsarmanesh, H., Camarinha-Matos, L. M., & Ermilova, E. (2008). VBE reference framework. In *Methods and Tools for Collaborative Networked Organizations*. https://doi.org/10.1007/978-0-387-79424-2_2
- Afsarmanesh, H., Camarinha-Matos, L. M., & Msanjila, S. S. (2009). On management of 2nd generation Virtual Organizations Breeding Environments. *Annual Reviews in Control*, 33(2). <https://doi.org/10.1016/j.arcontrol.2009.05.007>
- Aggarwal, S., & Kumar, N. (2021). Chapter Eight - History of blockchain-Blockchain 1.0: Currency☆☆Introduction to blockchain. In *The Blockchain Technology for Secure and Smart Applications across Industry Verticals* (Vol. 121).
- Antonopoulo, A. M. (2014). Mastering Bitcoin Unlocking Digital Cryptocurrencies. In *Sermaye Piyasası Kurulu Araştırma Dairesi* (Vol. 9, Issue 1).
- Arrais-Castro, A., Varela, M. L. R., Putnik, G. D., Ribeiro, R. A., Machado, J., & Ferreira, L. (2018). Collaborative framework for virtual organisation synthesis based on a dynamic multi-criteria decision model. *International Journal of Computer Integrated Manufacturing*, 31(9). <https://doi.org/10.1080/0951192X.2018.1447146>
- Baliga, A. (2017). Understanding Blockchain Consensus Models [White Paper]. *Semanticscholar.Org*, 4.
- Bamakan, S. M. H., Motavali, A., & Babaei Bondarti, A. (2020). A survey of blockchain consensus algorithms performance evaluation criteria. In *Expert Systems with Applications* (Vol. 154). <https://doi.org/10.1016/j.eswa.2020.113385>

- ben Salah, S., ben Yahia, W., Ayadi, O., & Masmoudi, F. (2019). An integrated Fuzzy ANP-MOP approach for partner selection problem and order allocation optimization: The case of virtual enterprise configuration. *RAIRO - Operations Research*, 53(1). <https://doi.org/10.1051/ro/2018047>
- Bernal Bernabe, J., Canovas, J. L., Hernandez-Ramos, J. L., Torres Moreno, R., & Skarmeta, A. (2019). Privacy-Preserving Solutions for Blockchain: Review and Challenges. In *IEEE Access* (Vol. 7). <https://doi.org/10.1109/ACCESS.2019.2950872>
- Bhutta, M. N. M., Khwaja, A. A., Nadeem, A., Ahmad, H. F., Khan, M. K., Hanif, M. A., Song, H., Alshamari, M., & Cao, Y. (2021). A Survey on Blockchain Technology: Evolution, Architecture and Security. In *IEEE Access* (Vol. 9). <https://doi.org/10.1109/ACCESS.2021.3072849>
- Camarinha-Matos, L. M. (2014). Collaborative networks: A mechanism for enterprise agility and resilience. In *Proceedings of the I-ESA Conferences* (Vol. 7). https://doi.org/10.1007/978-3-319-04948-9_1
- Camarinha-Matos, L. M., & Afsarmanesh, H. (2005). Collaborative networks: A new scientific discipline. *Journal of Intelligent Manufacturing*, 16(4–5), 439–452. <https://doi.org/10.1007/s10845-005-1656-3>
- Camarinha-Matos, L. M., & Afsarmanesh, H. (2007). A framework for virtual organization creation in a breeding environment. *Annual Reviews in Control*, 31(1). <https://doi.org/10.1016/j.arcontrol.2007.03.006>
- Camarinha-Matos, L. M., & Afsarmanesh, H. (2021). The Evolution Path to Collaborative Networks 4.0. In *IFIP Advances in Information and Communication Technology* (Vol. 600, pp. 170–193). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-030-81701-5_7
- Camarinha-Matos, L. M., Afsarmanesh, H., & Ollus, M. (2008). Ecolead and CNO base concepts. In *Methods and Tools for Collaborative Networked Organizations*. https://doi.org/10.1007/978-0-387-79424-2_1
- Camarinha-Matos, L. M., Ferrada, F., & Oliveira, A. I. (2013). Interplay of collaborative networks in product servicing. *IFIP Advances in Information and Communication Technology*, 408. https://doi.org/10.1007/978-3-642-40543-3_6
- Camarinha-Matos, L. M., Oliveira, A. I., Ferrada, F., Sobotka, P., Vataschinova, A., & Thamburaj, V. (2015). Collaborative enterprise networks for solar energy. *Proceedings of the International Conference on Computing and Communications Technologies, ICCCT 2015*. <https://doi.org/10.1109/ICCCT2.2015.7292726>

- Camarinha-Matos, L. M., Oliveira, A. I., Sesana, M., Galeano, N., Demsar, D., Baldo, F., & Jarimo, T. (2009). A framework for computer-assisted creation of dynamic virtual organisations. *International Journal of Production Research*, 47(17). <https://doi.org/10.1080/00207540902847272>
- Camarinha-Matos, L. M., Silveri, I., Afsarmanesh, H., & Oliveira, A. I. (2005). Towards a framework for creation of dynamic virtual organizations. *IFIP Advances in Information and Communication Technology*, 186. https://doi.org/10.1007/0-387-29360-4_7
- Chohan, U. W. (2018). The Double Spending Problem and Cryptocurrencies. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3090174>
- de Oliveira Albuquerque, R., García Villalba, L. J., Sandoval Orozco, A. L., de Sousa Júnior, R. T., & Kim, T. H. (2016). Leveraging information security and computational trust for cyberse-
curity. *Journal of Supercomputing*, 72(10). <https://doi.org/10.1007/s11227-015-1543-4>
- di Francesco Maesa, D., & Mori, P. (2020). Blockchain 3.0 applications survey. *Journal of Parallel and Distributed Computing*, 138. <https://doi.org/10.1016/j.jpdc.2019.12.019>
- Durugbo, C. (2016). Collaborative networks: A systematic review and multi-level framework. *International Journal of Production Research*, 54(12), 3749–3776. <https://doi.org/10.1080/00207543.2015.1122249>
- Ericson, A., Lugnet, J., Solvang, W. D., Kaartinen, H., & Wenngren, J. (2020). Challenges of In-
dustry 4.0 in SME businesses. *2020 3rd International Symposium on Small-Scale Intelligent Manufacturing Systems, SIMS 2020*. <https://doi.org/10.1109/SIMS49386.2020.9121542>
- Ferrag, M. A., Maglaras, L., Argyriou, A., Kosmanos, D., & Janicke, H. (2018). Security for 4G and 5G cellular networks: A survey of existing authentication and privacy-preserving schemes. In *Journal of Network and Computer Applications* (Vol. 101). <https://doi.org/10.1016/j.jnca.2017.10.017>
- Fiorentino, G., Occhipinti, C., Corsi, A., Moro, E., Davies, J., & Duke, A. (2020). Blockchain: Ena-
bling Trust on the Internet of Things. In *The Internet of Things*. <https://doi.org/10.1002/9781119545293.ch11>
- Folke, C., Hahn, T., Olsson, P., & Norberg, J. (2005). Adaptive governance of social-ecological
systems. In *Annual Review of Environment and Resources* (Vol. 30). <https://doi.org/10.1146/annurev.energy.30.050504.144511>
- Gasparotto, A. M. S., & Guerrini, F. M. (2013). A practical management model for supporting
virtual organizations creation within their breeding environments. *IFIP Advances in Infor-
mation and Communication Technology*, 408. https://doi.org/10.1007/978-3-642-40543-3_15

- Hafid, A., Hafid, A. S., & Samih, M. (2020). Scaling Blockchains: A Comprehensive Survey. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3007251>
- Hartwig, M. (2016). ECDSA Security in Bitcoin and Ethereum : a Research Survey. *Blog.Coinfabrik*.
- Helliar, C. v., Crawford, L., Rocca, L., Teodori, C., & Veneziani, M. (2020). Permissionless and permissioned blockchain diffusion. *International Journal of Information Management*, 54. <https://doi.org/10.1016/j.ijinfomgt.2020.102136>
- Hjalmarsson, F. P., Hreioarsson, G. K., Hamdaqa, M., & Hjalmtysson, G. (2018). Blockchain-Based E-Voting System. *IEEE International Conference on Cloud Computing, CLOUD, 2018-July*. <https://doi.org/10.1109/CLOUD.2018.00151>
- Hsieh, F. S. (2019). Dynamic configuration and collaborative scheduling in supply chains based on scalable multi-agent architecture. *Journal of Industrial Engineering International*, 15(2). <https://doi.org/10.1007/s40092-018-0291-5>
- Hyla, T., & Pejaś, J. (2020). Long-term verification of signatures based on a blockchain. *Computers and Electrical Engineering*, 81. <https://doi.org/10.1016/j.compeleceng.2019.106523>
- Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1). <https://doi.org/10.1007/s102070100002>
- Kim, J. H. (2017). A Review of Cyber-Physical System Research Relevant to the Emerging IT Trends: Industry 4.0, IoT, Big Data, and Cloud Computing. *Journal of Industrial Integration and Management*, 02(03), 1750011. <https://doi.org/10.1142/s2424862217500117>
- Kim, S.-G. (2018). A Study on the Blockchain 2.0 Ethereum Platform Analysis for DApp Development. *The Journal of Korea Institute of Information, Electronics, and Communication Technology*, 11(6).
- Lee Kuo Chuen, D., & Deng, R. (2017). Handbook of Blockchain, Digital Finance, and Inclusion. In *Handbook of Blockchain, Digital Finance, and Inclusion* (Vol. 2).
- Li, Y., Zheng, K., Yan, Y., Liu, Q., & Zhou, X. (2017). EtherQL: A query layer for blockchain system. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10178 LNCS. https://doi.org/10.1007/978-3-319-55699-4_34
- Mulyana, M., & Wasitowati, W. (2021). The Improvement of Collaborative Networks to Increase Small and Medium Enterprises (SMEs) Performance. *Serbian Journal of Management*, 16(1), 213–229. <https://doi.org/10.5937/sjm16-24369>
- Nakamoto, S. (2008). Bitcoin: a peer-to-peer electronic cash system. *Cited On*.

- Novo, O. (2018). Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet of Things Journal*, 5(2). <https://doi.org/10.1109/JIOT.2018.2812239>
- Oliveira, A. I. (2016). *An Environment to Support Negotiation and Contracting in Collaborative Networks*.
- Oliveira, A. I., & Camarinha-Matos, L. M. (2008). Agreement negotiation wizard. In *Methods and Tools for Collaborative Networked Organizations*. https://doi.org/10.1007/978-0-387-79424-2_7
- Pointcheval, D., & Stern, J. (2000). Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3). <https://doi.org/10.1007/S001450010003>
- Politou, E., Casino, F., Alepis, E., & Patsakis, C. (2021). Blockchain Mutability: Challenges and Proposed Solutions. *IEEE Transactions on Emerging Topics in Computing*, 9(4). <https://doi.org/10.1109/TETC.2019.2949510>
- Polyantchikov, I., Shevtshenko, E., Karaulova, T., Kangilaski, T., & Camarinha-Matos, L. M. (2017). Virtual enterprise formation in the context of a sustainable partner network. In *Industrial Management and Data Systems* (Vol. 117, Issue 7). <https://doi.org/10.1108/IMDS-07-2016-0274>
- Rabelo, R. J., Gusmeroli, S., Arana, C., & Nagellen, T. (2006). The ecolead ICT infrastructure for collaborative networked organizations. *IFIP International Federation for Information Processing*, 224. https://doi.org/10.1007/978-0-387-38269-2_47
- Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2). <https://doi.org/10.1145/359340.359342>
- Romero, D., Galeano, N., & Molina, A. (2008). A virtual breeding environment reference model and its instantiation methodology. *IFIP International Federation for Information Processing*, 283. https://doi.org/10.1007/978-0-387-84837-2_2
- Ruotsalainen, P., & Manning, B. (2007). A notary archive model for secure preservation and distribution of electrically signed patient documents. *International Journal of Medical Informatics*, 76(5–6). <https://doi.org/10.1016/j.ijmedinf.2006.09.011>
- Shadi, M., & Afsarmanesh, H. (2014). Behavioral norms in virtual organizations. *IFIP Advances in Information and Communication Technology*, 434. https://doi.org/10.1007/978-3-662-44745-1_5
- Shahnaz, A., Qamar, U., & Khalid, A. (2019). Using Blockchain for Electronic Health Records. *IEEE Access*, 7. <https://doi.org/10.1109/ACCESS.2019.2946373>

- Steiu, M.-F. (2020). Blockchain in education: Opportunities, applications, and challenges. *First Monday*. <https://doi.org/10.5210/fm.v25i9.10654>
- Vacca, A., Di Sorbo, A., Visaggio, C. A., & Canfora, G. (2021). A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges. *Journal of Systems and Software*, 174. <https://doi.org/10.1016/j.jss.2020.110891>
- Vaez-Alaei, M., Deniaud, I., Marmier, F., Gourc, D., & Cowan, R. (2022). A partner selection framework for strategic alliances based on project complexity and partner's past experience. *Enterprise Information Systems*, 16(6). <https://doi.org/10.1080/17517575.2021.1889038>
- Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018). Blockchain Technology Overview. *National Institute of Standard and Technology*.
- Yang, R., Wakefield, R., Lyu, S., Jayasuriya, S., Han, F., Yi, X., Yang, X., Amarasinghe, G., & Chen, S. (2020). Public and private blockchain in construction business process and information integration. *Automation in Construction*, 118. <https://doi.org/10.1016/j.autcon.2020.103276>
- Zhou, Q., Huang, H., Zheng, Z., & Bian, J. (2020). Solutions to Scalability of Blockchain: a Survey. *IEEE Access*, 8. <https://doi.org/10.1109/aACCESS.2020.2967218>

ANNEX A: SOURCE CODE

Solidity source code used for the e-notary smart contract.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.18;

contract FileVoting {
    struct FileInfo {
        bytes32 fileHash;
        address owner;
        uint256 voteCount;
        uint256 votePending;
    }

    mapping(bytes32 => FileInfo) private files;
    mapping(bytes32 => mapping(address => uint256)) private allowedlist;

    event FileAdded(bytes32 fileHash);
    event VoteCast(bytes32 fileHash, address voter);
    event VoterAdded(bytes32 fileHash, address voter);

    function addFile(string memory fileString) public {
        bytes32 _fileHash = stringToBytes32(fileString);
        require(files[_fileHash].fileHash == bytes32(0), "File already
exists");
        files[_fileHash].fileHash = _fileHash;
        files[_fileHash].owner = msg.sender;
        addToAllowedList(fileString, msg.sender);
        emit FileAdded(_fileHash);
    }
}
```

```

function addToAllowedList(string memory fileString, address _person)
public {
    bytes32 _fileHash = stringToBytes32(fileString);
    require(files[_fileHash].fileHash != bytes32(0), "File does not
exist!");
    require(files[_fileHash].owner == msg.sender, "You are not the
owner of this file!");
    allowedlist[_fileHash][_person] = 2;
    files[_fileHash].votePending++;
    emit VoterAdded(_fileHash, _person);
}

function vote(string memory fileString) public {
    bytes32 _fileHash = stringToBytes32(fileString);
    require(files[_fileHash].fileHash != bytes32(0), "File does not
exist!");
    require((allowedlist[_fileHash][msg.sender]) == 2, "You are not
allowed to vote!");

    allowedlist[_fileHash][msg.sender] = 3;
    files[_fileHash].voteCount++;
    files[_fileHash].votePending--;
}

function stringToBytes32(string memory source) internal pure returns
(bytes32 result) {
    bytes memory tempEmptyStringTest = bytes(source);
    if (tempEmptyStringTest.length == 0) {
        return 0x0;
    }

    assembly {
        result := mload(add(source, 32))
    }
}

function bytes32ToString(bytes32 _bytes32) internal pure returns
(string memory) {
    uint8 i = 0;
    while(i < 32 && _bytes32[i] != 0) {
        i++;
    }
    bytes memory byteArray = new bytes(i);
    for (i = 0; i < 32 && _bytes32[i] != 0; i++) {
        byteArray[i] = _bytes32[i];
    }
    return string(byteArray);
}

```

```
    function getFileVoteCount(string memory fileString) public view returns
(uint256) {
        bytes32 _fileHash = stringToBytes32(fileString);
        return files[_fileHash].voteCount;
    }

    function getFilePendingVotes(string memory fileString) public view returns
(uint256) {
        bytes32 _fileHash = stringToBytes32(fileString);
        return files[_fileHash].votePending;
    }
}
```




2023

JOÃO GOULÃO

Electronic Notary Environment for Collaborative Ecosystems