UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Developing Secure Voting Systems Using Blockchain

Hugo Manuel dos Santos da Silva Consciência

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Doutor Vinicius Vielmo Cogo
Eng.º Dino Coutinho

2024

Acknowledgments

I would like to thank professor Vinicius Cogo and my advisor Dino Coutinho for their guidance, availability, and help throughout this thesis development. I would also like to thank Innovation Makers and everyone involved for their support and for allowing me to develop this thesis.

Resumo

A votação é um dos mecanismos fundamentais pelos quais um grupo, como uma assembleia ou um eleitorado, se reúne com o propósito de tomar uma decisão coletiva ou expressar uma opinião, geralmente após discussões, debates ou campanhas eleitorais. É uma peça fundamental da governança democrática, garantindo que a vontade do povo é refletida e levada em consideração na seleção de líderes ou na tomada de decisões importantes.

A votação foi utilizada ao longo da história, desde a Grécia e Roma antigas aos tempos medievais para selecionar governantes. Ao longo do tempo, o conceito de votação permaneceu amplamente inalterado, servindo como um meio para juntar opiniões individuais em decisões coletivas. No entanto, os sistemas de votação utilizados evoluíram, dando origem a vários sistemas que diferem com base em contextos culturais, tecnológicos e políticos.

Os sistemas de votação são assim um dos principais instrumentos de uma democracia representativa, funcionando como o principal mecanismo pelo qual os cidadãos podem expressar as suas opiniões e participar ativamente na escolha dos seus representantes ou líderes. Estes sistemas garantem assim que a vontade dos eleitores seja refletida nas decisões, o que é crucial para a legitimidade e estabilidade de qualquer organização democrática.

Nos últimos anos, os sistemas de votação em papel, maioritariamente utilizados, têm sido cada vez mais criticados pelas suas vulnerabilidades, nomeadamente, fraudes, ineficiências e falta de transparência. A estes problemas, podemos ainda adicionar as dificuldades nas contagens de votos e a lentidão nos processos de verificação, sendo todos estes problemas que circunstâncias que afetam a confiança pública nos processos eleitorais.

Estas preocupações levaram a um grande aumento do interesse em explorar novas tecnologias, de forma a criar ou aprimorar os vários sistemas de votação, tornando-os mais seguros, transparentes e eficientes.

A tecnologia blockchain surge então como uma solução promissora, atendendo às suas características únicas como a descentralização, imutabilidade e transparência, levando desta maneira muitos a considerar a sua aplicação em sistemas de votação.

Porém, antes de explorarmos a tecnologia blockchain, foi necessário analisarmos e avaliarmos os sistemas de votação existentes. No nosso caso, decidimos avaliar e analisar três sistemas de votação distintos (Portugal, Brasil, e Sporting CP), dividindo-os em cinco fases diferentes: Inicialização, Identificação, Votação, Contagem, e Resultados. Com esta divisão foi possível analisar os três sistemas em diferentes fases e encontrar os possíveis problemas associados ao sistema de votação utilizado.

Para além disso, decidimos avaliar as diferentes fases de cada sistema de votação com base em propriedades de segurança importantes como a auditoria, eleitor verificável, voto mutável, integridade, pri-

vacidade, confidencialidade, imparcialidade, eligibilidade, transparência, e impossibilidade da prova de voto.

Várias abordagens foram propostas e desenvolvidas para implementar sistemas de votação com base na blockchain, cada uma com as suas características e vantagens únicas. Benabdallah forneceu-nos uma visão geral de várias soluções inovadoras de e-voting com base na blockchain.

Uma dessas soluções foi proposta por Hardwick, que apresentou um sistema de votação eletrónico com base na blockchain em Ethereum, oferecendo vantagens como a descentralização e desvantagens como escalabilidade.

Outra proposta interessante foi apresentada por David Khoury, que utilizou a plataforma Ethereum para autenticar eleitores através de números de telefone sem a necessidade de um servidor de terceiros. Já Bin Yu propôs um sistema de votação com base em blockchain no Hyperledger Fabric, utilizando o algoritmo de consenso PBFT e técnicas criptográficas de forma a garantir a integridade e anonimato dos votos, destacando-se ainda pela sua escalabilidade alta.

Em contraste, Bartolucci sugeriu o uso da Bitcoin como base de uma solução de e-voting, embora esta tivesse bastante limitações em relação a algumas propriedades de segurança. Por outro lado, na proposta de Ali Al-madani e Hossain Faruk foram utilizadas tecnologias como Ethereum e Hyperledger Fabric para criar sistemas descentralizados, sistemas estes que garantem várias propriedades de segurança como a privacidade, auditoria e elegibilidade através de smart contracts e técnicas biométricas de autenticação.

Depois de avaliarmos os diferentes sistemas de votação e estudarmos as soluções propostas na literatura, começámos então a explorar o uso da blockchain para melhorar a eficiência e a segurança nos sistemas eleitorais, com um foco específico nos sistemas de votação em papel.

De forma aproveitar as várias propriedades da blockchain, como a descentralização e imutabiliade, desenvolvemos um sistema que assegura a verificabilidade e a integridade dos votos, com foco nas eleições locais, reduzindo custos e melhorando a eficiência geral do sistema.

Para criar a nossa blockchain privada, utilizámos o Hyperledger Besu (que é um cliente de Ethereum projetado para facilitar o uso de redes privadas de blockchain) e o Docker (que é um software usado para criar containers virtuais) de forma a rapidamente montar o ambiente da nossa blockchain, sendo que estas plataformas oferecem-nos as melhores condições e orientação para implementar a nossa solução.

Além disso, também estudámos os vários algoritmos de consensus oferecidos pelo Hyperledger Besu, tendo em conta várias propriedades diferentes (como finalidade imediata, número mínimo de validadores, vivacidade, e velocidade) e concluímos que o algoritmo QBFT seria a melhor escolha (algoritmo este que também é recomendado pelo próprio Hyperledger Besu).

De maneira a melhorar a confidencialidade da nossa solução, decidimos utilizar uma Proxy, permitindo que a escolha do eleitor esteja separada da sua identificação (tornando assim impossível fazer qualquer ligação entre o voto e o eleitor). Esta foi uma peça-chave na nossa abordagem, diferenciando-nos das outras soluções e assegurando assim o anonimato do eleitor.

Através da implementação de smart contracts na nossa blockchain privada, foi possível implementar o processo de identificação e autorização do eleitor, e ao mesmo tempo o registo e contagem dos votos, reduzindo também risco de fraude.

Apesar das várias vantagens promissoras, desafios como escalabilidade e outras propriedades de segurança são discutidas e analisadas.

Para implementar a lógica do nosso sistema de votação, utilizámos scripts Node.js, tendo em conta as várias fases do processo (Inicialização, Votação, Contagem).

Para avaliar e testar o desempenho da solução proposta, simulámos uma carga de trabalho de 500 votos e votadores eligíveis, sendo estes automaticamente enviados para o sistema de votação. Desta forma, foi possível obter os tempos de execução das várias fases do sistema e testar a rede blockchain com votos reais, esta que acabou por apresentar bons tempos de execução.

Além disso, também analisámos o desempenho dos diferentes algoritmos de consenso disponibilizados pelo Hyperledger Besu, tendo ainda em conta diferentes métricas de desempenho.

Em conclusão, os resultados desta dissertação demonstram que a tecnologia blockchain pode oferecer bastantes melhorias tanto na segurança como na eficiência dos sistemas de votação, reduzindo custos e facilitando a sua implementação, sem prejuizo, naturalmente, da permanência de alguns desafios. Uma adoção bem sucedida da tecnologia blockchain pode, sem dúvida, transformar os sistemas de votação, tornando-os mais seguros e transparentes.

Palavras chave: Sistemas Votação, Blockchain, Smart Contract, Segurança, Votação Eletrônica

Abstract

The voting systems are one of the most fundamental elements of a representative democracy, serving as the primary mechanism through which people express their opinions. In recent years, concerns about the security and efficiency of the existing voting systems have surged, prompting a growing interest in exploring innovative technologies to make more secure, transparent and efficient voting systems. The recent growth in interest in blockchain technology has led many to consider its application in voting systems. In this thesis, we explore the potential of blockchain to improve the overall efficiency and security of the electoral processes, specifically focusing on local elections and paper-based election systems. By leveraging the decentralized and tamper-proof properties of blockchain, we developed a system that ensures the verifiability and integrity of votes, while also reducing costs and improving the systems overall efficiency. Through the implementation of smart contracts in our private Ethereum blockchain, the process includes voter authentication, secure ballot casting and vote counting, reducing the risk of fraud. Despite promising advantages, challenges such as scalability and other security properties are discussed and examined. We evaluated the proposed model performance, with a workload of 500 eligible voters, and analyzed multiple execution times across different stages of the system, while also analysing the performance of the multiple consensus algorithms provided by Hyperledger Besu.

Keywords: Voting Systems, Blockchain Technology, Smart Contract, Security, Electronic Voting

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Voting is one of the fundamental mechanisms by which a group, such as a meeting or an electorate, convenes together for the purpose of making a collective decision or expressing an opinion usually following discussions, debates, or election campaigns [45]. It is a cornerstone of democratic governance, ensuring that the will of the people is reflected and taken into account in the selection of leaders or important decisions.

Voting was used as early in history as ancient Greece and ancient Rome, and throughout the Medieval period to select rulers. Over time, the concept of voting has remained largely unchanged as it serves as a means to aggregate individual opinions into collective decisions. However, the voting systems used have evolved, giving rise to various voting systems that differ based on cultural, technological, and political contexts [22], [44].

Nowadays, voting systems all around the world differ greatly in terms of technology, security, and usability. Even while electronic voting is well-established in many nations, manual and paper-based procedures are still widely used in others. Nonetheless, worries about inefficiencies, lack of transparency, and voter fraud continue to be widespread. Such problems motivate organizations to find new and innovative technologies that can improve their current systems. This opportunity for improvement offers a chance to incorporate blockchain technology into voting processes [29].

In the last decades, most voting systems have remained unchanged, falling behind the advancements seen in other areas. The reliance on outdated technologies and manual processes not only hinders the efficiency of elections but also poses security risks and limits the ability to adapt to the evolving needs of modern society.

Addressing these problems with an innovative technology such as blockchain can completely transform the way that voting systems are implemented. Much research has been done to implement blockchain technology in online voting systems, but in our approach, we aim to develop a physical electronic system that utilizes blockchain technology to secure and improve the voting system, while also discarding the problems associated with online voting methods. Although implementing such technology presents many challenges, this work has the motivation to find solutions and develop a better system to implement, with the main focus being local elections.

1.2 Problem

The vulnerability of the present voting systems to many types of fraud and manipulation is a serious issue. Electronic and online voting systems are vulnerable to hacking and other fraudulent acts, while paper-based systems are vulnerable to ballot stuffing, tampering, counting problems and many other issues. Concerns exist over the electoral process's lack of security and confidence, which, in some cases, might raise questions about the validity of election results. In addition, many of these voting systems are also inefficient, often requiring significant resources in terms of time, human resources, and financial investment. Outdated technologies, manual processes, and time-consuming administrative procedures can contribute to delays in election results and increased costs. Ensuring trust in the voting system is crucial for the legitimacy of democracy. In response to these challenges, Innovation Makers decided to step in and propose this dissertation theme, exploring how the blockchain technology introduces an appealing way to tackle these concerns and addressing the need for more reliable, secure, and efficient voting systems.

1.3 Goals

In this work, we aim to develop a conceptual architecture for our blockchain-assisted voting system, as well as implementing it in a working prototype. By analyzing current voting methods' advantages and disadvantages, we intend to identify key security properties that can be improved. Our solution will be based on leveraging blockchain technology to address the limitations and security problems found in current voting systems, with a primary focus on local elections. Through the use of smart contracts, we plan to enhance the functionality and automation of the blockchain voting system prototype, while also reducing the potential for human error, and improving the overall efficiency and security of the voting process.

1.4 Contributions

This work, in partnership with Innovation Makers, is associated with the exploration of an innovative implementation for enhancing the security and efficiency of voting systems by leveraging blockchain technology. The contribution that this work intends to make is to utilize blockchains to enable the creation of secure in-person electronic voting systems, significantly reducing vulnerabilities related to fraud and manipulation while lowering the operational costs associated with traditional electoral processes.

To achieve these objectives, an in-person blockchain-assisted voting system was proposed, integrating blockchain technology into key stages of the electoral process (from voting to counting and auditing). This ensures enhanced transparency and security across the entire system.

Additionally, the system could benefit from leveraging Innovation Makers' HEFESTO product, explained in detail in appendix B, which focuses on security through biometric client identification, specifically facial recognition. By integrating HEFESTO into the voting system, voter identification security would be greatly enhanced, adding an extra layer of protection against identity fraud and ensuring that only legitimate voters participate.

This work can also serve as a foundation for future contributions that may explore expanding blockchain use in voting systems or scaling up the proposed solution for real-world applications.

1.5 Document Structure

The remainder of this document is organized as follows: Chapter 2 outlines how some current voting systems function, analyzing and evaluating their strengths, weaknesses, and areas for improvement. It also covers essential concepts such as blockchain technology, smart contracts, and other relevant concepts to help understand the main focus of this work. Additionally, the chapter explores the state of the art in blockchain voting systems, discussing the related work. Following up, Chapter 3 dives into a deeper analysis of the presented problem and also discusses a possible plan to solve it. The implementation process of our proposed solution is thoroughly explained in Chapter 4, followed up by an evaluation of its performance in Chapter 5. Finally, Chapter 6 concludes the document with key takeaways and insights.

Chapter 2

Background and Related Work

In this section, we analyze and evaluate three existing voting methods (Portugal's, Brazil's, and Sporting CP's), provide a brief overview of the main concepts used in our work, and discuss the related literature.

2.1 Existing Elections Voting Systems

Before proposing a possible solution, it is important to understand how current election voting systems function and what can be improved. We analyze three different election voting systems: Portugal's, Sporting CP's, and Brazil's, and divided them into five different key phases [5]. Although this comparison is not exhaustive, it is insightful and considers a broad variety of election types, encompassing different scales (national and local) and methods (paper-based and electronic).

1. ***Initialization (Phase 1):*** in this phase the organization prepares for the elections;
2. ***Identification (Phase 2):*** on the day of the election, the voters identify themselves before voting;
3. ***Voting (Phase 3):*** once identification is completed, voters choose one or several candidates according to the voting rules and then cast the vote;
4. ***Counting (Phase 4):*** when the elections end (becomes impossible to change or add votes), the counting process begins;
5. ***Results (Phase 5):*** the results are announced in detail and accessible to all;

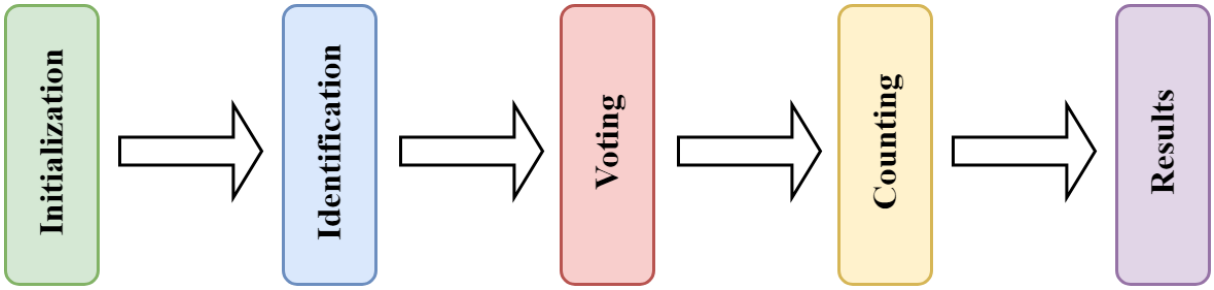


Figure 2.1: Voting System Phases

2.1.1 Portugal and Sporting CP Voting Systems

Despite being uncorrelated, Portugal's and Sporting CP's voting systems are nearly identical, as they are structured in a very similar way and differing only in a few aspects. Both Sporting CP and Portugal use a paper-based voting system with similar processes for identification and voting. However, the key difference is that in Sporting CP's system, the weight of a vote can increase depending on how many years the voter has been a member, whereas in Portugal's voting system, each vote always counts as one. Because of this, we decided to incorporate both of them in the same analysis. Figure 2.2 provides a visual representation of the main players and their interactions with the different phases of the voting systems.

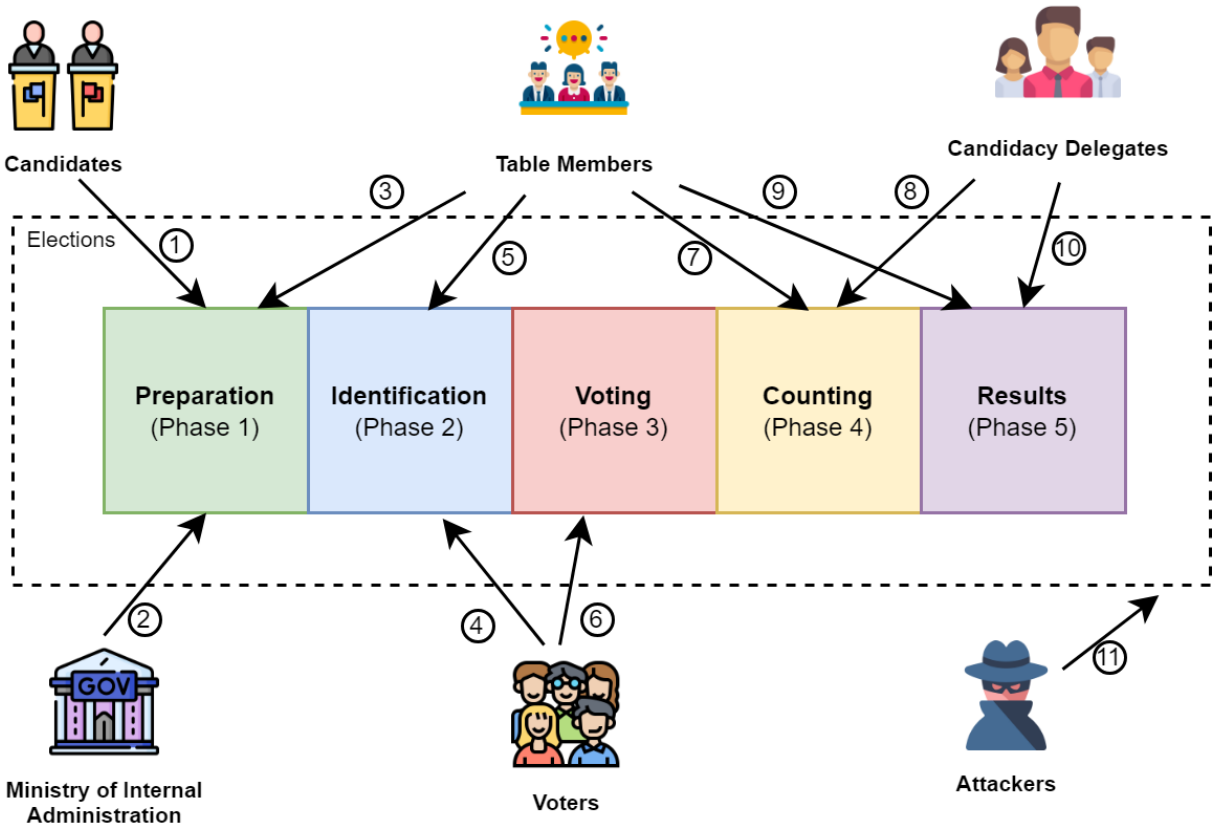


Figure 2.2: Portugal and Sporting Voting Ecosystem

A. Portugal's Voting System

Scenery: A board (each board has five members, a president, a vice-president, and three vocals), a ballot box, and voting booth(s).

1. Initialization: Initially, the candidates (1) announce their participation in the elections. The Ministry of Internal Administration (2) (responsible for the elections) organized everything prior to the election day. In the election day's morning, the board members (3) of each voting place start the preparations. The vote discharge of the anticipated votes is made (the board must verify if the voter is properly registered in the voter registration and make the corresponding discharge on the electoral journal) and also verifies the total number of voters that will be voting that day.

2. Identification: The eligible citizens (4) go to the voting location and present their identification

documents at the voting table. A table member (5) then verifies the citizens identity. After identifying themselves, a blank ballot is given by the table president.

3. Voting: The citizen (6) goes to the voting chamber where s/he fills in the vote and bends it four times. S/he then returns to the voting table and gives the filled ballot to the table president who then proceeds to insert the vote in the ballot box (in certain types of elections it is the citizen that inserts the vote in the ballot box).

4. Counting: The board members (7) count the votes following a specific protocol that has security rules (e.g., none of the board members that will be handling/counting votes should have writing material in their possession). The candidacy delegates (8) (from each candidacy) can be present and observe the counting of the votes at the voting place.

5. Results: After the votes are counted and all verifications are complete, the president of the board (9) announces the results. The candidacy delegates (10) can then request proof of the results, and if they find any irregularity, file a complaint to the Superior Court. It is important to note that the contesting phase is out of the scope of this analysis and will not be discussed.

B. Sporting CP's Voting System

Scenery: An identification board, voting booth(s), ballot box, and ballot box board.

1. Initialization: Initially, the candidates (1) announce their participation in the elections. The club (2) starts preparing everything the days before the elections. In the election day's morning, the table members (3) complete final preparations.

2. Identification: The voters (4) go to the identification table, where they provide their membership card and ID to the poll worker. The poll worker (5) then identifies and verifies the eligibility to vote in the election. After identifying the voter, the poll worker provides the ballot to the member.

3. Voting Procedures: The member (6) goes with the ballot to the booth where he fills out the ballot, and then places the vote in the corresponding ballot box (the process of placing the vote in the ballot box is watched by several poll workers present).

4. Counting: After the election day is over, the table workers (7) start counting the votes manually.

5. Results: When the count is over, the results are announced (9) by the president of the general assembly.

Possible Problems:

There are many problems with Portugal's and Sporting CP's voting systems, most of them related to human factors, which can be exploited by attackers (11). These are the most relevant problems we identified:

i. Pregnant Ballot Box: Before voting began, completed ballots were placed in the ballot box, that is, the ballot box that should have been empty arrived at the voting table with votes inside.

ii. Ballot Box Replacement: Before arriving at the voting locations, the official ballot boxes were replaced by others full of completed ballots. At the end of the voting day, the ballot boxes with the votes collected during the day are replaced by others with illegal votes.

iii. Ant Vote: The voters receive the ballot from the table, enter the voting booth, and instead of filling out the ballot and depositing it, keep the blank ballot and place any piece of paper in the ballot box. The

fraud organizer, who is outside the polling place, receives the official blank ballot, selects the desired candidate, and hands it to another voter. This voter takes the completed ballot with him, receives a new blank one (from the voting table), deposits the completed one, and brings the blank one outside to the fraud organizer, repeating this process indefinitely.

iv. Invalid Vote/Filled Vote: During the vote counting, votes are filled in by poll workers, making the vote invalid or in the case of a blank vote, making it valid for the chosen candidate.

v. Natural Disasters/Human Factor: Despite the low probabilities, there is always the risk of the results/records being destroyed due to natural disasters (earthquakes, tsunamis) or due to human intervention (vandalism).

2.1.2 Brazil's Voting System

Scenery: Voting booths with Electronic Voting Machines (EVMs) and board members.

Electronic Voting Machine (EVM): The Electronic Voting Machine is made up of two terminals:

- The polling station terminal, where the voter is identified and authorized to vote, including by biometric means;
- The voter terminal, where the vote is recorded;

1. Initialization: The first stage of preparation involves installing the operating system, libraries, programs, and electoral data. The second and final stage is completed by carrying out several tests to prove the correct functioning of the EVM. After preparing the EVMs, they are sealed with special seals produced by "Casa da Moeda" (Treasury), whose chemical properties prevent any attempted tampering.

After that, the EVMs are stored in a location designated by the Regional Electoral Court (TRE), so that they can be transported to the voting locations on the eve of the elections. Any attempt to use them before this will be in vain, as the EVMs have a system that only allows it to be used at the time scheduled for voting. Finally, a very important security measure is that the EVM is at no time connected to any type of external communication network (this way the data cannot be intercepted or suffer any external attack from hackers).

2. Identification: The voter is identified by the poll worker using an identification document with a photo, they can also carry a Voter ID (not mandatory) but have the electoral session number that facilitates identification and search of your electoral session.

3. Voting: After being identified by the poll worker, the voter is also identified using biometric identification. The biometric identification process takes place after checking the voter's documents at the polling station, to release the vote in the electronic voting machine. Each candidate is assigned a number. The voter must enter the desired code into the EVM whose sole function is to count the votes.

4. Counting: When the elections day ends, the presidents of the respective polling stations must enter a password in the ballot box to close the vote. The equipment then issues five copies of the ballot box, which informs the total number of votes cast by candidates, political parties, blank votes, invalid votes, in addition to the total number of votes received at the polls.

One copy is fixed at the respective section, which allows the results to be checked in that location; three other copies are added and sent to the respective electoral registry; and the last copy is delivered to party

representatives or inspectors. It is possible to print more copies, if necessary.

After that, the ballot boxes and their memory cards are sent for authenticity testing at the Regional Electoral Court (TRE). At this stage technicians analyze whether or not the 30 layers of protection [21] placed on the data have been breached (e.g., physical seals, version control systems, various software tests, multifaceted cryptography, software and hardware signatures, etc.) and also check the ballot box log, which works like a "black box".

After proving that there was no data breach, it is transmitted through a VPN to the Superior Electoral Court (TSE). In the TSE, all votes sent by the Regional Electoral Court (TRE) are added together. The calculation is done publicly, using a supercomputer from the American technology company Oracle, the data X8 Full Rack. Any citizen can access the TSE platform and follow the counting live.

5. Results: It is possible to have the results in a few hours after voting ends. After completing the count of all votes, parties and supervisory entities can request all files from the electronic voting machines and the database to check the result.

Possible Problems:

These are the most relevant problems we identified with Brazil's voting system:

i. Vandalism/Malfunction: During elections, electronic voting machines may be subject to vandalism (e.g., glue on the keyboard keys, paint on the screen, etc.) or have technical issues, which leads to the need to replace/repair the machine in question [27][16];

ii. Fraudulent Poll Worker: In some cases, table workers voted in place of people who did not vote. These people would justify their absence later, and the fraudulent poll worker would face an investigation [38].

Contingency Procedures:

a. Contingency Voting Machine: There are contingency voting machines that are used to replace those that are defective during voting. When replaced, the flash card and voting disk are transferred from the defective voting machine to this voting machine, thereby migrating the votes already registered (the encryption key is the same among all voting machines [17], so there is no need to transfer it).

b. Contingency Flash Card: This procedure is used when the contingency ballot box was used but did not solve the problem. It consists of using a flash card previously prepared for this function.

2.2 Voting Systems Evaluation

A voting system must be reliable whether it is paper voting or online voting. It must be secure enough to withstand any attacks but also transparent enough to proof everything is working as expected.

In order to evaluate the security and effectiveness of these voting systems, we conducted an analysis of key security properties for each phase of the voting systems.

2.2.1 Security Properties

We analyzed each voting system phase based on the following security properties [5]:

1. ***Audit***: The system provides proof that all its parts work as expected;
2. ***Voter Verifiable***: A user must be able to verify that his vote has been correctly cast and counted;
3. ***Vote-Alterable***: Gives a user the ability to alter his own vote (e.g., in case of coercion, a coercer cannot be sure that the coerced citizen will not change their vote later);
4. ***Integrity***: Guarantees that data must never be altered during transmission or processing, either intentionally or accidentally;
5. ***Privacy***: It is necessary to protect users from having their personal information leaked;
6. ***Confidentiality***: A voter should never be identified from his vote;
7. ***Fairness***: The election results are not counted in real time. This way, no one should know which candidate is ahead in the election and other voters will not be influenced;
8. ***Eligibility***: Only eligible voters can cast a vote;
9. ***Transparency***: The process must be transparent;
10. ***Receipt-Freeness***: The elector cannot construct a receipt to prove to a third person that he voted for a certain candidate;

2.2.2 Evaluation

In Table 2.1 we provide a thorough evaluation of the security properties of each voting system, using a graded scale of 3 stars, 2 stars, or 1 stars. A 3 star indicates that the security property is strongly assured, a 2 star denotes partial assurance, and a 1 star signifies that the property is not adequately addressed. To facilitate a quick distinction between properties graded with 2 star and 1 star, we highlighted them with yellow and red, respectively.

Each numbered property corresponds to a detailed explanation of the respective security property within a specific phase of the voting process, meaning that the same property might have different explanations depending on the phase.

The following paragraphs provide a detailed explanation of Table 2.1 points:

In **(1) Audit**: (Portugal) (Brazil) (Sporting CP) all voting systems demonstrate strong auditability. In Portugal and Sporting CP, detailed documentation and open scrutiny facilitate comprehensive audits. Brazil employs extensive testing procedures and independent audits.

In **(2) Integrity**: (Portugal) (Sporting CP) During the preparations for the elections, if the rules are followed, no data is alterable. The download of anticipated votes is made by the members of the board (in the electoral journal) and no votes are counted/opened. But due to human error/intentions, a voter that already voted might not be crossed out from the electoral journal, making it possible for that voter to vote more then one time.

(Brazil) No data can be alterable. The electronic voting machines are sealed with a special seal to stop/prevent any violation attempt.

Table 2.1: Voting Methods Framework with Security Properties

Phases	Portugal Voting	Brazil Voting	Sporting CP Voting
Initialization	(1)Audit-⋆⋆⋆ (2)Integrity -⋆⋆ (3)Confidentiality-⋆⋆⋆ (4)Eligibility-⋆⋆⋆ (5)Transparency-⋆⋆⋆	(1)Audit-⋆⋆⋆ (2)Integrity-⋆⋆⋆ (5)Transparency-⋆⋆⋆	(1)Audit-⋆⋆⋆ (2)Integrity -⋆⋆ (3)Confidentiality-⋆⋆⋆ (4)Eligibility-⋆⋆⋆ (5)Transparency-⋆⋆⋆
Identification	(6)Privacy-⋆⋆⋆ (4)Eligibility-⋆⋆⋆	(6)Privacy-⋆⋆⋆ (4)Eligibility-⋆⋆⋆	(6)Privacy-⋆⋆⋆ (4)Eligibility-⋆⋆⋆
Voting	(7)Voter-Verifiable; -⋆⋆ (8)Integrity-⋆⋆⋆ (9)Confidentiality -⋆⋆ (10)Receipt-Freeness -⋆ (11)Vote Alterable -⋆	(7)Voter-Verifiable -⋆⋆ (8)Integrity-⋆⋆⋆ (9)Confidentiality-⋆⋆⋆ (10)Receipt-Freeness -⋆ (11)Vote Alterable -⋆	(7)Voter-Verifiable -⋆⋆ (8)Integrity-⋆⋆⋆ (9)Confidentiality -⋆⋆ (10)Receipt-Freeness -⋆ (11)Vote Alterable -⋆
Counting	(1)Audit-⋆⋆⋆ (12)Integrity -⋆⋆ (13)Fairness-⋆⋆⋆ (5)Transparency-⋆⋆⋆	(1)Audit-⋆⋆⋆ (12)Integrity-⋆⋆⋆ (13)Fairness-⋆⋆⋆ (5)Transparency-⋆⋆⋆	(1)Audit-⋆⋆⋆ (12)Integrity -⋆⋆ (13)Fairness-⋆⋆⋆ (5)Transparency-⋆⋆⋆
Results	(14)Audit-⋆⋆⋆ (5)Transparency-⋆⋆⋆	(14)Audit-⋆⋆⋆ (5)Transparency-⋆⋆⋆	(14)Audit-⋆⋆⋆ (5)Transparency-⋆⋆⋆

In **(3) Confidentiality**: (Portugal) (Sporting CP) the anticipated votes cannot be linked to their voters because the votes are never opened/counted until the end of the election day.

In **(4) Eligibility**: (Portugal) (Sporting CP) The board verifies if the voter is properly registered in the voter registration journal. Also during the download of anticipated votes, the board verifies if the anticipated voters are properly registered in the voter registration.

In **(5) Transparency**: (Portugal) (Brazil) (Sporting CP) the entire process is open and transparent.

In **(6) Privacy**: (Portugal) (Brazil) (Sporting CP) The information of the voter is only shown to the board members identifying the voter.

In **(7) Voter Verifiable**: (Portugal) (Brazil) (Sporting CP) The voter can verify that his vote has been correctly cast (the ballot is put inside the ballot box or the voting machine confirms the vote) but cannot verify that it has been correctly counted.

In **(8) Integrity**: (Portugal) (Sporting CP) The ballot data is not alterable during the cast procedure of the ballot. It is bent 4 times and placed inside the ballot box.

(Brazil) The vote data cannot be manually alterable during the processing of the vote.

In **(9) Confidentiality**: (Portugal) (Sporting CP) The ballot doesn't have anything that can identify the voter but the voter can write something in the ballot that would identify him (note that this would make the ballot invalid).

(Brazil) The voter cannot be identified from his vote and the voter cannot do anything to be identified in his vote (as it is electronic).

In **(10) Receipt-Freeness**: (Portugal) (Sporting CP) (Brazil) In this process the elector can construct proof that he voted for a certain candidate. The voter can record the process of filling the ballot/voting machine as proof (but note that this is illegal).

In **(11) Vote Alterable**: (Portugal)(Brazil)(Sporting CP) The voter cannot alter his vote after submitting it. One time vote.

In **(12) Integrity**: (Portugal)(Sporting CP) During the counting of votes, no members of the board handling the votes have writing material, but this doesn't fully assure the integrity of the votes as it relies on the rules being followed.

(Brazil) The voting machines and the computers (that count all votes from all machines) guarantee that data is never altered during transmission or processing of the results. Multiple authenticity tests are also done on the voting machines before sending the local results to the TSE (Superior Electoral Court).

In **(13) Fairness**: (Portugal)(Brazil)(Sporting CP) The election results are not counted in real time. They are only counted when the voting phase is over.

In **(14) Audit**: (Portugal)(Sporting CP) The system provides proof that all parts work through documentation. When results are published, political parties/entities can file a request for a review of the results.

(Brazil) After the counting phase, parties and supervisory entities can request all files from electronic voting machines and database to check the results.

2.3 Blockchain Technology

The evaluation of these common voting systems revealed both advantages and disadvantages in their ability to safeguard the integrity of democratic processes.

As technological advancements continue to emerge, there is growing interest in exploring alternative approaches to voting, particularly those that leverage the use of blockchain technology.

2.3.1 Blockchain Definition

A blockchain is a distributed network that is leveraged for various purposes. It is an immutable ledger technology where the recorded information is open and can be viewed by everyone. It does not involve any central authority to monitor the regular flow of the network, making it less prone to attacks.

The immutable nature of blockchain also ensures that every single device connected to the network contains a copy of the contract and of all the blocks with all the transactions, thus securing a backup version.

To summarize, the key elements of a blockchain are: *Distributed Ledger Technology*, *Immutable Records*, and *Smart Contracts*.

2.3.2 Smart Contracts

Smart contracts are programs that reside within decentralized blockchains and are executed pursuant to triggered instructions. A smart contract acts in a similar way to traditional agreement but negates the necessity for the involvement of third party. After being deployed, smart contracts are capable of initiating their commands automatically, thus eliminating the involvement of a regulatory body. In the context of a blockchain voting system, the use of smart contracts is particularly crucial, as they will automate key aspects of the voting procedure.

2.3.3 Blockchain Taxonomy

There are many different classes of blockchains, each one with its benefits and drawbacks. The most used and well-known types are:

A. Permissionless Blockchain

A permissionless blockchain is open and accessible to everyone. Anyone can participate in the network and validate transactions, making them inherently transparent and decentralized. Bitcoin and Ethereum are prime examples of permissionless blockchains. Table 2.2 shows its benefits and drawbacks.

Table 2.2: Benefits and drawbacks of a permissionless blockchain

Benefits	Drawbacks
✓ Decentralization potential	✗ Scalability challenges
✓ Group consensus	✗ Bad actors
✓ Ease of access	✗ Excessive transparency

B. Permissioned Blockchain

A permissioned blockchain restricts access to authorized participants, limiting participation to those with specific permissions. Governance is centralized, with decision-making power vested in a select group of validators. Transparency can be limited, but network upgrade time and scalability are often greatly improved. Table 2.3 shows its benefits and drawbacks.

Table 2.3: Benefits and drawbacks of a permissioned blockchain

Benefits	Drawbacks
✓ Scalability	✗ Centralization
✓ Easy customization	✗ Vulnerabilities to Attacks
✓ Limited transparency	✗ Censorship Risk
✓ Invitation-only entry	

2.3.4 Blockchain Consensus Mechanisms

Consensus mechanisms are the protocols or algorithms used to achieve agreement among distributed participants in a blockchain network. They determine how transactions are validated, ordered, and added to the blockchain. There are countless consensus mechanisms, each one with its nuances. These are the four most commonly used protocols:

a. Proof-of-Work (PoW)

PoW is a consensus mechanism that requires participants, called miners, to solve computationally intensive puzzles to validate transactions and add new blocks to the blockchain. The first miner to solve the puzzle is rewarded with cryptocurrency. This process ensures that the blockchain remains secure and difficult to tamper with, as it would take a significant amount of computing power to control a majority of the network. However, PoW can be energy-intensive and may not scale well as blockchains grow.

b. Proof-of-Stake (PoS)

PoS is a consensus mechanism where participants, called validators, are selected based on their stake in the network. Staking involves locking up cryptocurrency as collateral (it can be unlocked back). The more cryptocurrency a validator stakes, the higher their chance of being selected to validate the next block. PoS is less energy-intensive than PoW and can potentially handle higher transaction volumes. However, it also has the potential for centralization as the concentration of wealth in the network increases.

c. Practical Bizantine Fault Tolerance (PBFT)

PBFT [39] is a consensus mechanism that achieves consensus among a subset of network participants known as the "ordering service nodes." These nodes are responsible for ordering and validating transactions. PBFT ensures that as long as a majority of the ordering service nodes are honest and can communicate reliably, the network can reach a consensus on the order of transactions. It is a variant of the Byzantine Fault Tolerance (BFT) algorithm, tailored for blockchain systems.

d. Proof-of-Authority (PoA)

PoA is a consensus mechanism suitable for private and permissioned blockchain which relies on a number of pre-chosen authoritative nodes called the validators. It is designed to optimize the PoS mechanism and be used, ideally, in permissioned networks. Instead of choosing block miners on the basis of their stakes in cryptocurrency tokens, PoA selects a small group of authorities as transaction validators by their identity or reputation staked in the network.

Existing validators can vote to add additional users into the authority group. A PoA-based system also rewards authorities for certifying and ordering transactions to incentivize honest behavior in providing service and moderating the network.

2.4 Related Work

The development of secure, transparent voting systems has been a longstanding challenge for communities all around the world. With the advancements in blockchain technology, many researchers have

begun exploring its potential to address these challenges and revolutionize the voting process.

Various approaches have been proposed and developed to implement blockchain-based voting systems, each with its unique characteristics and advantages. Benabdallah *et al.* [5] provides an insightful overview of the most revealing e-voting solutions based on blockchain technology.

One such solution is proposed by Hardwick *et al.* [28], as it presents an e-voting scheme based on blockchain technology on the Ethereum blockchain (that supports smart contracts and uses proof-of-stake for consensus), providing a degree of decentralization and putting as much control of the process, as possible, in the hands of the voters. It also guarantees privacy and supports custom encryption methods. It addresses various security properties such as auditability, voter-verifiable, vote-alterable, integrity, privacy, confidentiality, fairness, eligibility, and transparency. However, despite promising advantages, smart contracts on Ethereum suffer from scalability issues that could be solved by deploying them on a less decentralized network. The solution employs cryptographic techniques, including blind signatures, to protect voter anonymity. Additionally, since the e-voting application is based on the Ethereum blockchain, it utilizes an Ethereum-specific Hash function that can be added to other encryption algorithms to enhance security.

Still on the Ethereum environment, the solution shown in David Khoury *et al.* [34] proposes a decentralized voting platform based on the Ethereum Blockchain for online voting. They used Solidity, a smart-contract oriented programming language for writing the smart contracts. In their approach, users are authenticated through their mobile phone number without the need for a third party server. This proposal addresses multiple security properties such as transparency, eligibility, confidentiality, privacy, integrity, voter-verifiable and auditability.

Another interesting solution is shown by Bin Yu *et al.* [48], as it proposes a blockchain-based voting system that utilizes smart contracts on the Hyperledger Fabric blockchain framework (that can hold up to 100000 transations per second). This system uses practical Byzantine fault tolerance (PBFT) for consensus and addresses important security properties including auditability, voter-verifiable, integrity, privacy, confidentiality, fairness, eligibility, transparency, receipt-freeness, and scalability. It also uses cryptographic techniques like linkable ring signatures and homomorphic encryption to ensure voter anonymity while preserving the integrity of votes.

In contrast to the previous solutions, Bartolucci *et al.* [4] proposes using Bitcoin as a blockchain implementation for e-voting. This implementation employs secret sharing to ensure voter anonymity and homomorphic encryption to enable the tallying of votes without revealing individual votes. The system also uses a different consensus mechanism, proof-of-work, to ensure that votes are registered correctly. While this solution presents some differences, it falls short of addressing almost all security properties, meeting only the eligibility requirement.

Expanding on more blockchain solutions, in Ali Al-madani *et al.*[2] the authors present a decentralized electronic voting system based on smart contracts by using Ethereum's blockchain technology. The researchers developed a smart contract and deployed it to a local blockchain, that worked like a network and a decentralized database all in one for storing voter's accounts, votes and candidates details. They concluded that the application is able to overcome the limitations and security issues of the centralized voting systems. Regarding the security properties, it addresses only the integrity property, leaving other key security requirements unmet.

To conclude, in Hossain Faruk *et al.*[30], the authors propose a web-based online voting system that

utilizes blockchain technology and biometric idenfication techniques to improve the security, privacy, and transparency of elections. Their system also utilized Hyperledger Fabric (as in Bin Yu *et al.* [48]) to store and maintain a secure and tamper-evident voting record. They also used biometric modalities such as fingerprint and facial recognition for dual-factor voter authentication and security. They concluded that their implementation offered a seamless, secure, and transparent experience for the voters. This solution addresses many important security properties, such as transparency, integrity, scalability, privacy and elegibility.

Table 2.4 provides an overview of the key security properties (previously explained in 2.2.1 Security Properties) addressed by the solutions presented, with the addition of the 'In-Person Voting' property, which checks whether or not the solution is focused on an in-person voting system.

Table 2.4: Key security properties addressed by the related works

	Hardwick *et al.* [28]	**Khoury** *et al.* [34]	**Bin Yu** *et al.* [48]	**Bartolucci** *et al.* [4]	**Al-madani** *et al.*[2]	**Faruk** *et al.*[30]
Audit	✔	✔	✔	✗	✗	✗
Voter Verifiable	✔	✔	✔	✗	✗	✗
Vote-Alterable	✔	✗	✗	✗	✗	✗
Integrity	✔	✔	✔	✗	✔	✔
Privacy	✔	✔	✔	✗	✗	✔
Confidentiality	✔	✔	✔	✗	✗	✗
Fairness	✔	✗	✔	✗	✗	✗
Eligibility	✔	✔	✔	✔	✗	✔
Transparency	✔	✔	✔	✗	✗	✔
Receipt-Freeness	✗	✗	✔	✗	✗	✗
In-Person Voting	✗	✗	✗	✗	✗	✗

Chapter 3

The Proposed Blockchain-Assisted Voting System

In this chapter, we present a possible solution for a blockchain-based voting system. Our proposition aims to address the problems of current voting systems, particularly in the context of Portugal and Sporting CP, as it could be implemented in various voting systems all over the country.

3.1 Conceptual Solution

3.1.1 Overview

In our proposed solution, the aim of the blockchain-based voting system is to address common issues such as ballot tampering, voter fraud, and lack of transparency, while improving and enhancing its security, transparency, efficiency and integrity. This solution is particularly aimed at improving voting systems in Portugal such as football clubs, political parties and other smaller organizations.

Similarly to the voting systems discussed in Chapter 2, our conceptual solution is also divided into five different phases: Initialization, Identification, Voting, Counting, and Results (as shown in Figure 2.1).

3.1.2 Design Principles

- The system is designed to be permissioned and decentralized, reducing possible failure points.
- Transparency is achieved by making the blockchain view public for everyone.
- Consensus algorithms and cryptographic techniques are used to ensure security.
- Voter privacy and anonymity are preserved while maintaining the integrity of the voting process.

3.2 Blockchain Properties

To develop a reliable and secure blockchain voting solution, careful selection of the blockchain type and its inherent properties is key.

3.2.1 Permissioned Blockchain

For a voting system solution, we believe a permissioned blockchain would be more suitable, as it is in a government (or organization) setting (having its level of centralization associated with it). But, it does not need to follow the usual characteristics associated with a permissioned blockchain (e.g., it could be completely transparent and open for public view).

Hyperledger [25] is an open-source, global ecosystem for enterprise-grade blockchain technologies hosted by The Linux Foundation. Its networks are permissioned, meaning all participating member's identities are known and authenticated.

Initially, Hyperledger Sawtooth [36] was considered due to its permissioned nature and implementation. However, because it was archived and had no support available anymore, Hyperledger Besu [6] was ultimately chosen, considering its enterprise-grade features and support for Ethereum-based smart contracts. This choice facilitates the implementation and aligns better with the system's needs.

3.2.2 Hybrid Approach

An interesting approach could be having the voting system in a permissioned blockchain for vote processing and publishing aggregate results on an independent permissionless blockchain. This guarantees verifiability and transparency without requiring the permissioned blockchain to remain active all the time for result checking.

3.2.2.1 Permissionless Blockchain

Permissionless blockchains offer great solutions for data storage and integrity. Because these networks are public and always running (maintained by people all around the world), we can take advantage of this to publish a summary of the results as well as a pointer and a hash of the complete results stored on IPFS. This approach reduces the costs associated with maintaining data safety and accessibility for anyone. The most well-known examples of permissionless blockchains include Bitcoin, Ethereum, XRP, Binance Smart Chain and Polygon, among many others.

3.3 System Architecture

3.3.1 Architecture Components

- **Voting Machines**: Deployed at voting locations to handle voters identification and vote casting. These machines, developed by Innovation Makers would be integral to the voting process;
- **Permissioned Blockchain**: Manages voter eligibility and vote registering. The network executes a verification process to check if the voter is eligible to vote and records their vote on the blockchain after being sent from the proxy;
- **Proxy**: Receives votes from the machine, it then mixes and randomizes the votes order and sends them to the blockchain (it is assumed that the proxy is trusted).
- **Counting Machine**: Counts votes from the permissioned blockchain, then sends the full results to an off-chain storage.

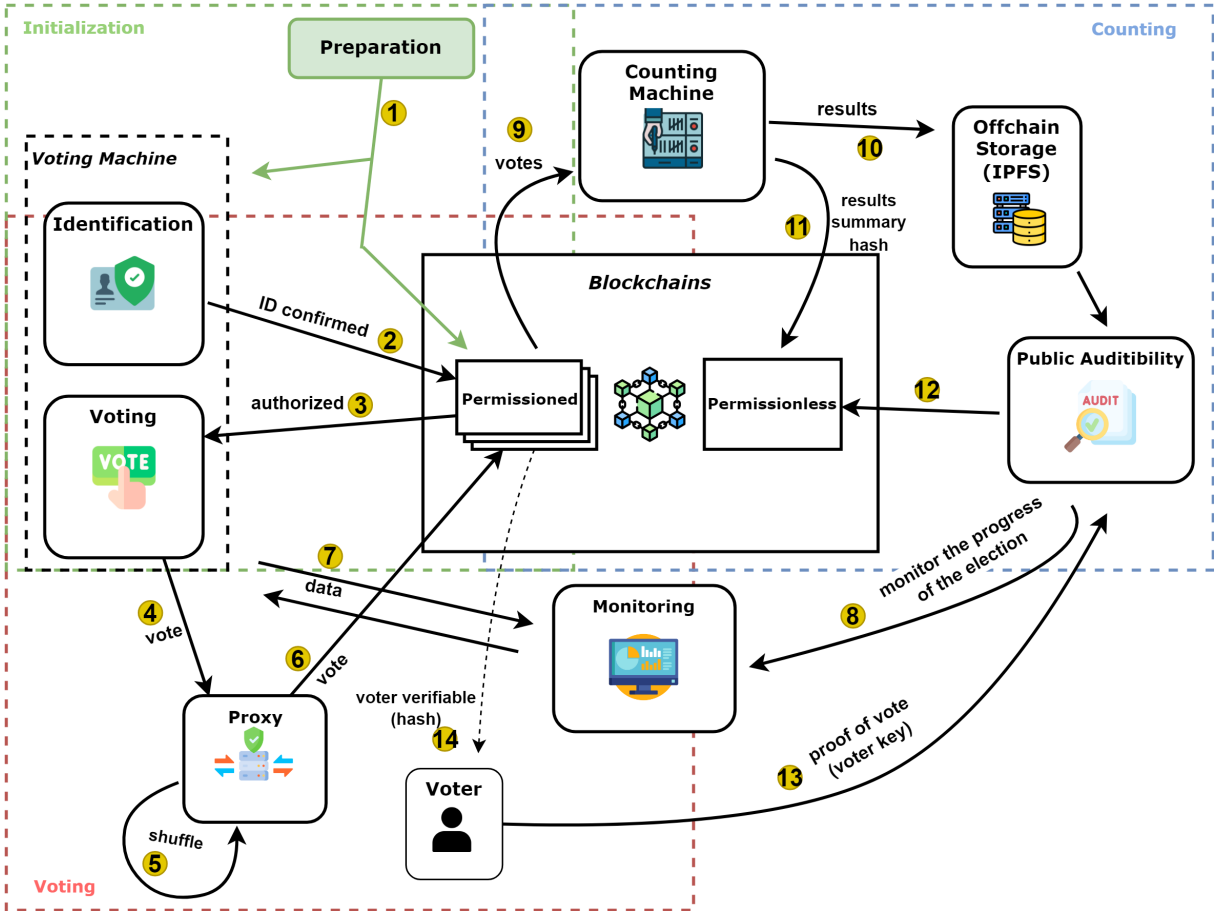


Figure 3.1: Solution Architecture

- **Permissionless Blockchain**: It is used to store the aggregate results of the elections, as well as a pointer or hash of the full results;
- **Public Interface**: Allows the public to follow the election progress. Real-time updates provide information about the number of voters, registered voters and system status.

3.3.2 Architecture Description

Figure 3.1 depicts a possible architecture for our proposed system. It is divided into three main phases:

1. Initialization Phase

Initially, to (1) prepare the voting system, the smart contract is defined, the voting machines are set up in the voting locations, the permissioned blockchain is set up (smart contract is also deployed), and all other necessary preparations are done (e.g., list of eligible voters is registered in the smart contract).

2. Voting Phase

When the elections start, the voter provides identification to the machine (which will be an in-house hardware solution from Innovation Makers), which then (2) confirms the identity of the voter, after that the voter chooses the candidate they wish to cast their vote on. The permissioned network then executes a verification process to check if the voter is eligible to vote (checks the eligible voters list stored in the deployed smart contract). If the verification is successful (the voter is eligible to vote), it registers the identification of the voter in the blockchain and (3) signals the machine to release the vote.

After the vote is released the machine then (4) sends the vote to the proxy, which then (5) mixes and randomizes the votes order with other votes (making it impossible to link the vote to the identification transaction). After mixing the votes, the proxy (6) sends the votes to the permissioned blockchain.

While the elections take place, the voting machines also (7) provide real-time updates (e.g., how many voters voted; the number of votes registered in the blockchain; heartbeats every x seconds) enabling the public to (8) track the progress of the elections and also check if all the machines are working correctly.

3. Couting Phase

When the voting period ends, a counting machine (validator) (9) verifies and tallies the votes for each candidate, and when the count is over it (10) stores the complete results of the elections in an off-chain storage (e.g., IPFS) and (11) publishes a hash of the results with the results summary on a permissionless blockchain (which can then be (12) accessed by the public).

In case it is needed, voters (13) can verify and prove their participation by providing their (14) vote hash, which is publicly verifiable.

3.4 Improvements Introduced

By implementing a blockchain voting system, some security properties could be improved as well as the overall efficiency and security of the system.

3.4.1 Security Enhancements

In Table 2.1, we evaluated the current voting systems based on important security properties. We believe some of those properties could be improved, such as:

- ***Voter Verifiable*: 2 stars -> 3 stars** (voters could verify if their vote was counted by viewing the blockchain transactions history and could also use their transaction hash to check if the transaction is in the blockchain);

- ***Confidentiality***: (Portugal) **2 stars -> 3 stars** (the voter would not be able to do anything in order to be identified on his vote (e.g., write or draw something on the ballot));

On the other hand, security properties such as ***Receipt-Freeness*** (the citizen could still record themselves voting) and ***Vote Alterable*** (people would only be able to vote once) would not be improved by our solution (due to the in person nature of our solution).

3.4.2 Addressed Issues

Other possible problems, related to Portugal and Sporting CP voting systems, would also be solved such as:

1. **Pregnant Ballot Box**: Solved by electronic recording of votes;
2. **Ballot Box Replacement**: Prevented by the immutable blockchain;
3. **Ant Vote**: Mitigated by the electronic casting of votes;
4. **Invalid/Filled Vote**: Addressed by the electronic votes;

3.4.3 Security Analysis

While blockchain technology provides significant security and transparency in the voting process, there are still some specific features that require improved security measures. As discussed previously in 3.4.1 Security Enhancements, crucial aspects such as *Voter Verifiability* and *Confidentiality* can be further improved, and ensuring the maintenance of *Fairness* is essential. While the nature of blockchain inherently improves Voter Verifiability, enhancing Confidentiality requires additional security measures. By using a proxy, we can dissociate votes from voter ids, as the proxy randomly shuffles the votes before they are sent to the blockchain. To maintain the Fairness of elections and enhance overall security and confidentiality, votes are encrypted before being sent to the proxy. This ensures that the votes remain confidential throughout the process.

Chapter 4

Implementation

In this chapter, we explain the technical implementation for our presented solution of the blockchain-based voting system. This chapter details the steps taken to build the system, encompassing the setup and configuration of the Hyperledger Besu Network (including the consensus algorithm), the development and deployment of the smart contract, the integration of the voting system and proxy servers, the interactions between all the election process and the execution of rigorous testing procedures.

4.1 Blockchain Network Setup

In order to implement our solution as efficiently and simply as possible, we decided to use the Quorum Developer Quickstart Tutorial [9] provided by Hyperledger Besu. This streamlined approach minimizes configuration, as it allows to setup a simulation of a private network using Docker for enterprise applications, that require secure and high-performance transaction processing.

The Quorum Quickstart facilitates the creation of the blockchain configuration files, enabling a fast setup of a private network with four Besu IBFT 2.0 validator nodes and a non-validator node (RpcNode), simulating a base network, like shown in Figure 4.1. More nodes can be added to the network, but for our implementation we consider one node to be sufficient.

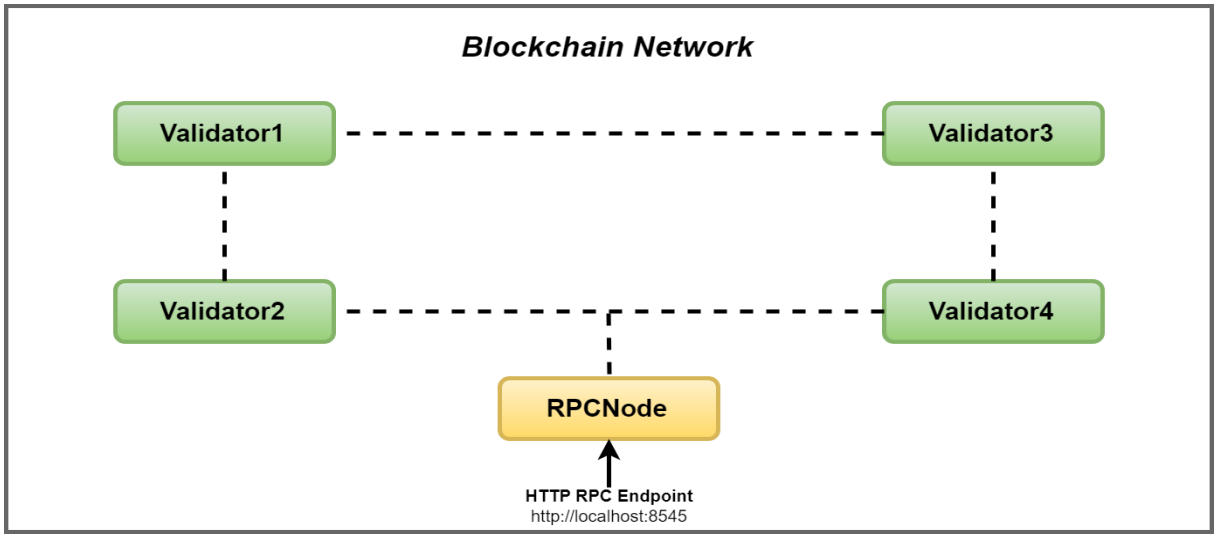


Figure 4.1: Blockchain Network

4.1.1 Consensus Algorithm

The Quorum Developer Quickstart offers a bit of flexibility in terms of consensus algorithms, allowing us to choose the one that best suits our network's needs. Besu implements the QBFT, IBFT 2.0, and Clique proof of authority (PoA) consensus protocols. PoA consensus protocols work when participants know each other and there is a level of trust between them. For example, in a permissioned network.

In QBFT, IBFT 2.0, or Clique, a group of nodes in the network act as validators (QBFT and IBFT 2.0) or signers (Clique). The existing nodes in the signer/validator pool vote to add nodes to or remove nodes from the pool.

4.1.1.1 Properties (QBFT vs IBFT 2.0 vs Clique)

When comparing QBFT, IBFT 2.0, and Clique we must consider four key properties:

- Immediate finality
- Minimum number of validators
- Liveness
- Speed

a) Immediate Finality

QBFT and IBFT 2.0 have immediate finality. Once a block is validated and added to the chain, it´s considered finalized and irreversible. There are no forks or chain reorganizations.

Clique does not have immediate finality. Implementations using Clique must be aware of forks and chain reorganizations occurring.

b) Minimum Number of Validators

To be Byzantine Fault tolerant, QBFT and IBFT 2.0 require a minimum of four validators. With less then 4 validators, the system becomes vulnerable to malicious actors who could disrupt consensus.

Clique can operate with a single validator but operating with a single validator offers no redundancy if the validator fails. Clique with multiple validators achieves BFT through redundancy. While highly unlikely to fail under realistic conditions, there´s no mathematical proof of BFT. In contrary, QBFT and IBFT2.0 offer a provably secure BFT.

c) Liveness

Clique is more fault tolerant than QBFT and IBFT 2.0. Clique tolerates up to half of the validators failing (n/2). QBFT and IBFT 2.0 networks require greater than or equal to two-thirds (>=2/3) of validators to be operating to create blocks.

For example, an QBFT and IBFT 2.0 network of:

- Four to five (4-5) validators tolerates one unresponsive validator
- Six to eight (6-8) validators tolerates two unresponsive validators.

Networks with three or less validators can produce blocks but do not guarantee finality when operating in adversarial environments.

d) Speed

In Clique networks, reaching consensus and adding blocks is faster. The probability of forks increases as the number of validators increases.
For QBFT and IBFT 2.0, the time to add new blocks increases as the number of validators increases.

4.1.1.2 Consensus Algorithm Choice

After analysing and comparing the three consensus algorithms, QBFT, IBFT 2.0, and Clique, based on their key properties, we came to the conclusion that both QBFT and IBFT 2.0 emerge as the best options to consider when choosing the consensus algorithm, as they are very similiar. For this work, we decided to go with QBFT as it is the recommended enterprise-grade consensus protocol for private networks by Hyperledger Besu.

4.2 Smart Contract

In this section we dive into the technical details of the smart contract implemented. We will explain its deployment and structure.

4.2.1 Deployment Process

In order to deploy our smart contract, we created a *node.js* script that executes the deployment. By connecting to the *RPCNode* using the *ethers.js* library, the script establishes a communication channel with the blockchain. It then utilizes the rpc account's credentials to interact with the network and deploy the compiled contract. The script waits for the deployment to finalize and retrieves the contract's address on the blockchain for future interactions.

4.2.2 Structure

The smart contract is written in *Solidity*, as it is one of the most used smart contract programming languages for building smart contracts on the Ethereum blockchain. The main purpose of our smart contract is to store the encrypted votes and voters ids, as well as verifying voters eligibility to vote.

The smart contract is divided into three phases (Figure 4.2): Initialization, Voting and Counting. This means that certain functions can only be called when the assigned phase is active, otherwise denying their execution. There are also restrictions to change the active phase, meaning that the Initialization phase can only change to the Voting phase, and the Voting phase can only change to the Counting phase.

It is also possible to limit who can access which functions (by storing the authorized addresses in the smart contract), limiting even more the access to the smart contract. In our case, we've only implemented the possible restricted access to the smart contract owner's address (the address which deployed it).

The smart contract is mainly composed by seven key functions (the code of these functions is shown in appendix B):

- ***statusVoting()***: Changes the status from Initializing to Voting;

- ***statusCounting()***: Changes the status from Voting to Counting;
- ***initializeVotersList()***: Initializes a map with the eligible voters value set to 1;
- ***addId()***: Checks if id is eligible to vote, if eligible stores id in the ids list and returns true;
- ***addVote()***: Adds encrypted vote to the votes list;
- ***getIdsList()***: Returns the stored list of ids;
- ***getVotesList***: Returns the stores list of encrypted votes;

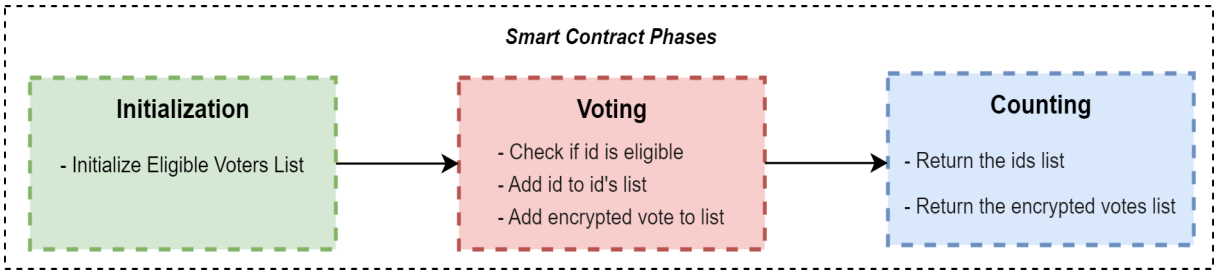


Figure 4.2: Smart Contract Phases

4.3 Proxy

In this chapter we explain the implementation of the Proxy, a critical component designed to enhance voter confidentiality in our election system. It is important to note that, in our implementation, we assume the proxy is trusted (i.e., it never fails), as it can be easily replicated and/or use digital signatures for verification.

4.3.1 Structure

The proxy's main purpose is to act as an intermediary between the voting machine and the blockchain. When a voter registers their vote, two transactions will be made to the blockchain: one containing the ID of the voter, and the other with the vote itself. To make it impossible to link the transactions with IDs to the transactions with votes (which would compromise voter anonimity), we decided to implement a "middleman" for the vote transactions.

While the transaction with the ID is registered immediately after the voter confirms their vote, the vote itself goes to the proxy. Within the proxy (as shown in Figure 4.3), the vote is kept safe until one of two conditions is met: either a predefined number of votes (X) is received, or a timeout (of X time) is triggered.

After this, the votes are randomly shuffled (making their position order random) and sent one by one to the blockchain. This process ensures that the order of the ID transactions cannot be correlated with the order of the vote transactions, therefore preserving voter anonymity.

Appendix C shows a code snippet of the Proxy script implementation.

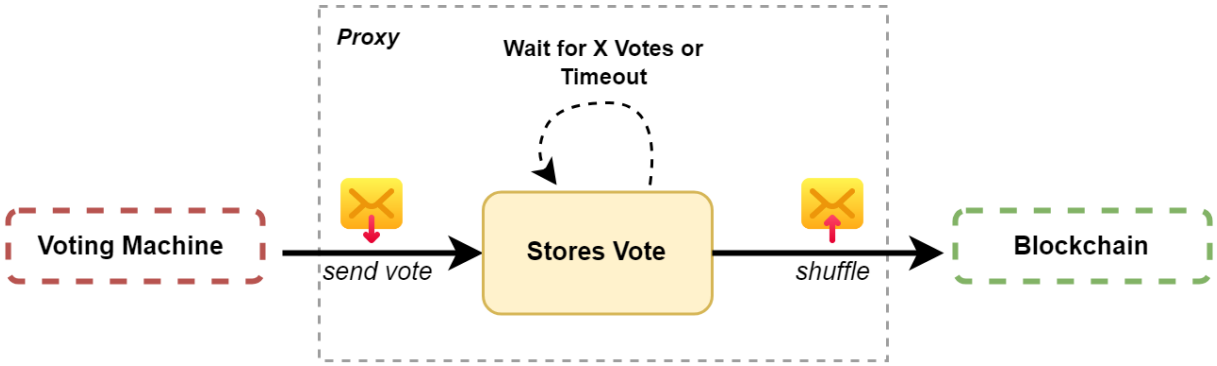


Figure 4.3: Proxy

4.4 Election Process

To implement the logic of our voting system and interact with the blockchain, we utilized Node.js scripts. Appendix D contains code snippets from the four main executable scripts, along with a more detailed explanation of their functionality:

- ***SC_deploy.js***: Deploys the compiled smart contract on the blockchain;
- ***SC_InitializeEligibleVoters.js***: Initializes the eligible voters list in the smart contract;
- ***Voting_Process.js***: Starts the voting process and launches the proxy;
- ***Counter.js***: Retrieves, decrypts, and counts all the votes registered on the blockchain;

As detailed in Subsection 4.2.2, our voting process consists of three stages: *Initialization*, *Voting*, and *Counting*.

During the *Initialization* phase, we begin by deploying the smart contract on the blockchain, as it is the core component of the voting process. Once deployed, the smart contract address can either be manually placed on the scripts or we can make use of an implemented function that automatically retrieves the address of the first smart contract deployed. After getting the smart contract address, we initialize the eligible voters list in the smart contract. This ensures that only voters on the list can vote; otherwise, their attempts will be rejected.

After initializing the eligible voters list, we proceed to the *Voting* phase. Here, the proxy is started, and the system is prepared to receive votes. When a vote is received, the voter's ID is verified to ensure eligibility. If eligible, the vote is encrypted and then sent to the proxy. To encrypt the votes we decided to use the *Cypheriv* encryption method from the *crypto* library with the "aes-256-cbc" algorithm, as it is simple to implement and provides sufficient security for our Proof-of-Concept (PoC) implementation. When the voting phase finishes, we enter the *Counting* phase. In this stage, all votes registered on the blockchain are decrypted (using the same key used in the encryption) and counted, returning the final results of the elections.

4.5 Implementation Challenges

During the implementation, we encountered different challenges, one of which was a random error that occurred when executing transactions on the blockchain (registering the Ids and Votes on the smart contract). This error, named 'REPLACEMENT_UNDERPRICED', was particularly hard to solve because

it occurred randomly, making the debbuging harder. Because of this error, not all eligible votes were registered on the blockchain, which would be a serious flaw in our implementation.

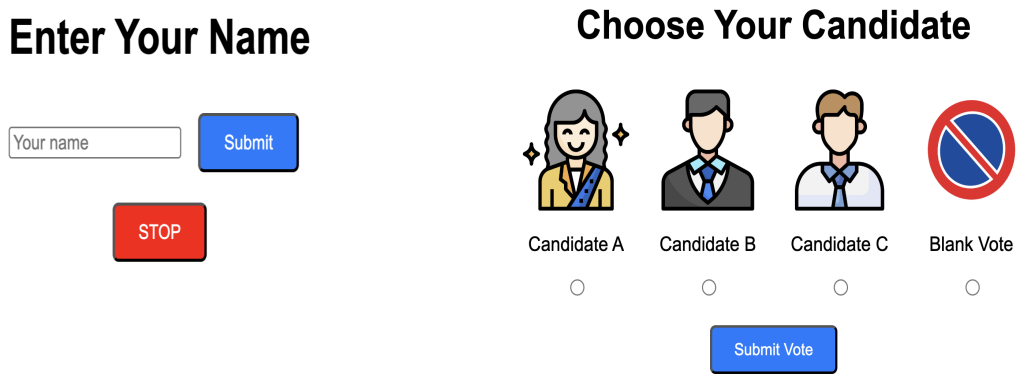
To address this issue, we implemented a solution involving a 'try' and 'catch' mechanism. This allowed us to catch the specific error and automatically retry sending the transactions through a recursive loop, fixing the issue and ensuring that all of the transactions were successfully registered in the blockchain.

4.6 Voting System API

In an effort to facilitate the registration of votes, we implemented an API for our voting system using FastAPI. This addition enabled us to create a web-based interface that allowed us to register the votes manually and manually end the voting phase, making it more efficient and user-friendly during testing.

When a vote is registered or the stop button is pressed, an event is sent to the voting system, that then receives and processes the information.

In Figure 4.4(a) we can see the interface where we input the voters id, as well as the stop button to stop the election. And Figure 4.4(b) shows the interface where we choose the candidate we wish to cast the vote for.



(a) FastAPI HTML Id Registration

(b) FastAPI HTML Vote Registration

Figure 4.4: FastAPI HTML of the Voting System

4.7 Voting System Overview

To overview our private network, the Quorum Developer Quickstart provides multiple tools that enables us to check multiple stats related to our implemented private blockchain, such as the Grafana dashboard, the Prometheus dashboard and the Web Quorum Block Explorer. In our case, we decided to use the Web Quorum Block Explorer as it offered the information we wanted in an user-friendly interface.

Figure 4.5 depicts the Web Quorum Block Explorer interface, where we can see, in real-time, the mined blocks, registered transactions, and deployed smart contracts. And figure 4.6 shows the Nodes interface, where we can observe the overall statistics of the network (such as the number of mined blocks, the number of peers, and the current status of the network).

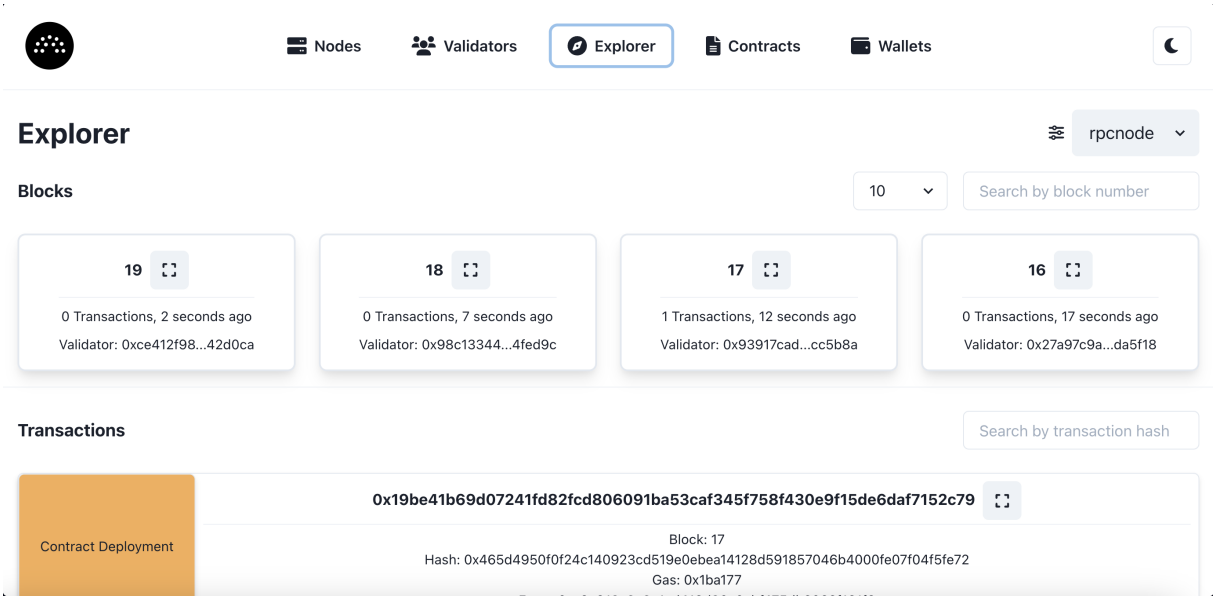


Figure 4.5: Quorum Explorer View 1

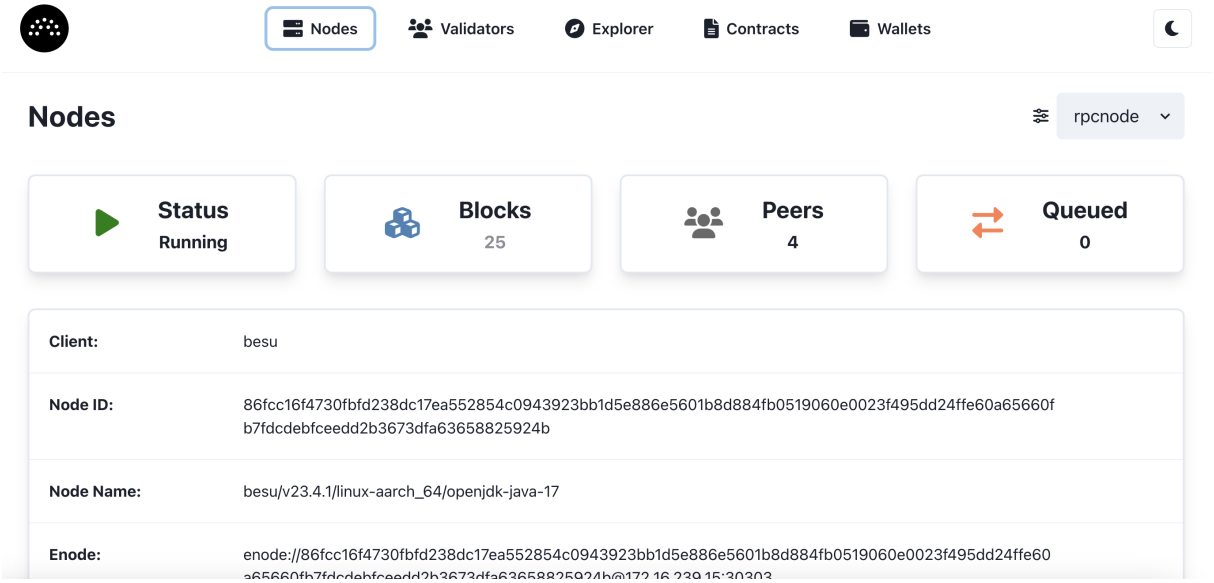


Figure 4.6: Quorum Explorer View 2

Chapter 5

Evaluation

Until now we have presented our detailed solution proposal and its implementation. In this chapter, we address the evaluation of our voting system implementation. The main focus of this evaluation is to explain and understand the performance, and associated costs of the system. We will investigate the efficiency of the system in terms of execution times for various processes, performance of the different types of consensus algorithms, and analyze the associated costs of implementing a blockchain-based voting system. By doing so, we aim to provide a comprehensive understanding of the practical viability and potential areas for improvement of our voting system.

5.1 Experimental Environment

In this section, we detail the hardware and software environment used for evaluating our blockchain voting system implementation. The tests performed in this chapter were executed in one machine.

Machine Used

- MacBook Air M1:
  - Processor: Apple M1 Chip
  - Memory: 8 GB
  - Operating System: macOS Sonoma (version 14.5)

Software Used

- Docker:
  - Version: 4.29.0
  - Used for containerizing and managing the Hyperledger Besu Docker Image to run the private network (the implementation of Docker is also explained in the Quorum Developer Quickstart Tutorial Stable 24.9.1 [9]).

5.2 Research Questions

The evaluation focuses on addressing the following research questions:

- **Q1:** How efficient is the voting system in terms of execution time for each stage of the process (Initialization, Voting and Counting)?
- **Q2:** What are the associated costs of our blockchain-based voting system solution, including smart contract deployment, data storage and transaction costs?
- **Q3:** How does the system ensure security and confidentiality while maintaining performance?

5.3 Performance

In this section, we present an analysis of the time profiling of our scripts, supported by box plots graphs, as well as a comparison between the three available consensus algorithms performance (Clique, QBFT, and IBFT 2.0).

5.3.1 Workload

In order to evaluate our blockchain-based voting system, we tested the network with a workload of 500 eligible votes and voters. This workload was designed to ensure that none of the votes would be rejected by the network, allowing us to focus on the performance of the system under optimal conditions.

To facilitate the process of registering 500 votes, instead of manually sending the votes through the API web page, we implemented a function that would automatically start sending votes to the voting system, sending votes at a rate of 1 vote every 5 seconds.

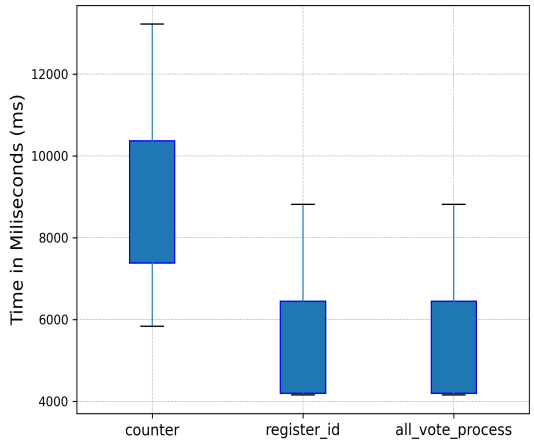
An important factor to take into account is that in our implementation, only the RpcNode interacts with the blockchain, ence the 5 second delay between votes.

5.3.2 Time Profiling Analysis

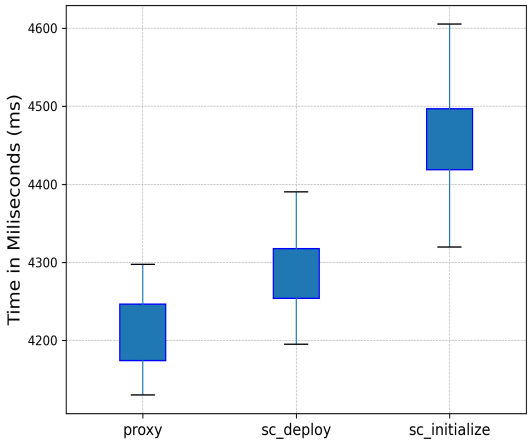
The time profiling was conducted on important scripts and functions within the voting system execution. The following box plots (5.1(a), 5.1(b), and 5.1(c)) illustrate the distribution of execution times (in milliseconds) across different stages. With the higher execution times being in 5.1(a), and the lower execution times in 5.1(c).

Initialization Phase

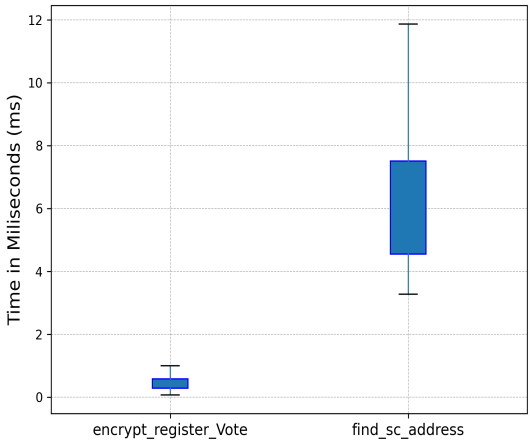
1. **sc_deploy**: In the Box Plot Graph 5.1(b), the box plot illustrates the deployment time for the smart contract on our private network. The minimum deployment time being around 4200 ms, while the maximum time being just under 4400 ms;
2. **find_sc_address**: In the Box Plot Graph 5.1(c), the box plot illustrates the time to find the address of the deployed smart contract in the private network (this applies to all the processes that interact with the smart contract). The minimum time is just above 3 ms, while the maximum time is close to 12 ms;
3. **sc_initialize**: In the Box Plot Graph 5.1(b), the box plot represents the time it takes to initialize the eligible voters in the smart contract. The minimum time being approximately 4300 ms, and the maximum time being just under 4600 ms;



(a) Profiling Box Plot Graph I



(b) Profiling Box Plot Graph II



(c) Profiling Box Plot Graph III

Figure 5.1: Profiling Box Plot Graphs of important scripts and functions

Voting Phase

4. **register_id**: In the Box Plot Graph 5.1(a), the box plot shows the time for the 'register_id' process during the voting phase, being the time it takes for the voting process to register the voters Id in the blockchain. The minimum time is around 4000 ms, and the maximum time is approximately 9000 ms;

5. **encrypt_register_vote**: In the Box Plot Graph 5.1(c), the box plot illustrates the time for the 'encrypt_register_vote' process, being the time spent encrypting the vote and sending it to the proxy. The minimum time being near 0 ms, and the maximum time being just under 2 ms;

6. **all_vote_process**: In the Box Plot Graph 5.1(a), the box plot indicates the time for the entire voting process (from receiving the vote, to registering the voter id and sending the vote to the proxy). The minimum time is around 4000 ms, and the maximum time is approximately 9000 ms;

7. **proxy**: In the Box Plot Graph 5.1(b), the box plot depicts the time for the proxy to register a single vote in the blockchain. The minimum time being around 4100 ms, and the maximum time being close to 4300 ms;

Counting Phase

8. **counter**: In the Box Plot Graph 5.1(a), the box plot represents the time for the counter to retrieve all the votes, decrypt them, and count them (returning the results). The minimum time being around 6000 ms, and the maximum time being approximately 13500 ms;

By looking at the execution times of the various processes, we can see that some of them have similar times, such as 'register_id' and 'all_vote_process'. This is because inside the voting process, the execution that takes most of the time is the 'register_id', while the 'find_sc_address' and the 'encrypt_register_vote' have really small execution times (both under 15 ms). Then we have the 'proxy', 'sc_deploy' and 'sc_initalize' processes, all of these three processes interact with the blockchain by either deploying or calling functions of a smart contract. The times vary depending on the complexity of the execution (changing states of lists, returning information). For last we have the 'counter' process, this is the execution that takes more time on average, as it gathers all of the votes registered on the blockchain transactions, and then decrypts and counts them to return the results. Overall, the entire voting process presents good execution times, indicating efficient performance and reliability.

5.3.3 Consensus Algorithms Performance Comparison

In analyzing the performance of the Hyperledger Besu proof-of-authority (PoA) consensus algorithms, we benefited from an analysis conducted by Caixiang Fan *et al.* [23], which compared the performance of Clique, QBFT and IBFT 2.0. The cited study provides a detailed comparative analysis, saving us significant time.

5.3.3.1 Performance Metrics

The study evaluated the consensus algorithms between four key performance metrics, i.e., throughput, latency, resource consumption, and scalability, here are the definitions of the four key metrics:

- **Throughput** is the number of successfully committed transactions or query operations per second. The transaction throughput is expressed as transactions per second (TPS) at a network size. Query throughput is expressed as transactions per second (TPS) from a single node.
- **Latency** is the amount of time taken for a transaction's effect to be usable across the network. The measurement includes the propagation time and the consensus time. Query latency is the time between when the read request is submitted and when the reply is received.
- **Resource consumption** is the CPU and memory (RAM) utilization in each node container.
- **Scalability** is the ability to support increasing network participants or computation resources of nodes. This measurement is indicated by the throughput and latency when the network or node size increases.

5.3.3.2 Key Findings

1. **Consensus Performance**:
    - Figure 5.2 shows the transaction performance of Besu against the three proof of authority algorithms, QBFT, IBFT 2.0, and Clique, under varying send rates and different types of

transactions (Open transactions performs one single write operation to the blockchain while Transfer transactions perform two write operations).

- **Throughput**: We can see that these three consensus algorithms have very close throughput under low workload, e.g., 400 req/s. But in higher send rates, QBFT subtly starts to have a better performance, while Clique starts to have a slightly lower throughput then the other two.
- **Latency**: In terms of latency, Clique has a relatively lower transfer latency, while the other two have similar results.
- One thing to note is that the open transaction ends up having a better performance in terms of both throughput and latency in all three consensus algorithms.

2. **Scalability**:

- **QBFT** showed better scalability compared to IBFT 2.0 and Clique. It scaled efficiently up to 14 validators without noticeable performance loss, although the performance started to decline beyond this point. Figure 5.3(a) and Figure 5.3(b) show the scalability of different consensus algorithms at 1,000 req/s open transaction rate.
- **IBFT 2.0** is officially claimed to handle up to 30 validators with no significant performance drop, but this study noted that QBFT's scalability was more favorable up to a moderate number of validators.

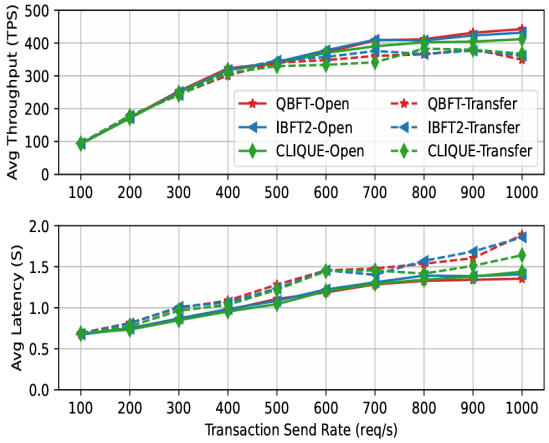
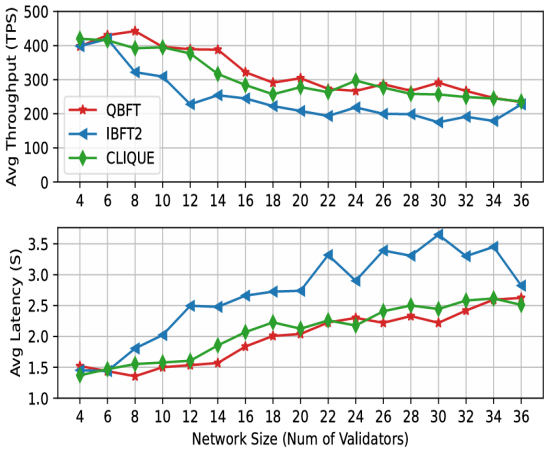


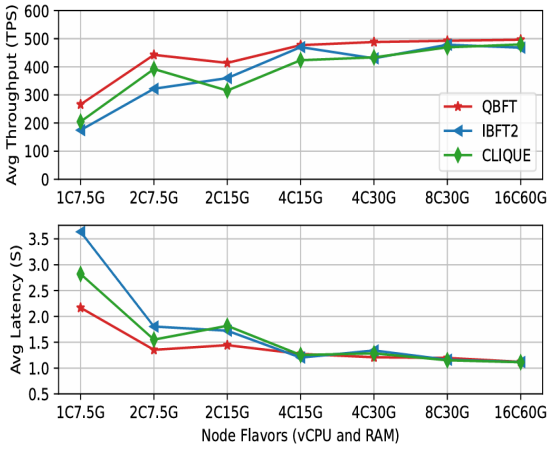
Figure 5.2: Performance against different proof of authority consensus algorithms under varying transaction send rates.

5.4 Associated Costs

When implementing a blockchain-based voting system there are multiple costs that need to be considered. These include the costs associated with deploying and maintaining smart contracts, using a permissionless blockchain, data storage, and the cost per vote. In this section, we will talk about each of the costs associated with our solution.



(a) Horizontal scalability against different PoA consensus algorithms at 1,000 req/s open transaction send rate.



(b) Vertical scalability against different PoA consensus algorithms at 1,000 req/s open transaction send rate.

Figure 5.3: Horizontal and Vertical scalability of the different PoA consensus algorithms

5.4.1 Smart Contract Costs

Smart contracts are self-executing contracts with the terms of the agreement directly written into the code. On a public blockchain, deploying and interacting with smart contracts incurs costs, primarily due to transaction, also known as gas fees. However, on a private blockchain (which is our case), these costs are significantly different.

- **Deployment Costs**: In a private blockchain, there are no direct gas fees when deploying smart contracts. The costs are primarily associated with the computational resources and energy required to deploy the contract within the network.
- **Transactions Costs**: Similarly, interactions with smart contracts (transactions) do not incur gas fees in a private blockchain. The costs are related to the processing power and energy needed to execute the functions.

In conclusion, for a private blockchain, the main costs are the power and the computational resources needed to maintain the network.

5.4.2 Permissionless Blockchain Costs

As discussed in the Section 3.2.2, the main goal of using a permissionless blockchain is to store a summary of the results as well as a pointer and a hash of the complete results (stored on IPFS). But before choosing a permissionless blockchain, we must take into account the costs associated with them.

We analyzed the transaction fees and the storage capacity of the five most known blockchains: Ethereum, Bitcoin, Binance Smart Chain, XRP, and Polygon.

1. **Ethereum** usually has high and variable transaction fees. In the last year, Ethereum transaction fees varied from a minimum of 1.35$ to a maximum of 29.50$ per transaction [10]. Ethereum supports storing data directly on the blockchain, but due to high gas fees, this can be quite expensive. The capacity for data storage in a single transaction is also limited, making it less pratical for large-scale data storage.

2. **Bitcoin** also has high and variable transaction fees. In the past year, it had a minimum transaction fee of around 1.30$ and a maximum transaction fee of 130$ [10]. Bitcoin can store small amounts of data, but it was not designed for large-scale data storage, ending up not being cost effective.

3. **Binance Smart Chain** (BSC) has a more stable and lower transaction fee then the previous two. In the last year, it had a minimum transaction fee of 0.02$ and a maximum transaction fee of 0.33$ [47]. BSC supports on-chain data storage with more reasonable costs compared to Ethereum and Bitcoin, providing a more efficient solution for moderate data storage.

4. **XRP** has even lower transaction fees, having reached in the last year a minimum transaction fee price of under 0.01$ and a maximum of 0.027$ [11]. XRP supports data storage but it was primarily optimized for financial transactions, thus having its data storage capacity relatively limited and not suitable for large-scale data.

5. **Polygon** also has low and stable transaction fees. In the past year the minimum transaction fee price reached under 0.1$, while the maximum transaction fee price reached 0.03$ [35]. As a Layer 2 solution on Ethereum, Polygon offers lower fees and faster transactions while also supporting moderate data storage.

In summary, Ethereum and Bitcoin have notably higher and more variable transaction fees compared to BSC, XRP and Polygon, which could make them less desirable for applications for data storage. Among the five blockchains, BSC and Polygon emerge as better options for data storage due to their lower fees and more efficient data storage capacities.

5.4.3 Data Storage Costs (IPFS)

The InterPlanetary File System [32] is a set of composable, peer-to-peer protocols for addressing, routing, and transferring content-addressed data in a decentralized file system. Because IPFS is open-source, there are multiple implementations of IPFS (such as Pinata and Temporal), thus resulting in different costs depending on the platform. In most implementations, the costs of storing data is around 0.10-0.20/gb per month, which compared to permissionless blockchains, ends up being a much cheaper option.

5.5 Security and Confidentiality

In our blockchain-assisted voting system, we were able to ensure security and confidentiality through a combination of a secure consensus algorithm, a proxy, and the encryption of votes. Below we present the key ways our system achieves these goals while maintaining performance:

- **Consensus Algorithm:** The network uses the QBFT Proof-of-Authority consensus protocol as it showed the best overall performance (as discussed in 5.3.3 Consensus Algorithms Performance Comparison), providing us with a secure voter validation and tampering prevention blockchain network.

- **Proxy:** Votes are submitted through a proxy, which separates voter identities from their votes. This ensures that voter identities remain confidential while preserving vote integrity. When adding the votes to the blockchain, the proxy takes between 4100 ms to 4300 ms to register a single vote, maintaining an efficient performance.

- **Encryption:** Votes are encrypted before being sent to the proxy. This ensures that the data inside each vote remains confidential. This way we also ensure that the elections are fair, stopping the votes from being counted in real time (*fairness* property in Security Properties 2.2.1). The encryption process is fast, with times under 2 ms, ensuring it does not slow down the execution.

This combination ensures data security and confidentiality while adding only a minimal overhead, well within our acceptable margins.

Chapter 6

Conclusion

In this work, we presented the design and implementation of our blockchain-assisted voting system with the purpose of improving security, transparency, and voter trust in the electoral process, with a primary focus on local elections. By studying and analysing the current voting systems and other proposed solutions, we were able to find the advantages and disadvantages of certain approaches, which helped us refine our own solution. We observed that the existing approaches primarily focused on leveraging blockchain for online voting, rather than adapting it for an in-person voting system, which was the primary goal of our solution.

We utilized Hyperledger Besu and Docker to quickly setup our private blockchain environment, as these platforms offered us the best guidance and conditions for implementing our solution. This approach significantly reduced setup time, which allowed us to focus more on the development and refinement of the voting system itself.

We also studied the different consensus algorithms offered by Hyperledger Besu, taking into account different key properties (immediate finality, minimum number of validators, liveness, and speed), and from our study, we concluded that QBFT was the best pick (as it was also recommended by Hyperledger Besu for enterprise-grade consensus protocols for private networks).

The proposed solution uses a Proxy, as a way to improve voter confidentiality, allowing the voters pick and identification to be separate (meaning it would be impossible to link the vote to the voter). This was a key element of our approach, differing from other implementations, as it ensured voter anonymity while simultaneously maintaining the integrity and security of the voting process. We developed the smart contract using Solidity and implemented our voting system logic using Node.js scripts, taking into account the different stages of the process (Initialization, Voting, and Counting).

To evaluate and test the proposed solution, we created a workload of 500 eligible votes and voters, automatically sending the votes to the voting system. In this way, we were able to time profile our developed scripts and test the network with real votes, which presented good execution times. We also showed a performance analysis on the different proof-of-authority consensus algorithms offered by Hyperledger Besu, analysing different performance metrics and key findings, which were inline with our consensus algorithm pick (QBFT).

In conclusion, the blockchain-assisted voting system presented in this thesis offers a new and promising solution to the challenges faced by traditional voting systems. By leveraging the potential of blockchain

technology, our approach not only addresses existing issues but also introduces a more secure and efficient alternative to conventional methods, particularly for local elections.

Bibliography

[1] Binance Academy. What Are Permissioned and Permissionless Blockchains? https://academy.binance.com/en/articles/what-are-permissioned-and-permissionless-blockchains. [Online; accessed 20-November-2023].

[2] Ali Mansour Al-madani, Ashok T. Gaikwad, Vivek Mahale, and Zeyad A.T. Ahmed. Decentralized e-voting system based on smart contract by using blockchain technology. In *Proceedings of the 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, pages 176–180, 2020.

[3] Maria Teixeira Alves. Innovation makers lança máquinas de autoatendimento hefesto para o setor bancário. https://jornaleconomico.sapo.pt/noticias/innovation-makers-lanca-maquinas-de-autoatendimento-hefesto-para-o-setor-bancario/, 2024. [Online; accessed 24-September-2024].

[4] Silvia Bartolucci, Pauline Bernat, and Daniel Joseph. Sharvot: secret share-based voting on the blockchain. In *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, ICSE '18. ACM, May 2018.

[5] Ali Benabdallah, Antoine Audras, Louis Coudert, Nour El Madhoun, and Mohamad Badra. Analysis of blockchain solutions for e-voting: A systematic literature review. *IEEE Access*, 10:70746–70759, 2022.

[6] Hyperledger Besu. Hyperledger besu. https://besu.hyperledger.org/, 2019. [Online; accessed 16-March-2024].

[7] Hyperledger Besu. Consensus protocols. https://besu.hyperledger.org/development/private-networks/how-to/configure/consensus, 2024. [Online; accessed 19-April-2024].

[8] Hyperledger Besu. Proof of authority consensus protocols. https://besu.hyperledger.org/development/private-networks/concepts/poa, 2024. [Online; accessed 19-April-2024].

[9] Hyperledger Besu. Quorum developer quickstart. https://besu.hyperledger.org/private-networks/tutorials/quickstart, 2024. [Version stable 24.9.1; Online; accessed 20-April-2024].

[10] bitinfocharts.com. Bitcoin, Ethereum Avg. Transaction Fee historical chart. https://bitinfocharts.com/comparison/transactionfees-btc-eth.html#1y. [Online; accessed 25-August-2024].

[11] bitinfocharts.com. XRP Avg. Transaction Fee historical chart. https://bitinfocharts.com/comparison/xrp-transactionfees.html#1y. [Online; accessed 25-August-2024].

[12] Blockchain.com. Blockchain fees per transaction. https://www.blockchain.com/explorer/charts/cost-per-transaction, 2024. [Online; accessed 25-July-2024].

[13] Comissão Nacional Eleições (CNE). Perguntas Frequentes: Apuramento. https://www.cne.pt/faq2/109/3. [Online; accessed 27-October-2023].

[14] Comissão Nacional Eleições (CNE). Perguntas Frequentes: Gerais. https://www.cne.pt/faq2/117/3. [Online; accessed 28-October-2023].

[15] CNNBrasil. Como é feita a contagem de votos depois do término das eleições. https://www.cnnbrasil.com.br/politica/como-e-feita-a-contagem-de-votos-depois-do-termino-das-eleicoes/, 2022. [Online; accessed 03-November-2023].

[16] Correio da Manhã. Teclas de urna eletrónica no brasil coladas com cola de secagem rápida. https://www.cmjornal.pt/mundo/detalhe/eleicoes-brasil-teclas-de-urna-eletronica-coladas-com-cola-de-secagem-rapida, 2022. [Online; accessed 02-November-2023].

[17] Thiago N.C. Cardoso Caio Lüders Araújo Paulo Matias Diego F. Aranha, Pedro Y.S. Barbosa. The return of software vulnerabilities in the brazilian voting machine. https://doi.org/10.1016/j.cose.2019.06.009, 2019. [Online; accessed 02-November-2023].

[18] Portal do Eleitor. Membros da mesa de voto. https://www.portaldoeleitor.pt/pt/Eleicao/MembrosMesaVoto/Pages/default.aspx. [Online; accessed 25-October-2023].

[19] Justiça Eleitoral. Por que o processo é eletrônico? https://www.justicaeleitoral.jus.br/urna-eletronica/historico-das-fraudes-nas-eleicoes.html. [Online; accessed 30-October-2023].

[20] Justiça Eleitoral. Detalhes técnicos da urna eletrônica modelo 2020. https://portallitoralsul.com.br/mesario-e-condenado-a-prisao-por-fraude-nas-eleicoes-de-pescaria-brava, 2020. [Online; accessed 02-November-2023].

[21] Tribunal Regional Eleitoral. Urna eletrônica tem mais de 30 camadas de segurança. https://www.tre-sp.jus.br/comunicacao/noticias/2021/Junho/urna-eletronica-tem-mais-de-30-camadas-de-seguranca-1, 2022. [Online; accessed 02-November-2023].

[22] Gibbins Roger Eulau, Heinz and Paul David Webb. election. https://www.britannica.com/topic/election-political-science, 2024. [Online; accessed 14-August-2024].

[23] Caixiang Fan, Changyuan Lin, Hamzeh Khazaei, and Petr Musilek. Performance analysis of hyperledger besu in private blockchain. In *Proceedings of the 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 64–73, 2022.

[24] FastAPI. FastAPI. https://fastapi.tiangolo.com/#create-it. [Online; accessed 25-August-2024].

[25] Hyperledger Foundation. Hyperledger foundation. https://www.hyperledger.org/, 2024. [Online; accessed 02-November-2023].

[26] US Vote Foundation. The Myth of "Secure" Blockchain Voting. https://www.usvotefoundation.org/blockchain-voting-is-not-a-security-strategy. [Online; accessed 15-November-2023].

[27] G1Globo. Eleitor quebra urnas eletrônicas a socos e é preso em flagrante em sp. https://glo.bo/1oM3u91, 2014. [Online; accessed 02-November-2023].

[28] Freya Sheer Hardwick, Apostolos Gioulis, Raja Naeem Akram, and Konstantinos Markantonakis. E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. https://arxiv.org/abs/1805.10258, 2018.

[29] Hillary. How blockchain is enabling the future of secure and transparent voting systems. https://techbullion.com/how-blockchain-is-enabling-the-future-of-secure-and-transparent-voting-systems/, 2024.

[30] Md Jobair Hossain Faruk, Fazlul Alam, Mazharul Islam, and Akond Rahman. Transforming online voting: a novel system utilizing blockchain and biometric verification for enhanced security, privacy, and transparency. *Cluster Computing*, 27, 04 2024.

[31] IBM. What is blockchain technology? https://www.ibm.com/topics/blockchain. [Online; accessed 10-November-2023].

[32] IPFS. Ipfs. https://docs.ipfs.tech/, 2024. [Online; accessed 25-July-2024].

[33] Secretário-Geral Adjunto da SGMAI Joaquim Morgado. Autarquias Locais 2021 Manual dos Membros das Mesas Eleitorais. https://www.sg.mai.gov.pt/AdministracaoEleitoral/EleicoesReferendos/AutarquiasLocais/Documents/Doc_Manual_MMesas_AL%20-%20site.pdf, 2021. [Online; accessed 25-October-2023].

[34] David Khoury, Elie F. Kfoury, Ali Kassem, and Hamza Harb. Decentralized voting platform based on ethereum blockchain. In *Proceedings of the 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 1–6, 2018.

[35] polygonscan. Average Transaction Fee Chart. https://polygonscan.com/chart/avg-txfee-usd. [Online; accessed 25-August-2024].

[36] Hyperledger Sawtooth. Hyperledger sawtooth. https://www.hyperledger.org/hyperledger-sawtooth-1-0, 2018. [Online; accessed 16-March-2024].

[37] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. Smart contract: Attacks and protections. *IEEE Access*, 8:24416–24427, 2020.

[38] Portal Litoral Sul. Mesário é condenado à prisão por fraude nas eleições de pescaria brava. https://portallitoralsul.com.br/mesario-e-condenado-a-prisao-por-fraude-nas-eleicoes-de-pescaria-brava/, 2023. [Online; accessed 02-November-2023].

[39] Shivani Tripathi. Hyperledger Fabric Consensus Mechanisms: Exploring the Options. https://www.spydra.app/blog/hyperledger-fabric-consensus-mechanisms-exploring-the-options, 2023. [Online; accessed 20-November-2023].

[40] Tribunal Superior Eleitoral (TSE). Preparação das urnas. https://www.tse.jus.br/eleicoes/historia/processo-eleitoral-brasileiro/logistica-e-preparacao/preparacao-das-urnas. [Online; accessed 01-November-2023].

[41] Tribunal Superior Eleitoral (TSE. Procedimentos de contingência. https://www.tse.jus.br/eleicoes/urna-eletronica/seguranca-da-urna/procedimentos-de-contingencia. [Online; accessed 04-November-2023].

[42] Tribunal Superior Eleitoral (TSE. Urna eletrônica. https://www.tse.jus.br/internet/temporarios/urna-seguranca/identificacao-biometrica.html. [Online; accessed 02-November-2023].

[43] votarEleicoes. Votar em eleições portuguesas. https://eportugal.gov.pt/guias/votar, 2022. [Online; accessed 26-October-2023].

[44] Wikipedia contributors. Election — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Election&oldid=1241535654, 2024. [Online; accessed 21-August-2024].

[45] Wikipedia contributors. Voting — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Voting&oldid=1240582592, 2024. [Online; accessed 21-August-2024].

[46] wikiSporting. Regulamento Eleitoral. https://www.wikisporting.com/index.php?title=Regulamento_Eleitoral. [Online; accessed 05-November-2023].

[47] YCharts. Binance Smart Chain Average Transaction Fee. https://ycharts.com/indicators/binance_smart_chain_average_transaction_fee_es. [Online; accessed 25-August-2024].

[48] Bin Yu, Joseph Liu, Amin Sakzad, Surya Nepal, Paul Rimba, Ron Steinfeld, and Man Ho Au. Platform-independent secure blockchain-based voting system. Cryptology ePrint Archive, Paper 2018/657, 2018. https://eprint.iacr.org/2018/657.

[49] Peng Zhang, Douglas C. Schmidt, Jules White, and Abhishek Dubey. Chapter seven - consensus mechanisms and information security technologies. In Shiho Kim, Ganesh Chandra Deka, and Peng Zhang, editors, *Role of Blockchain Technology in IoT Applications*, volume 115 of *Advances in Computers*, pages 181–209. Elsevier, 2019.

Appendix A

Innovation Makers HEFESTO

HEFESTO is a self-service machine developed by Innovation Makers, designed to enhance banking and financial services by integrating advanced technologies such as Artificial Intelligence (AI), Machine Learning (ML), and Biometrics. This innovative solution is built with a modular and customizable architecture, allowing financial institutions to tailor services to meet specific market demands and customer preferences [3].

One of HEFESTO's primary focuses is security, specifically client identification through biometric data. The system is designed to address and enhance security in three critical areas:

- **Anti-Money Laundering:** Previously, some individuals would deposit large sums of money, cancel the transaction, and then receive a credit note allowing them to withdraw the amount at a service counter. HEFESTO overcomes this issue by incorporating financial components that immediately return the exact notes deposited if the operation is canceled, thus preventing exploitation of the system.
- **Identification Document Verification:** HEFESTO includes a card scanner that uses multiple methods to verify the authenticity of identification documents. The scanner performs scans under infrared and ultraviolet light, checking for various security features such as embedded images, opacity, and ultraviolet markings to ensure the document is genuine.
- **Biometric Identification:** HEFESTO ensures that the individual using the machine is the legitimate owner of the identification document presented. If biometric identification cannot be confirmed, the system imposes a low transaction limit to mitigate potential risks.

HEFESTO's modular architecture allows financial institutions to configure machines based on the specific services they want to offer, providing both flexibility and efficiency. This adaptability makes it well-suited for the potential integration of a secure voting system. In image A.1, we can see the 3D models of the various components that can be integrated into HEFESTO's system, showcasing its modular design and flexibility.

Nomenclatura

IF – Interface de utilizador **PG** – Máquina de pagamentos **DP** – Máquina de depósitos **CO** – Utilitário complementar **DS** – Dispensador de notas **MD** – Recolha e processamento de moedas

IF-X00 Interfaces multifunções de utilizador

	IF100	IF300	IF500
Ecrã Tátil	15"	17"	27"
Filtro Privacidade	Não	Sim	Sim
POS com NFC	Não	Opcional	Opcional
Biometria	Opcional	Opcional	Opcional
Scan Cartões	Opcional	Opcional	Opcional
Finger / Pad Ass.	Não	Opcional	Opcional

PG-X00 Pagamentos e depósitos, aceita notas e/ou moedas com tratamento de trocos

	PG300	PG500
Noteiro	Nota a nota (1s)	Bulk 50 notas (6s)
Escrow	Não	60 notas
Moedeiro	Opcional	Opcional
Troco	Sim	Sim
Reciclador	60 notas/300 moedas*	240 notas/300 moedas*
Cofre	Opcional	Reversível

* N.º médio de moedas, depende do tipo de divisa

DP-X00 Depósitos de alto desempenho com e sem reciclagem

	DP300	DP500	DP700
Noteiros	Bulk 100 notas (6s)	Bulk 200 notas (12s)	Bulk 200 notas (12s)
Dispensação	Não	Não	Não
Informação de Dados da Nota	Não	Imagem/n.º de série	Imagem/n.º de série
Cofre	Opcional	Reversível	Frontal
Transporte Valores	Saco até 20.000 notas	Saco até 30.000 notas	5 cofres de 3.000 notas *
Escrow	Até 100 notas	Até 200 notas	Até 200 notas

* Contém reciclagem de notas e notas rejeitadas

CO-X00 Integração de periféricos complementares

	CO300	CO500
Impressora Laser A4	Sim	Sim
Scanner A4	Opcional	Opcional
Impressão e Disp. de cartões	Opcional	Opcional
Laminador de cartões	Opcional	Opcional
Gravação em cartões	Chip e/ou banda	Chip e/ou banda
Impressão de bilhetes	Opcional	Opcional

Figure A.1: HEFESTO Machine Components 3D Models

Appendix B

Smart Contract Functions Code

In this appendix we present the code of the smart contract functions developed:

- ***statusVoting()***: Changes the status from Initializing to Voting;

```
function statusVoting() public onlyOwner returns (Status){
    require(currentStatus == Status.Init || currentStatus == Status.Voting,"Contract
        isn't in Init status");
    if(currentStatus == Status.Init){
      currentStatus = Status.Voting;
    }
    return currentStatus;
}
```

- ***statusCounting()***: Changes the status from Voting to Counting;

```
function statusCounting() public onlyOwner returns (Status){
    require(currentStatus == Status.Voting || currentStatus == Status.Counting,"
       Contract isn't in Voting status");
    if(currentStatus == Status.Voting){
      currentStatus = Status.Counting;
    }
    return currentStatus;
}
```

- ***initializeVotersList()***: Initializes a map with the eligible voters value set to 1;

```
function initializeVotersList(string[] memory keys) public onlyInit onlyOwner {
    for(uint i=0; i < keys.length; i++){
        eligibleVoters[keys[i]] = 1;
    }
}
```

- ***addId()***: Checks if id is eligible to vote, if eligible stores id in the ids list and returns true;

```
function addId(string memory id) public onlyVoting returns (bool){
    require(eligibleVoters[id] == 1,"The user is not eligible to vote (already voted
        or isn't eligible)");
    ids.push(id);
    eligibleVoters[id] = 2;
```

```
    return true;
}
```

- ***addVote()***: Adds encrypted vote to the votes list;

```
function addVote(bytes memory iv, bytes memory encryptedData) public onlyVoting {
    EncryptedVote memory newVote = EncryptedVote(iv, encryptedData);
    votes.push(newVote);
}
```

- ***getIdsList()***: Returns the stored list of ids;

```
function getIdsList() public view onlyCounting returns (string[] memory) {
    return ids;
}
```

- ***getVotesList***: Returns the stores list of encrypted votes;

```
function getVotesList() public view onlyCounting returns (EncryptedVote[] memory) {
    return votes;
}
```

Appendix C

Proxy Script

In this appendix, we present a code snippet of our Proxy.js file. This script receives votes and stores them in a list. When the number of stored votes exceeds X or when the timeout is reached, it shuffles the votes and sends them to the blockchain.

```
async function processVotes(contractWithSigner){
  list = shuffle(list);
  const votesToSend = list.splice(0, list.length); //list.length
  for(let i=0; i<votesToSend.length;i++){
      const tx = await send_to_network(contractWithSigner, votesToSend[i]);
  }
  processOpen=true;
}

async function main(){
    //Find SmartContract Address
    const sc_address = await find_SC_Adress();

    const contract = new ethers.Contract(sc_address, contractAbi, provider);
    const contractWithSigner = contract.connect(wallet);
    var timeoutId = resetTimer(); //timedout=true

    //Add vote to list
    process.on("message", (msg) => {
        if(msg == "stop"){
          to_stop = true;
        } else{
          send_to_proxy(msg.vote);
        }
    });

    //Timers (checks number of votes every x time)
    setInterval(() => {
        if(list.length >= 10 && processOpen){
            processOpen=false;
            clearTimeout(timeoutId);
            processVotes(contractWithSigner);
            timeoutId = resetTimer();
        }
```

```
        else if(timedout || to_stop){
            if((list.length > 0 && processOpen)){
                processOpen = false;
                processVotes(contractWithSigner);
            } else if(to_stop && list.length == 0 && processOpen){
                process.exit(0);
            }
            timedout=false;
            timeoutId=resetTimer();
        }
        else if(list.length == 0 && to_stop && processOpen){
            process.exit(0);
        }
    }, 3000);
}
```

Appendix D

Main Scripts

In this appendix, we present code snippets of key functions from the four main *scripts* developed for the election process. These *scripts* were written using *JavaScript* and the *ethers* library.

D.1 SC_deploy.js

The script deploys the compiled smart contract to the blockchain using the ContractFactory from the *ethers* library.

```
async function createContract(provider, wallet, contractAbi, contractByteCode) {
    const factory = new ethers.ContractFactory(contractAbi, contractByteCode, wallet
        );
    const contract = await factory.deploy();
    // The contract is NOT deployed yet; We must wait until it is mined;
    const deployed = await contract.waitForDeployment();
    //The contract is deployed now
    return contract
};

async function main(){
    const provider = new ethers.JsonRpcProvider(host);
    const wallet = new ethers.Wallet(accountPrivateKey, provider);
    try {
        const contract = await createContract(provider, wallet, contractAbi,
            contractBytecode);
        await contract.waitForDeployment();
        const contractAddress = await contract.getAddress();
        console.log("Contract deployed at address: " + contractAddress);
    } catch (error) {
        console.error('Error deploying contract:', error);
    }
}
```

D.2 SC_InitializeEligibleVoters.js

The script initializes the eligible voters list in the smart contract by reading voter IDs from the 'VotersIds.txt' file and calling the *initializeVotersList* function on the smart contract.

```
async function initializeEligibleVoters(contractWithSigner, namesList){
    const tx = await contractWithSigner.initializeVotersList(namesList);
    await tx.wait();
    console.log("List of voters initialized!");
    return tx;
}

async function main(){
    const provider = new ethers.JsonRpcProvider(host);
    const wallet = new ethers.Wallet(accountPrivateKey, provider);
    const sc_address = await find_SC_Adress();
    const contract = new ethers.Contract(sc_address, contractAbi, provider);

    const filename = 'VotersIds.txt';
    const namesList = readNamesFromFile(filename);

    const contractWithSigner = contract.connect(wallet);
    const status = await others.getCurrentStatus(contractWithSigner);
    console.log("Status: " + status);
    await initializeEligibleVoters(contractWithSigner, namesList);
}
```

D.3 Voting_Process.js

The script begins by finding the deployed smart contract address and initializing the proxy. Next, it updates the smart contract status to 'Voting' before connecting to the *EventSource API* to await incoming votes. Upon receiving a vote, the script verifies and registers the voter's ID on the blockchain. If the voter is deemed eligible, the vote is encrypted and sent to the proxy.

```
async function main(){

    // Find SmartContract Address
    const sc_address = await find_SC_Adress();

    // Smart Contract with Signer
    const contract = new ethers.Contract(sc_address, contractAbi, provider);
    const contractWithSigner = contract.connect(wallet);

    // "Turn ON" Proxy asynchronously
    const child = fork("./Proxy.js"); // Fork the worker process

    // Changes the smart contract status to Voting
    const tx_status = await others.statusToVoting(contractWithSigner);
    await tx_status.wait();
```

```
    // Waits for exit signal from child, turns off loop (voting) after
    var sent_stop_to_child = false;
    child.on("close", (code) => {
        console.log("Child process exited with code:", code);
        console.log("Ending voting process...");
        process.exit()
    });

    // Connect to EventSource on API
    const EventSource = require('eventsource');
    const es = new EventSource('http://localhost:8000/events');

    // Recieves msg from API events (vote & stop)
    es.onmessage = async function(event) {
        try {
            const message = JSON.parse(event.data); // Parse the JSON string into an
                object
            console.log('Received Event:', message);
            if (message.type === "vote") {
                const vote = message.data;
                var id = vote.id;
                var canVote = false;
                try {
                    canVote = await registerIdAtAddress(contractWithSigner, id);
                    await canVote.wait();
                } catch(error){
                    console.log(error);
                }
                if(canVote){
                    v = vote.choice;
                    const voteencrypt = encrypt(v.toString()); // Encrypt vote
                    await registerVoteAtAddress(child, voteencrypt);
                }
            } else if (message.type === "stop") {
                mess = JSON.parse(message.data);
                if(mess.is_stopped && !sent_stop_to_child){
                    child.send("stop");
                    console.log("STOP: Proxy will shutdown after processing votes...
                        ");
                    sent_stop_to_child = true;
                }
            }
        } catch (error) {
            console.error('Failed to parse event data:', error);
        }
    };
    es.onerror = function(err) {
      console.error('EventSource failed:', err);
    };
}
```

D.4 Counter.js

The script begins by locating the smart contract address, updates the contract status to 'Counting,' and then retrieves, decrypts, and counts all votes recorded on the blockchain.

```
async function countVotes(){
    const votes =[];
    var txList = [];
    txList = await fetchTxs.getAllTransactions();
    for(let i=0; i<txList.length;i++){
        const decoded = decoder.decodeData(txList[i].input);
        if(decoded.method == "addVote"){
            decryptedVote = decrypt(decoded.inputs[0], decoded.inputs[1]);
            votes.push(decryptedVote);
        }
    }
    const candidates = readNamesFromFile("Candidates.txt");
    const results = findResults(votes, candidates);
    return results;
}

async function main(){
    const sc_address = await find_SC_Adress();
    const contract = new ethers.Contract(sc_address, contractAbi, provider);
    const contractWithSigner = contract.connect(wallet);

    // Change SC Status to Counting
    const tx_status = await others.statusToCounting(contractWithSigner);
    await tx_status.wait();

    const results = await countVotes();
    console.log(results);
}
```