¿ES LA TECNOLOGÍA BLOCKCHAIN EL FUTURO DEL VOTO?

IS BLOCKCHAIN TECHNOLOGY THE FUTURE OF VOTING?

TRABAJO FIN DE GRADO
CURSO 2021-2022

AUTOR
MATÍAS LUIS LOTITO RALLI

DIRECTOR
JESÚS CORREAS FERNÁNDEZ

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

RESUMEN

¿Es la tecnología Blockchain el futuro del voto?

Ponerse de acuerdo nunca ha sido tarea fácil. Desde siempre el diálogo y el debate de ideas han sido necesarios para ganar apoyos y de esta forma organizar y hacer prosperar una sociedad. No obstante, es cuando este debate se desvía cuando la situación se complica y se estanca.

Se ha podido ver en las últimas elecciones en Estados Unidos cómo, cada vez más, los partidos políticos, e incluso los propios ciudadanos, ponen en cuestionamiento los resultados de unas elecciones y el conflicto que surge a raíz de ello. Esto se debe a que los sistemas actuales de votación, como las urnas, y más aún los sistemas automatizados de votación, son manipulables y carecen de transparencia y trazabilidad, y en un momento donde la población es cada vez más desconfiada, esto supone una mayor inestabilidad.

En paralelo a todo esto, en los últimos años la tecnología Blockchain y las redes descentralizadas se han ido desarrollando continuamente. Sistemas de gobernanza, entidades financieras, cadenas de suministros, IoT… Esta tecnología ha suscitado mucho interés en multitud de sectores por las cualidades únicas que proporciona, entre las cuales destacan la incorruptibilidad de los datos, la trazabilidad y la transparencia.

Es por ello que la tecnología Blockchain se presenta como una buena opción para los problemas actuales de los sistemas de votación. Sin embargo, ¿es viable actualmente? ¿O por el contrario hay otros retos que deben ser superados previamente?

Palabras clave

Blockchain, sistemas de gobernanza, e-voting, active citizen, elecciones, voto, nodos, derechos de voto, sistemas de votación, ethereum

ABSTRACT

Is blockchain technology the future of voting systems?

It has never been an easy task to come to an agreement. Dialogue and discussion have been always necessary to obtain support from others so societies can be arranged and developed. Nonetheless, it is when this dialogue and discussion diverts when society comes to a stop.

As it could be seen in the last United States elections, political parties, and even civilians, are increasingly questioning the results from a democratic election and the conflict that emerges from it. This has to do with the current voting systems, like ballots boxes, and more importantly automated voting systems, as they are corruptible and have a lack of transparency and traceability, and in a moment where an untrusting society arises, this causes mayor instability.

In parallel, in the recent years Blockchain technology and decentralized networks have been evolving unstoppably. Governance systems, financial entities, supply chains, IoT… Blockchain has awaken much interest in various sectors for the unique characteristics it offers, like the incorruptibility of data, traceability, and transparency.

For these reasons, Blockchain technology presents as a good option for current voting system problems. However, is it currently a viable option? Or there are other challenges that must be addressed previously?

Keywords

Blockchain, governance systems, e-voting, active citizen, elections, voting, nodes, voting rights, voting systems, ethereum

TABLE OF CONTENTS

1. Introduction

Voting systems are vital for a healthy democracy. They carry the voice of the people, and people carry the will of a country. If they see that power is occupied by a governor that they think it is illegitimate, they are going to rebel, and conflict will emerge.

In the last United States election (2020) it could be seen how political parties, and even civilians, are questioning increasingly the results from a democratic election, and the unfortunate consequences that could be seen from this, as the raid to the Capitol. It is not about if those elections were tampered or not, it is about the scepticism the system caused. Thus, it is essential to offer a solid tamper-proof system that everyone, even if they do not want to, will agree on that.

In parallel, in the recent years Blockchain technology and decentralized networks have been evolving unstoppably. Governance systems, financial entities, supply chains, IoT… Blockchain has awaken much interest in various sectors for the unique characteristics it offers, like the incorruptibility of data, traceability, and transparency. Therefore, based on its properties, first-hand it seems that blockchain could be suitable for a voting system implementation.

Therefore, this Final Degree Project intends to provide an answer to the question that heads this report cover page: **Is Blockchain technology the future of voting systems?**

To accomplish this goal, we have elaborated the following work plan:

1. Gather documentation about what is a voting system and the properties it should have, as designing something requires knowing its properties. Gather characteristics and flaws of traditional systems like ballot boxes and modern ones, like e-voting.

2. Understand the state of art by investigating current blockchain-based voting systems, how they work and if see if they comply with the

properties that a voting system must have. Investigate if some country applies blockchain technology for this goal.

3. Study research works and academic papers published on this field and study what the contribute and what they lack comparing them with each other and the voting properties we previously defined.

4. Design and build a prototype of a blockchain-based voting system that solves the studied issues and implement it.

5. Test the system implemented.

2. Voting Systems

2.1 Definition and characteristics

First, we should define what do we mean by a voting system. And, as the title question suggests, why does it need a future? Is it not enough with what it is today?

A **voting system** is a set of rules that determine how elections and referendums are conducted among a set of voters and how their results are determined[1]. There are many different voting systems around the globe, like plurality systems, majoritarian systems, proportional systems… Each of these with a different set of rules. However, these rules extend to a broad range of social and political fields that are out of the scope of this report and are not going to be worked with, like who can be elected, how votes are counted, who wins the elections, how political campaigns are managed, the degree of political freedom…

Nevertheless, apart from voting systems, there is another vital aspect of elections. Whereas voting systems set the rules of the game, **voting rights** set the criteria of *who* can participate in the game and the rights they should have.

2.2 Voting rights

Voting rights is something that we all know and take for granted nowadays. However, they have suffered a constant evolution along history. Each country had their own pace, but they have ended converging into the following principles:

- **Universal suffrage**: every citizen from a country has the right to participate in elections and must be offered the means to do so.
- **Equal suffrage**: there are no graded votes (all votes weight the same), and all voters have the same number of votes, typically one. It is built by the US principle "One man, one vote"[2].

[1] https://en.wikipedia.org/wiki/Electoral_system

[2] https://en.wikipedia.org/wiki/One_man,_one_vote

- **Freedom of choice**: voters have the freedom to choose the option that they prefer, without the coercion of any individual nor third party.
- **Secret vote**: this guarantees the preservation of the previous rule, as not keeping voters' identity in secret may end up influencing them by intimidation or extortion.

However, it is widely known that not every country in the modern days follow these rights. For example, Russia, despite proclaiming themselves as a democratic country (Russia Constitution, 1993), is known to have jailed and repressed political opponents[3], to have no real opposition, and to have committed electoral fraud (R. Mebane & Kalinin, 2010). Another similar case is China, who directly does not perform elections at national level, but at local level[4], and there is a single legal party. Nonetheless, they state themselves as a *people's democratic dictatorship* (China Constitution, 2019). All these cases violate the freedom of choice principle, as they do not allow the existence of other options or directly commit tamper fraud.

Once the latter said, we can all agree that it is important to preserve voting rights to have fair elections. No matter what voting system is used, voting rights must be followed.

2.3 Current approaches

Different approaches have been conceived to implement tamperproof democratic voting systems. Some of them look to simplify the voting process, others look to accelerate it, while other attempt to facilitate the process for some part of the population.

In this section different systems are going to be analysed, as well as their tamperproof methods.

[3] https://www.bbc.com/news/world-europe-60832310

[4] https://en.wikipedia.org/wiki/Elections_in_China#Legislation

2.3.1 Ballot boxes

We begin this revision with the most traditional system to register votes with the objective of analysing how it preserves voting rights, as this analysis will help us to define later the configuration an automated voting system must have, and particularly one based on blockchain technology.

A ballot box is a temporarily sealed container with a narrow slot int the top sufficient to accept a ballot paper which prevents anyone from accessing votes cast until the voting period is closed[5]. They are well known for their ease of use and are indeed the most used system around the world. They are directly managed by officials who verify the identity of voters so they control who can vote and the vote cast. So, how do they preserve voting rights?

- **Universal suffrage:** officials allow access to vote to any citizen registered in that polling stations census.
- **Equal suffrage:** officials control that each voter cast a single vote.
- **Freedom of choice:** the place where the ballot box is in have ballot papers for each option presented in the election.
- **Secret vote:** voters must not be watched when picking up their ballot paper. Sometimes a voting booth is offered. Also, their vote is sealed in a proper opaque envelope so nobody can see their choice until the end of elections.

Ballot boxes do a great job preserving voting rights. However, as it can be seeing officials play a key role on ballot boxes as they guarantee their correct functioning. So, what if they are not so well-meaning?

Normally officials are watched by each other or even by politic members so they do not act by their own. But what if they sneak, previous to the election, and fill the ballot box with pre-casted votes? This would directly tamper election results. To prevent that, ballot boxes must be transparent so everyone can see that they start

[5] https://en.wikipedia.org/wiki/Ballot_box

empty. Also, this lets people directly see if their vote is correctly casted into the ballot box.

Another thing that could happen is that, when they are counting votes, the officials purposely alter them into their favour or directly do not count them. Again, officials watch each other, but it could happen that a bad-intentioned official takes advantage of other, distracted, officials, and successfully alter some of the votes.

However, these are minor manipulations when scaled to a whole country. Big manipulations, those that significantly affect to results or directly changes them into another direction, are those which are organized by governments. Russia or Latin American governments, like Venezuela, are known to have altered results in favour of the government that was in power at that moment. These situations are particularly the ones that worry most to voters, as they do not really have any power nor mean to accurately show to the world that their vote was altered.

2.3.2 Mail voting

It is a type of absentee voting implemented in various countries, like Spain and United States, where ballot papers are distributed by post to electors, who typically cannot go in person to vote, and then they can send them back to the corresponding administration[6]. There are other cases, as in Spain as well, that if the voter is residing abroad at that moment those ballot papers sent by post can also be casted in ballot boxes available in the embassy of the corresponding country the voter is currently living[7].

The issues here are quite the same as ballot boxes: there are no means for voters to know if their vote was correctly cast, and those votes could be tampered by a malicious central administration. However, there is an additional issue here: the voter does not directly see if the vote is correctly taken into account, like in ballot boxes

[6] https://en.wikipedia.org/wiki/Postal_voting

[7]https://administracion.gob.es/pag_Home/ca/Tu-espacio-europeo/derechos-obligaciones/ciudadanos/residencia/elecciones/votar-desde-extranjero.html

where you literally see how your ballot enters into the box. Another problem is that the process for postal voting requires more steps than traditional voting and is more error-prone. For example, in Virginia, August 2020, around 500.000 applications for an absentee ballot (how postal voting is called in the US) were sent out with the wrong return address. The electoral authorities said that there was no fraudulent intent and that the mistake was corrected. A similar problem occurred in Ohio, in early October, where about 50.000 voters received the wrong ballot in the post. Again, it was serious mistake, but there was no evidence of fraud[8].

Therefore, fraudulent conspiracies may arise from these events, like the ones arisen on the 2020 US elections where mail voting was one of the causes that Trump claimed for voter fraud. It is important to note that, afterwards, proofs and statistical demonstrations have been presented against the idea that there was voter fraud, and all legal challenges were unsuccessful, but 75% of Republican voters found merit in claims that millions of fraudulent ballot were cast, voting machines were manipulated, and thousands of votes were recorded for dead people (C. Eggers, Garro, & Grimmer, 2021). Thus, scepticism was still around, and if it cannot be guaranteed that there was no tampering, it does not matter if any took place. That is the main issue.

2.3.3 Voting machines

Voting machines are a type of electronic voting device that are used to record votes without paper. The first ones were mechanical, but nowadays electronic voting machines are commonly used. They are faster than manual counting, but, as they are machines, they can show vulnerabilities and other issues.

Voting machines are commonly used in the US and, as mentioned previously, they were one of the causes of the 2020 election suspicions. A particular case is the WINVote touchscreen machine, made by Advanced Voting Solutions, that was used

[8] https://www.bbc.com/news/world-us-canada-53353404

in Virginia elections between 2003 and 2015[9]. A security expert found that anyone within a half-mile of a voting machine could have altered votes without detection[10]. There was no evidence that these machines were altered but, as mentioned before, if it cannot be proved, suspicions will arise.

2.3.4 Online voting (E-Voting)

Online voting is another type of electronic voting where votes are casted and tallied through an internet service. These systems rely on a central server that manages all operations and security issues.

Online voting has been used for a long time. Estonia became the first country to use it in general elections in 2005 and it is still being used; Switzerland used it since 2003 in some cantons but is considering banning it; Canada used it just for municipal elections in some provinces, like Ontario and Nova Scotia.

As it can be seen, e-voting is a voting mechanism used quite often, and it has been studied for a long time, and experts have found security problems in every attempt at online voting.

To see a particular case, Swiss Post's, the national postal service of Switzerland, created the system "sVote" which was used in Fribourg, Neuchâtel and Thurgau in 2018. This company has recently partnered in 2021 with the French platform YesWeHack to launch a bug bounty reward of up to 230.000$ for critical vulnerabilities in their system[11], as, in 2019, politicians and computer experts launched a people's initiative aimed at banning the use of e-voting for at least 5 years for security reasons[12].

---

[9] https://www.wired.com/2015/08/virginia-finally-drops-americas-worst-voting-machines/

[10] https://www.wired.com/2016/08/americas-voting-machines-arent-ready-election/

[11] https://www.securityweek.com/swiss-post-offers-%E2%82%AC230000-critical-vulnerabilities-e-voting-system

[12] https://www.swissinfo.ch/eng/politics/online-democracy_opposition-against-e-voting-project-gathers-pace/44708930

This initiative was followed by some security analysis from experts among the internet. Currently, it has 477 bug reports[13]. For example, Ruben Santamarta is a European security researcher that participated in this project and found some software and hardware security issues (Santamarta, 2022).

Online voting is also widely used among the private sector for shareholders votes. For example, EVoting is a Chilean company that offers vote services and helps organizing them. They explain how their system works at their home page[14], like how homomorphic algorithms are used to tally the votes without incurring into the violation of voter privacy. However, as the code is private, there is no way of verifying its security. The unique way of trusting them is by the certificates they possess[15].

Therefore, online voting is a complex matter. On one hand, they offer unique properties, like absentee voting and faster tally of the results, but on the other hand they need to be well audited and are prone to have many security issues.

In conclusion, online voting also acts as a black box for voters which may cause scepticism among voters and, on top of that, it directly depends on a central server that offers the service, which makes a single point of failure.

2.4 Expanding voting rights

It can be seen that all previous approaches have several common issues:

- Once the vote is casted, you lose track of it. There is no traceability.
- The tally process is made privately. Voters must trust that the results were calculated correctly. There is no transparency.
- A corrupted government or administration could alter the election results. Data is corruptible.

[13] https://yeswehack.com/programs/swiss-post

[14] https://evoting.com/en/seguridad-integral/secreto-voto/?lang=en

[15] https://evoting.com/seguridad-integral/certificaciones/

- All systems depend on a central authority that manages the tally and offers the voting service, either by ballot boxes or by online voting.

So, taking into account the drawbacks from the previous approaches, it becomes a necessity to incorporate to the system new "voter rights": **vote verification** and **public tallying**:

- Vote verification would give voters the right to check whether their vote was correctly casted and if it was altered or not.
- Public tallying gives all participants the right to check the tallying process, and if it was done correctly or not.

Blockchain features seem to address all issues and appear to be the ideal foundation for trustworthy automated voting.

3. Blockchain approach

The Blockchain technology was first introduced in 2008 with Satoshi Nakamoto's paper (Nakamoto, 2008) and few months later, the first Blockchain system, Bitcoin, was created in the 3th of January of 2009. This paper aimed at creating a system for electronic transactions without relying on trust nor a central authority and also at giving a solution to the double-spending problem of digital currencies. Later, more blockchains and cryptocurrencies were created, like Ethereum or Solana, among others.

However, how can Blockchain concepts be applied to a voting system? Firstly, it is important to understand how a Blockchain works in order to work with it.

3.1 How blockchain works

A Blockchain, in its most simple and intuitive form, is a distributed ledger of transactions between participants, called ***nodes***. Each node has a copy of the whole ledger and, every time a new transaction is performed, it is mathematically linked to the previous one by a method called hash function and written into everyone else's ledger. Normally, to save time and resources, transactions are packed into ***blocks of transactions,*** so it is easier to link and store them. Since every block is linked to the previous one, they form a "chain" of blocks, from where it gets its name: blockchain.

For example, for the sake of the explanation, let us assume that we have three friends, John, Maria and Sofia, who decided to set up their own blockchain economy because they heard all of the amazing properties it offers. To do so, they reach an agreement of how much balance each of them will start with in this closed economy. Also, they agree that, in order to secure the nodes money, each of the participants will sign the transactions they make so no one can spend money from others.

So, they decide to begin with the following configuration and a block size of just one transaction, as it is shown in Figure 3-1.

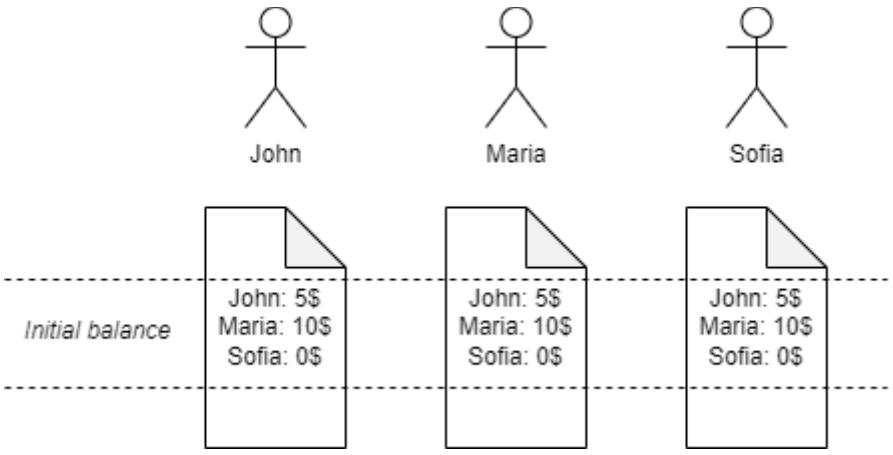


Image 3-1 Initial blockchain configuration example

Now, imagine John wants to transfer Sofia 3$ to, for example, pay off a debt. To do that, John writes into his own ledger the following transaction and signs it:

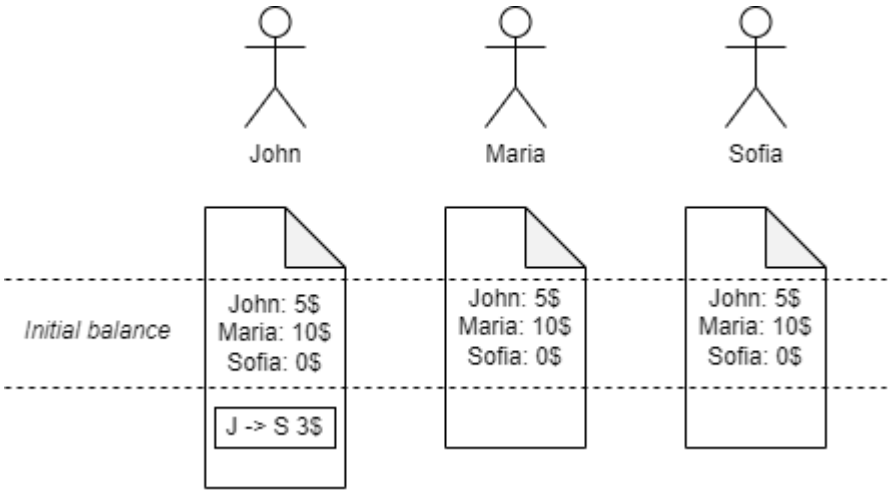


Image 3-2 John creates a new transaction in his own ledger

Now, Maria's and Sofia's ledgers are outdated with John's. Therefore, now John communicates his two peers about the new transaction. When they receive it, they verify that it is effectively signed by John and validate if he owns the enough amount of money to perform the transaction he wants. As both conditions are true, they write it down to their own ledger and update it:

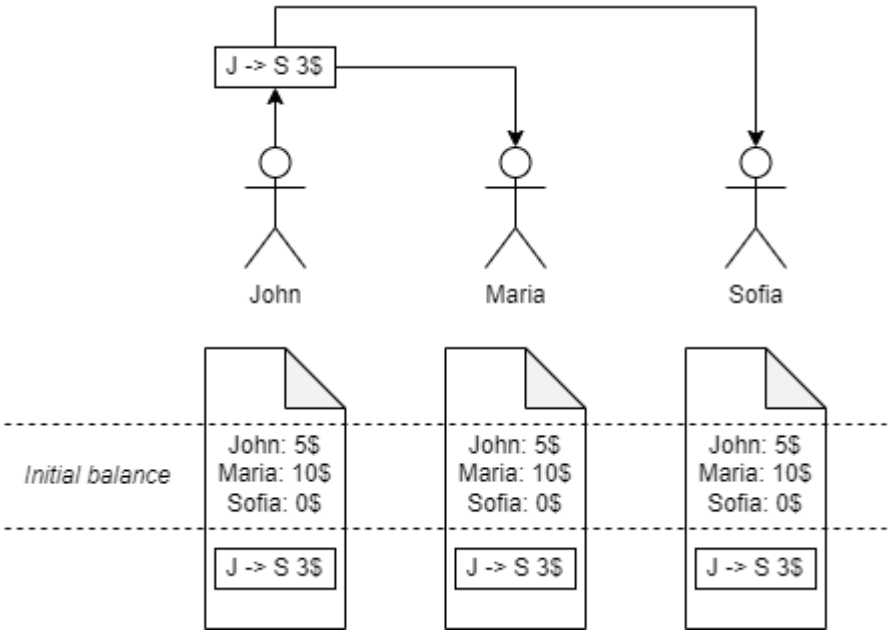


Image 3-3 John communicates the new transaction to his peers and each peer writes it down into their own ledger

Now, if Sofia wants to check her balance she would just have to add or subtract all transactions where she is implicated (just one) from his initial balance (0$). In this case, she starts with 0$ and receives 3$ from John, so now she has 3$ and John already paid his debt.

This process can continue endlessly with more transactions from any node:

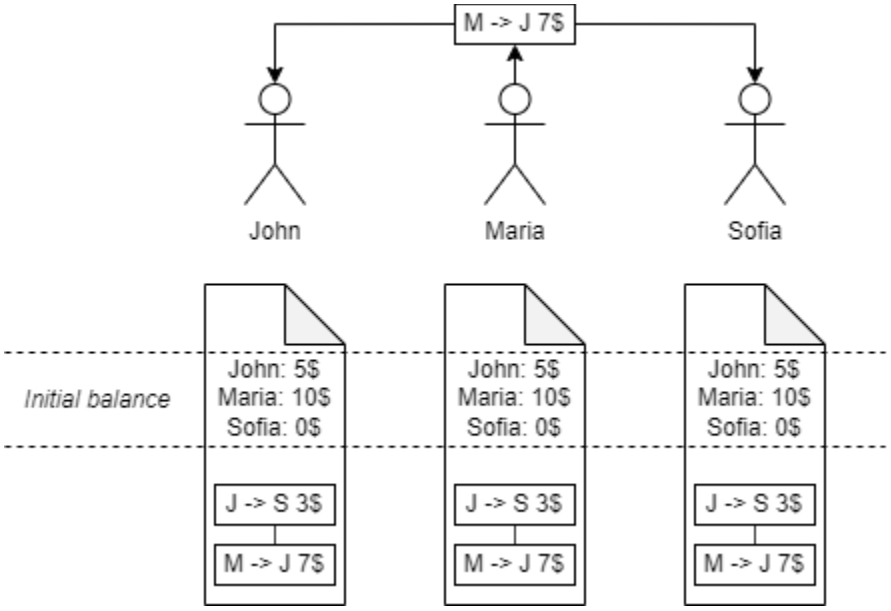


Image 3-4 Maria transfers 7$ to John

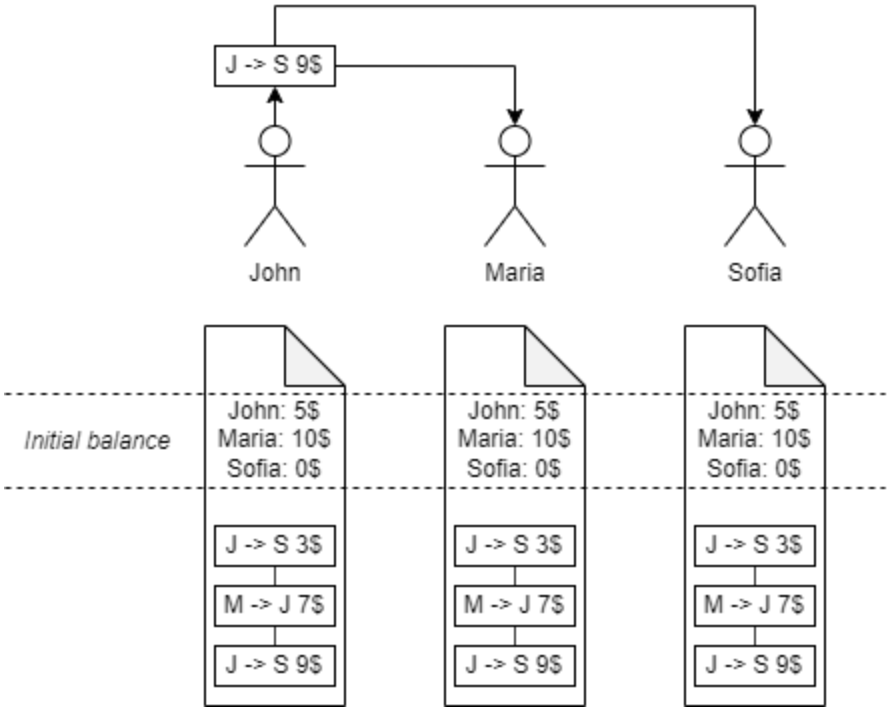


Image 3-5 John transfers 9$ to Sofia

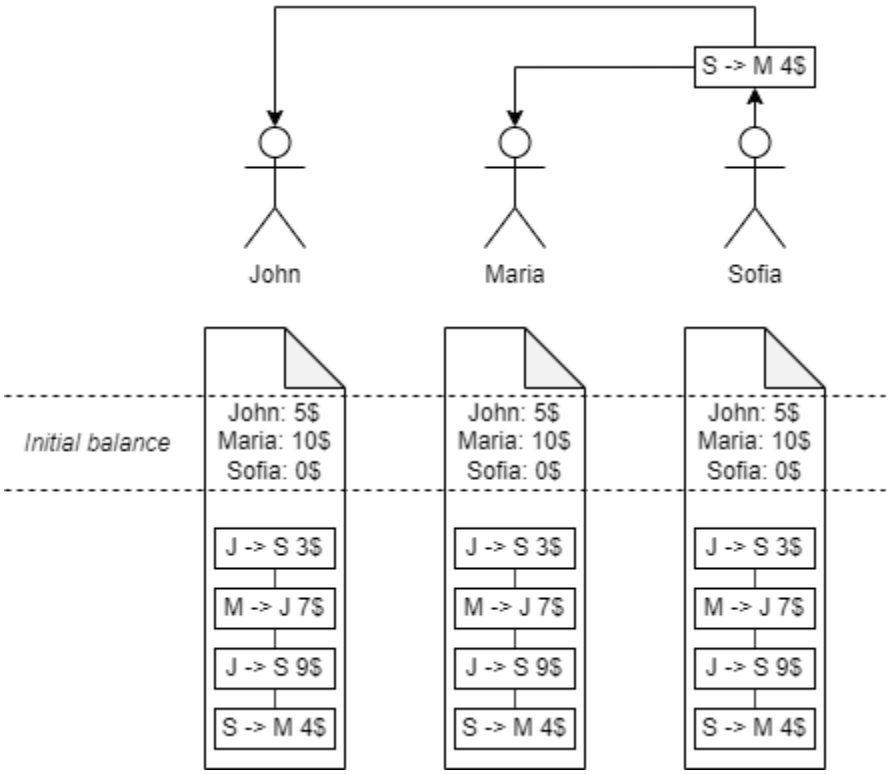


Image 3-6 Sofia transfers 4$ to Maria

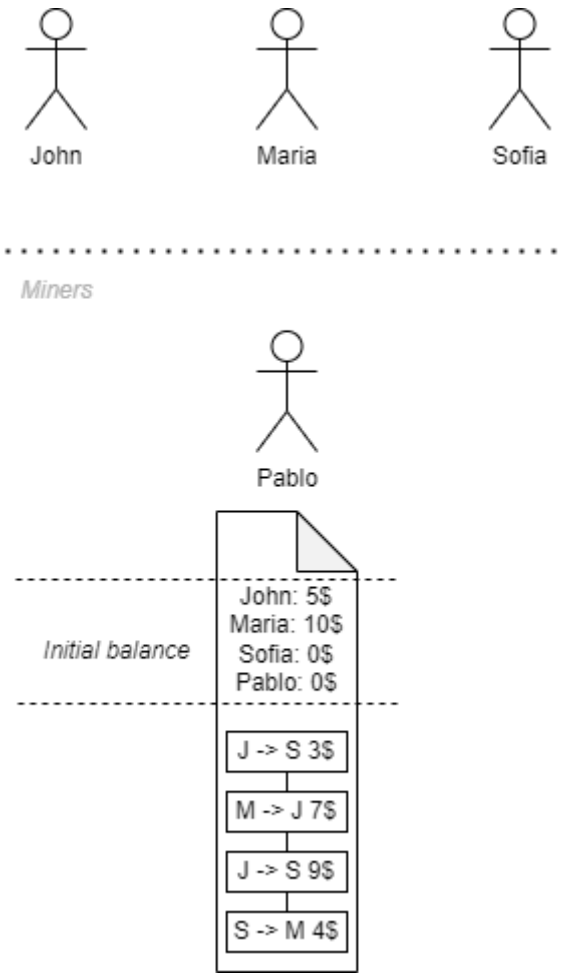
In this final state, John would have 0$, Maria 7$ and Sofia 8$.

It is important to note that, despite being friends, they do not trust each other when speaking in money terms. Thus, every node is verifying (so the transaction is sent from who it says it is sent), validating (so no node spends money that it does not own), calculating the hash function of the previous block (so each block is chained to the previous one) and writing all transactions they receive into their ledger (keeping record of all transactions). This process is called ***mining*** and is performed by a kind of nodes called ***miners***, and as can be seen, they perform a vital role on blockchains as they are who store all transactions, ensure their integrity, and overall keep the blockchain working correctly.

Now, let us assume that the three friends conclude that it is a waste of time being a miner and they have other more important tasks to attend to. So, they decide to hire someone else to play the role of miner, and for each transaction mined, they pay him with one dollar created out of nowhere. This is called a ***miner reward***.

So, they do as planned: they stop creating new transactions, check that every node ledger contains the same data, and give it to Pablo, the new miner they hired.

With these changes, the blockchain structure would be like this:



Now if any node wants to perform a transaction, they need to communicate it to Pablo first, and Pablo receives a reward from it:

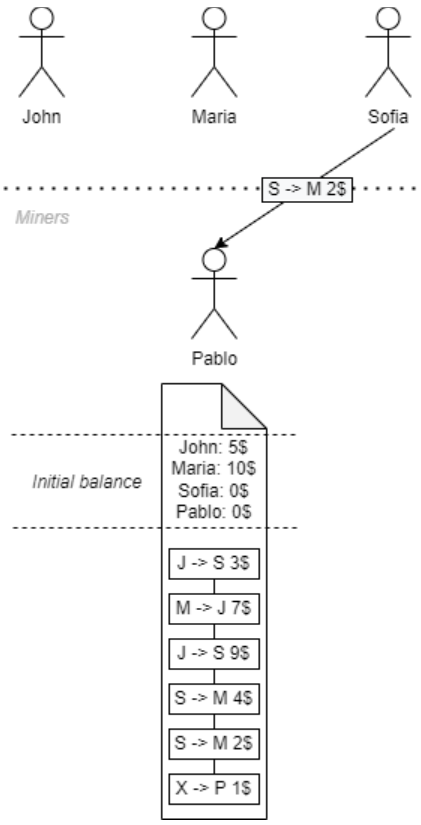


Image 3-7 Sofia communicates the transaction to Pablo and Pablo mines it and receives a reward

With this transaction, Sofia transferred 2$ to Maria and Pablo earned 1$ for verifying, validating, chaining and storing the block associated to it (he got that dollar from nowhere, X does not represent any node).

However, as it can be seen, this blockchain is built upon just one miner node that represents 100% of miners, so it is remarkably like a centralized authority. Pablo realizes this vulnerability, so he decides to go to exploit it. He goes to Maria, who has an ice cream shop, and buys an ice cream. Maria, who trusts blindly their blockchain economy system, accepts the payment and Pablo transfers 1$ to Maria (for the sake of the example, Pablo does not receive rewards for his own transactions):

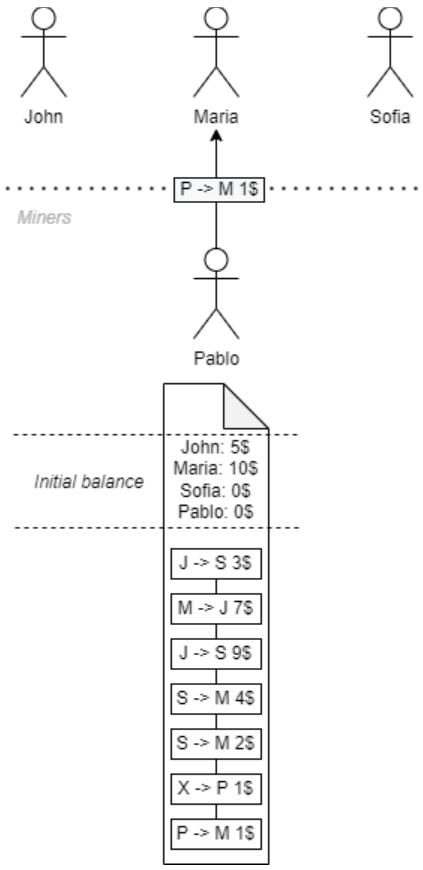


Image 3-8 Pablo sends 1$ to Maria and mines the block

Once Maria checks she received the payment, she gives the ice cream to Pablo. However, the next day, Pablo removes the last transaction from the Blockchain ledger and recovers his spent dollar.

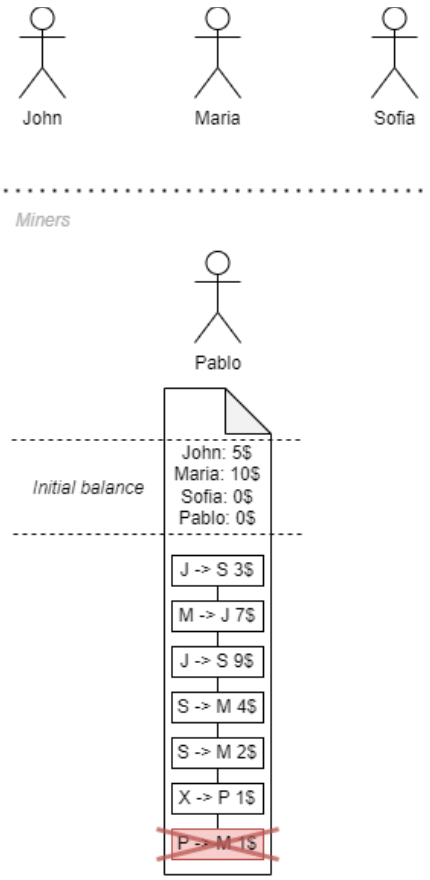


Image 3-9 Pablo removes the last block from the blockchain and recovers his dollar spent

As the three friends trust the system, nobody realizes this, so he keeps doing it for a long time. However, at some point they end up noticing his malicious actions, and fire him as a miner. They cannot prevent him from using the blockchain, so now Pablo is just a normal node like John, Maria and Sofia.

The three friends are shocked by this incident, so, in order to prevent it of happening again, this time they hire five miners and give each of them a copy of the ledger:

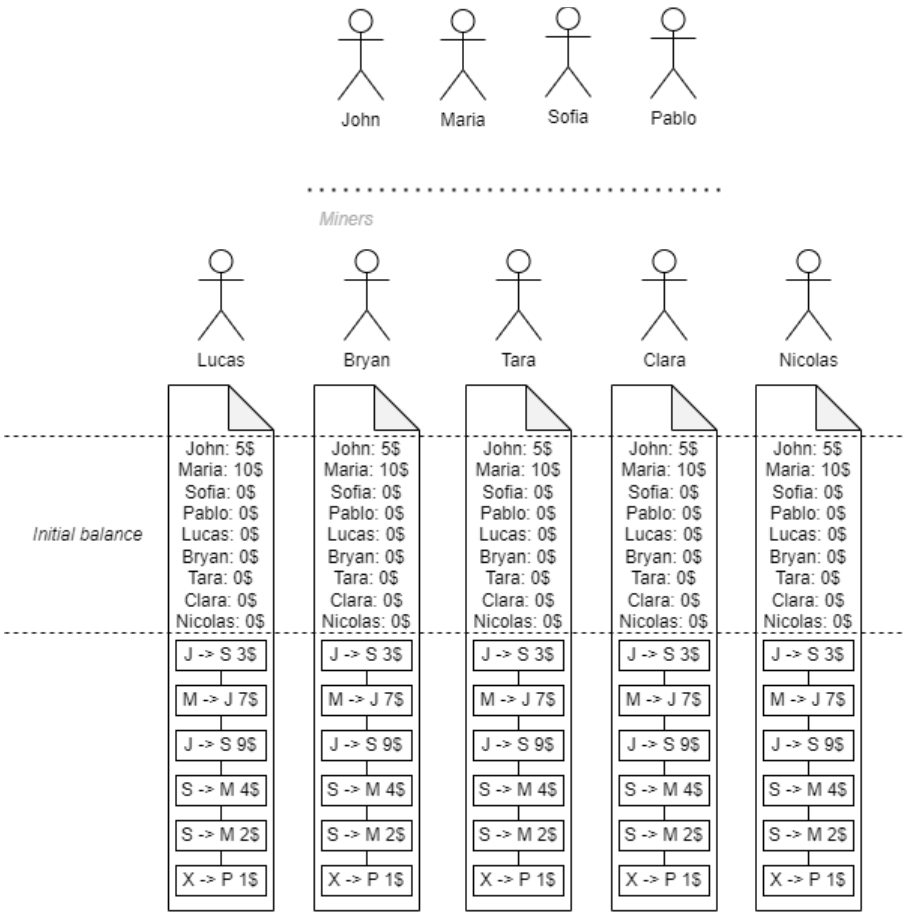


Image 3-10 Blockchain with 9 nodes, from which 5 are miners

Now, each time a node makes a transaction, it is transmitted to all miners. However, the reward just goes to the one who mines the block first, so they all hurry to be the first ones to verify, validate, link, and store it:

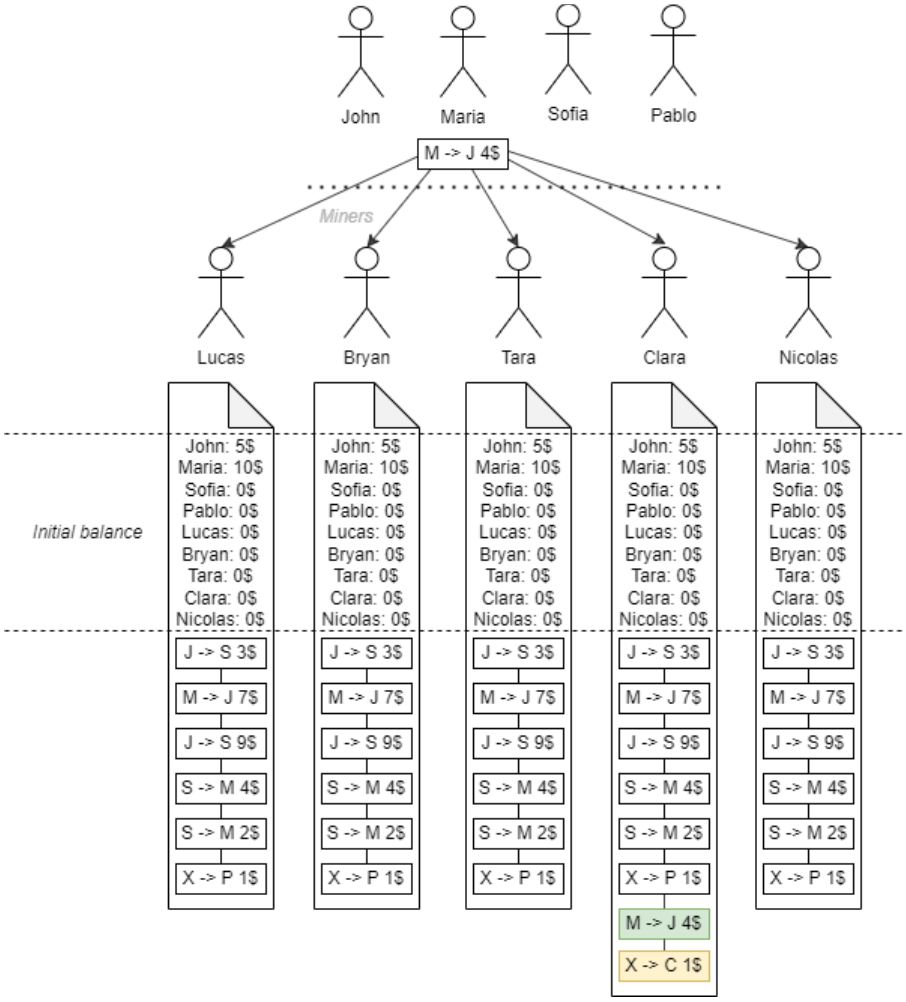


Image 3-11 Maria sends a transaction and Clara is the first one to mine it

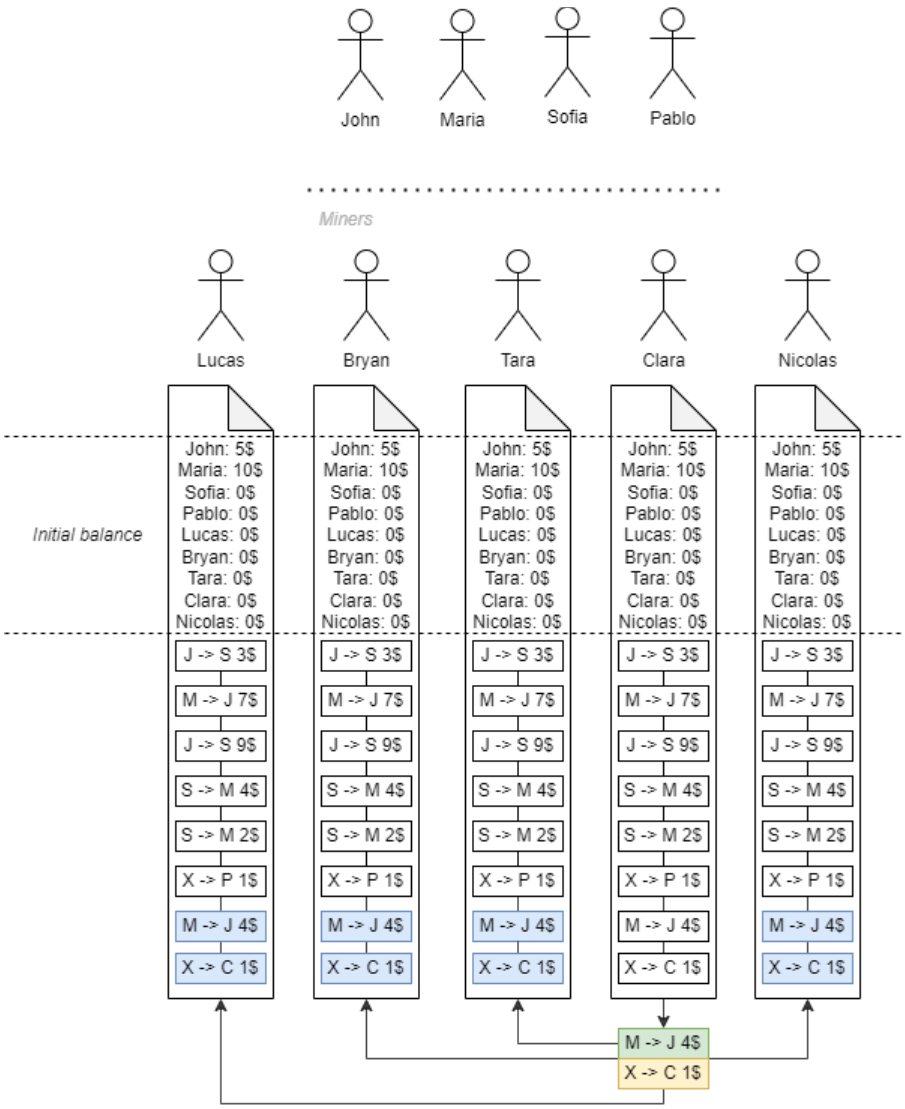


Image 3-12 Clara transmits the mined block to the rest of the miners so they can append it to their ledger.

In a real setting, there are many miners generating new blocks simultaneously and thus possibly broadcasting different blocks to other miners. A miner may actually receive several sequences of blocks that claim to be considered the last blocks in the chain, so the miner must choose one of the received sequences as the last blocks in order to create the next block after it. In that case, all members agree that if there is dispute about who mined the next block, each node will choose the longest current branch.

3.2 Blockchain general issues

Blockchain, as all technologies, has its downsides and it is not free of vulnerabilities, so in order to design a secure voting system with it, it is vital to understand it well.

Phishing attacks are the most common, with the Metamask pop-up incident as one of the most recent examples[16]. Popular cryptocurrencies websites, like Etherscan, CoinGecko and DexTools warned all users that they were aware of suspicious pop-ups appearing for visitors and advised them not to confirm any transactions based on pop-ups. In February, another phishing attack targeted NFT marketplace OpenSea and resulted in the theft of $1.7 million worth of NFTs from platform users[17].

However, one of the most dangerous attacks against a blockchain are the so called 51% attacks. When an individual or an organization takes the control of the 51% of the nodes of a blockchain, they gain the power to perform a DDoS attack or to directly remove blocks from the blockchain record. This is what Pablo did in the previous example.

Another attack that could be against a blockchain is a Sybil attack. Named after a person who had multiple personalities, Sybil attacks consist of a single node that acts like multiple ones. In this way, the node can easily take over a blockchain and perform a 51% attack. However, consensus protocols like proof of work or proof of stake makes Sybil attacks not viable.

In the following example it is going to be proved how a blockchain resists against tampering attacks when more of the 51% of nodes are honest and not malicious.

---

16 https://www.theverge.com/2022/5/13/23071786/etherscan-coingecko-crypto-phishing-ad-popup-coinzilla-metamask

17 https://finance.yahoo.com/news/crypto-website-phishing-attack-targets-035435753.html?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAI-i7sfGR2mTwgh1rmrXScsYw47lj0vtKxkJi03knYUniXEb8GxACUnUPHe0eDjNAWpQSJtplv1T4sFAnQ6JhYNZ0MJwfXipNzL23_hLsYVdszhpLDpJAFdxRKaMMZ_O_i0a_1CF-jjmgCtBjAlvYNDnR8Dh507cZbXNflCwQbND

Clara, who heard Pablo's story, wants to try the same trick he did. She goes to buy an ice cream to Maria's shop and transfers 1$ to her:

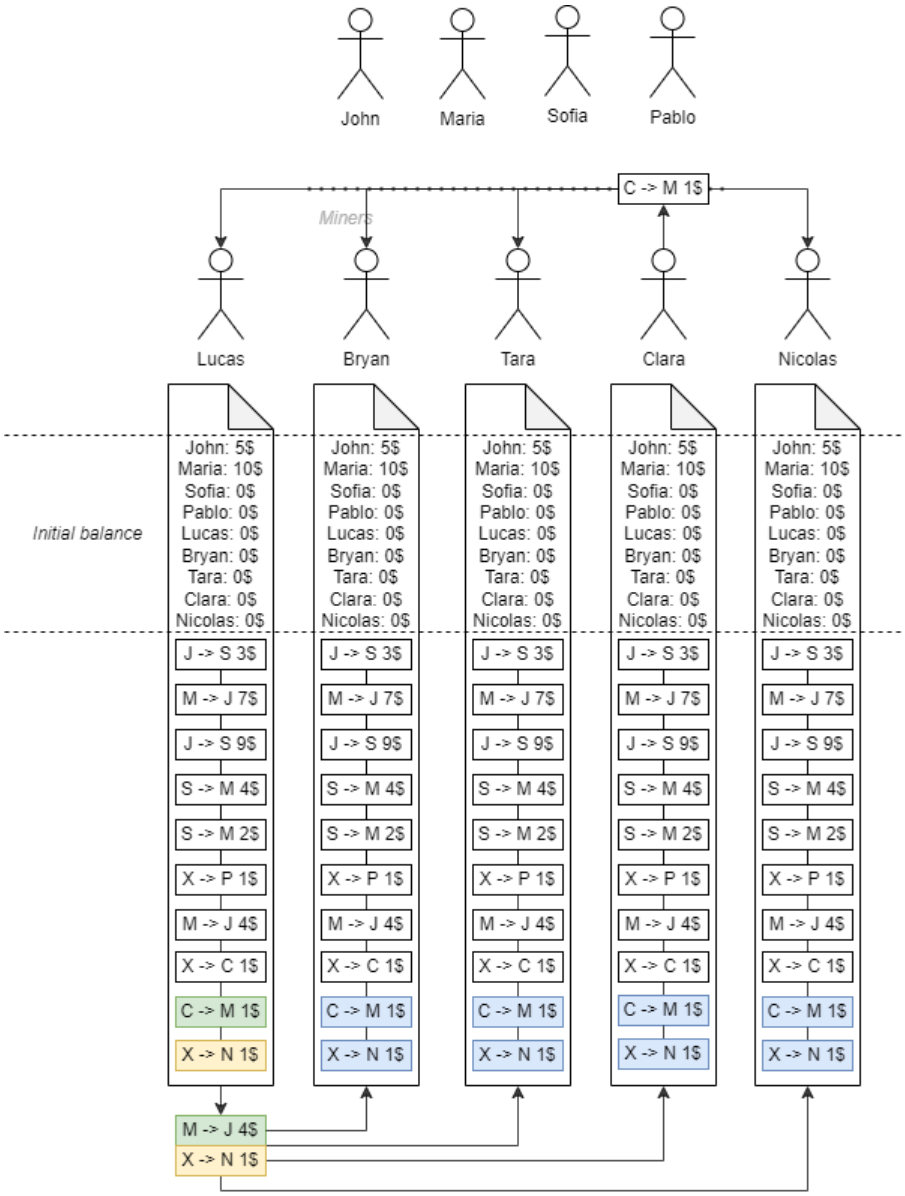
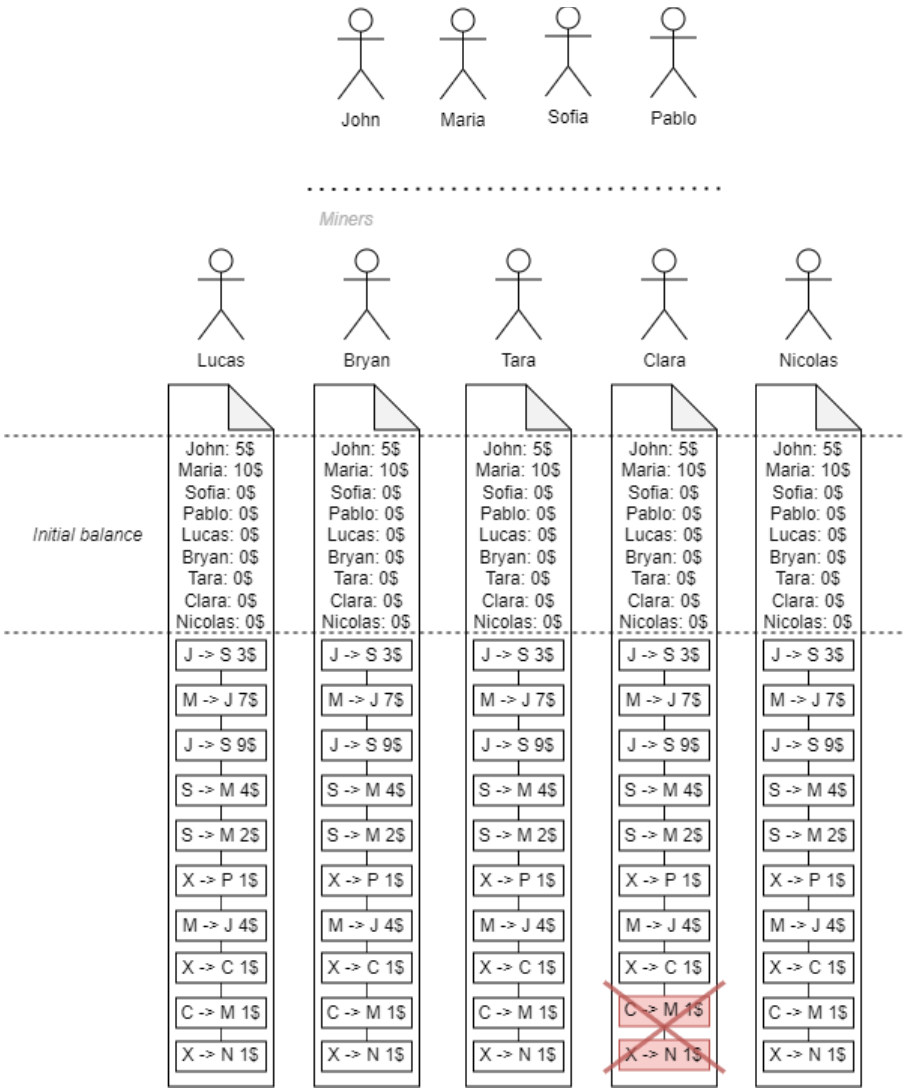


Image 3-13 Clara sends 1$ to Maria. Nicolas achieves to mine the block first and gets the reward.

Now she removes the last two transactions from its ledger to erase any trace of her trick:



She knows that, in order to convince the other miners from her modification, she needs to keep her version of the chain as the longest branch. By doing that, the rest of the miners will always continue the longest branch.

So, when a new transaction is sent, two situations may occur:

- Clara achieves to mine it first, so she sends the mined block to the rest of the miners, they receive that block and start a new branch of the chain from the transaction Clara says that was the last one. In this scenario, she earnt one more dollar for mining and got free ice cream.

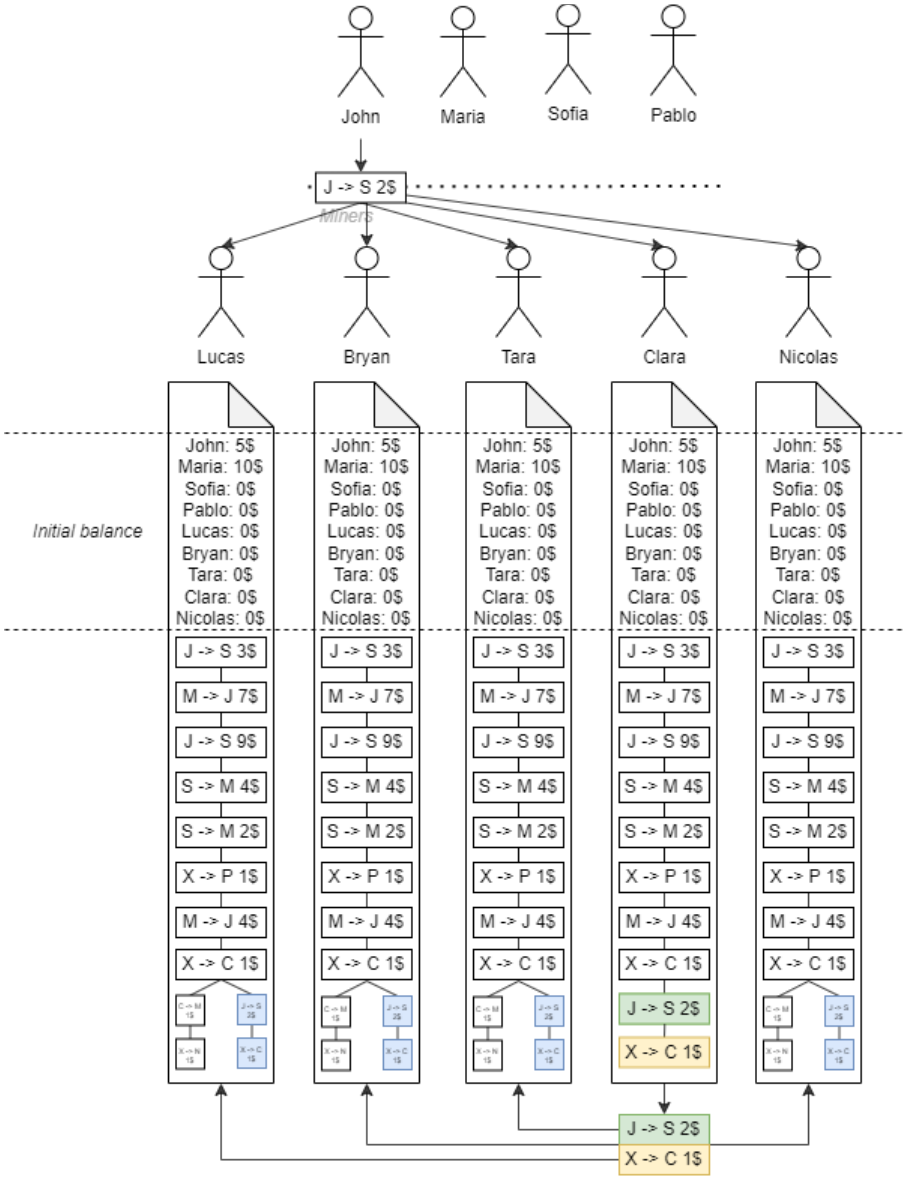


Image 3-14 Clara succeeds and achieves to write her block into the other miners ledgers.

- Clara does not achieve to mine it first, so the rest of miners extend their main chain branch two more blocks and continue to operate normally. Now Clara needs to mine first 4 blocks instead of two.

So, in order to Clara maintain her malicious act, she would have to mine first all blocks from the transaction she wants to remove until she achieves to build the longest branch of all and maintain her fraudulent branch as the largest. However, to do so, she would need to have, at least, the same computational power as half of the rest of the miners together, which is expensive and hard to accomplish.

This kind of attacks are called 51% attacks and are one of the most known and dangerous vulnerabilities of a blockchain. Therefore, it can be concluded that the number of nodes of a blockchain is directly related to its trustworthiness, as more nodes more difficult is to manipulate the main chain.

3.3 Types of Blockchains

3.3.1 Types of Blockchains based on access and permissions[18]

There are two big groups of Blockchains, depicted in Figure 3-15:

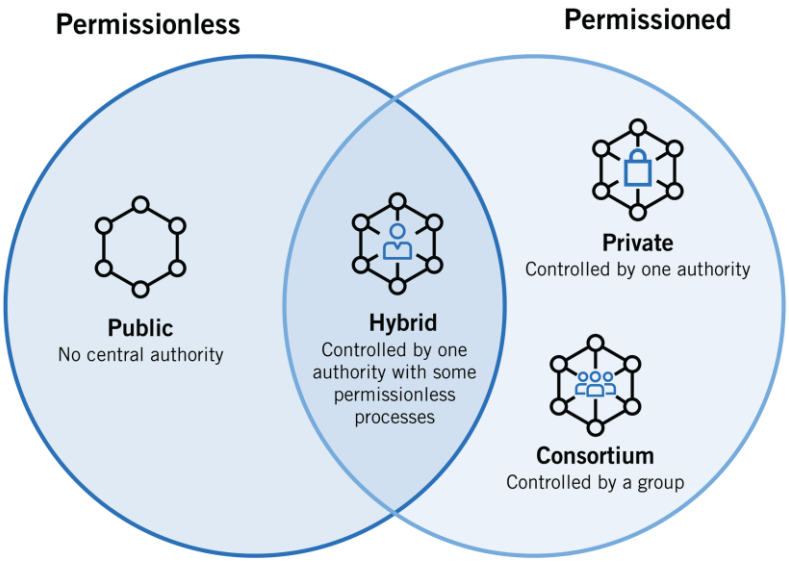


Image 3-15 Types of blockchain

- **Permissionless Blockchains**: These blockchains allow any user to access the network and does not apply any restrictions to nodes. Bitcoin and Ethereum are two examples.
- **Permissioned Blockchains**: They restrict access to the network; not every user can participate. Also, they might apply restrictions to nodes. They tend to be more efficient.

[18] For this section 3.3.1, we follow the contents of this website: https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between

Inside of these two general groups, there are four types of structures:

3.3.1.1 Public blockchains

Public blockchains are permissionless by nature. They allow anyone to join them and are completely decentralized. Also, all nodes have the same rights to access the blockchain, create new blocks of data, and become miners. Some examples are Bitcoin, Ethereum or Solana.

3.3.1.2 Private blockchains

These blockchains are not completely decentralized like the previous ones as they are managed by a central authority or organization. They control the access and permissions of nodes, like who can read, who can send transactions or who can validate blocks. An example is Hyperledger Fabric[19].

3.3.1.3 Consortium blockchains

These type of blockchains are governed by a group of organizations instead of a single entity, which makes them enjoy more decentralization than private blockchains and, therefore, more security as there are more nodes in the network. An example of these blockchains is Global Shipping Business Network Consortium, a non-profit blockchain infrastructure which aims to digitalize the shipping industry.

3.3.1.4 Hybrid blockchains

These blockchains combine both public and private properties. They are controlled by a single organization and transactions and records are made private, but they are still verifiable when needed. Nodes have to be granted access but all share the same rights.

3.3.2 Types of blockchains based on technology

As mentioned earlier, currently exist many blockchain projects, each one of them with their own technology, philosophy and roadmap.

[19] https://hyperledger-fabric.readthedocs.io/en/release-2.4/whatis.html

Smart contracts are a powerful technology exclusive of blockchains that allows the execution of code. This means that any behaviour can be programmed into a blockchain, but with some restrictions. One of them, which separates blockchains into two types, is Touring-completeness, which refers to the capability of a system to support logical programming with loops:

3.3.2.1 Classical blockchain (not Touring-complete blockchains)

"Classical blockchains" is a term that refers to the first type of blockchains created, following Bitcoin philosophy. These blockchains are made exclusively to register transactions and do not allow the execution of Touring-complete programs inside them. This is made to avoid DDoS attacks, as Touring-complete programs may enter in an infinite loop and block all nodes in a blockchain.

3.3.2.2 Touring-complete blockchains (smart contracts blockchains)

A Touring-complete blockchain is a programmable Blockchain that allows the secured execution of non-trivial scripts (roughly speaking, scripts containing loops) called smart contracts inside the blockchain. The most famous example is Ethereum with its programming language Solidity, but another minor blockchains like Solana also allow Touring-completeness, with the programming language Ruby or C.

3.4 Blockchain consensus

The mining process in a blockchain is vital for the network security. It is the base of its correct functioning and without it blockchain technology would just be not possible. Therefore, the blockchain system needs to encourage mining but avoiding at the same time the previously mentioned sybil attack problem.

It is quite easy to encourage nodes to mine blocks: blockchain systems rewards them with a certain number of coins created out of nowhere. This is how it is done in roughly all existing public blockchains.

However, to ensure that mining is done correctly is not a trivial task as anyone can tamper the data. A general agreement must be reached between nodes so they set up a mechanism through they can agree on the next state of the blockchain without incurring into previously seen attacks like sybil or 51% attacks. Consensus

mechanisms allow distributed systems to work together and stay secure making this type of attacks unfeasible forcing nodes to "prove" that their mining process is honest[20]. Diverse mechanisms have been engineered to solve this security problem:

3.4.1 Proof of work

Proof of work (PoW)[21] (Nakamoto, 2008) can be explained as a calculation problem race. Miners compete to calculate the hash of the next block of the chain, and the one who solves it first gets the reward. It was the consensus used in the example used to explain how blockchain works.

PoW is kept secure by the fact that you would need 51% of the network's computing power to defraud the chain. This would require enormous investments in equipment and energy, so you are likely to spend more than you would gain.

PoW, however, is quite slow as the security relies on the calculation difficulty. Bitcoin, for example, uses PoW to operate, and their transaction times are around 10 minutes.

Another known issue of PoW is its huge energy consumption. To rely on computational power implies also to rely on the energy needed to obtain that calculation. Therefore, Bitcoin, Ethereum, and other currents PoW blockchains have been criticized for being too contaminating for the environment, surpassing even some countries, like Argentina or Finland (University of Cambridge, 2022).

[20] https://ethereum.org/en/developers/docs/consensus-mechanisms/

[21] https://ethereum.org/en/developers/docs/consensus-mechanisms/#proof-of-work

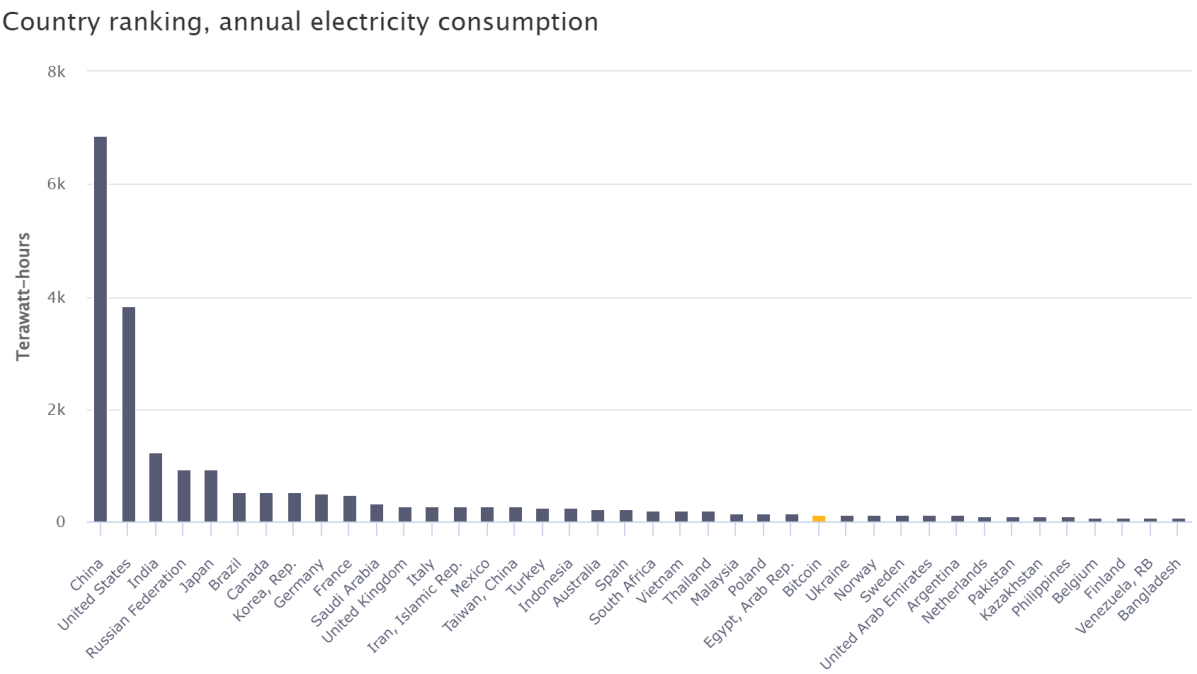


Image 3-16 Cambridge Bitcoin Electriciy Consumption comparison

3.4.2 Proof of stake[22]

In proof of stake (PoS) miners are called validators. Validators have the correspondent blockchain cryptocurrency staked to participate in the system. A validator is chosen randomly to create new blocks, share them with the network and earn rewards. There is no need of computation-intensive calculations.

Ethereum plans to transition to proof of stake this year. The staking process is already open, and the price to be a full validator is 32 Ether (currently, around 64640$).

3.4.3 Proof of authority

This consensus type is characteristic of private/permissioned blockchains. Like PoS, validators do not need to perform complicated calculations in order to create new blocks. In this case, each node proves their honesty by putting their reputation at stake by making public their identity with, for example, a X.509 certificate emitted by a central authority.

[22] https://ethereum.org/en/developers/docs/consensus-mechanisms/#proof-of-stake

PoA, as previously mentioned, is commonly used in private blockchains or enterprise blockchains (like Enterprise Ethereum Alliance[23] or Hyperledger Fabric).

3.5 Blockchain voting cases

We have seen in previous sections an overview of how blockchain technology works as a decentralized way of recording transactions in a ledger. Now we see some existing applications of this technology to real voting cases.

3.5.1 Moscow

In December 2017, Active Citizen, the program that Moscow's citizens used for holding open online referenda with community purposes since 2014, started using blockchain technology for voting through an Ethereum private network and made the results publicly auditable. The platform has almost two million registered users and 88 million votes have been cast.

They also launched Digital Home in 2018, an online service that uses Active Citizen that allows high-rises neighbours to vote and communicate on community issues, like whether to replace the building entrance door or hire a new management company[24].

According to a French cryptography expert, Active Citizen was quite easy to hack due to the poor encryption algorithms that were used (ElGamal encryption system) and to some security vulnerabilities (Gaudry & Golovnev, 2019) although he mentions that these issues were fixed later.

Polls categories in this system vary widely: education, transportation, tourism, culture… Also, it is important to note that voting is rewarded in the form of points that can be exchanged for services, such as Metro rides or museum tickets. This feature is

[23] https://entethalliance.github.io/client-spec/chainspec.html

[24] https://www.coindesk.com/markets/2018/03/15/moscows-blockchain-voting-platform-adds-service-for-high-rise-neighbors/

highly problematic as it may push citizens to vote just for getting rewards and compromise the results objectivity (Gritsenko & Indukaev, 2021).

The following information was extracted from their official website[25].

AC is composed of 3 layers:

- The frontend or user interface (implemented as a web and mobile application)
- The backend (composed of a centralized server and a centralized database)
- The Blockchain environment

[25] https://ag.mos.ru/blockchain

Image 3-17 Active Citizen Architecture. Figure downloaded from their website

The backend and the Blockchain layers communicate through an integration module (by means of web3 libraries). Also, the Blockchain can be audited through a module of independent counting.

Now that we have seen its structure, how does Active Citizen keep votes principles?

3.5.1.1 Universal vote

As long as you have a muscovite phone number, you can register into the application and participate. So, as long as they have a mobile, every muscovite is allowed to access the system.

However, certain polls are restricted to a specific group of users. For example, some polls are restricted to a certain area so, in this way, someone who lives far away from there cannot influence over a poll that almost does not affect him/her. This is explained in the "Why do I need to enter my exact address?" section of their FAQ[26]:

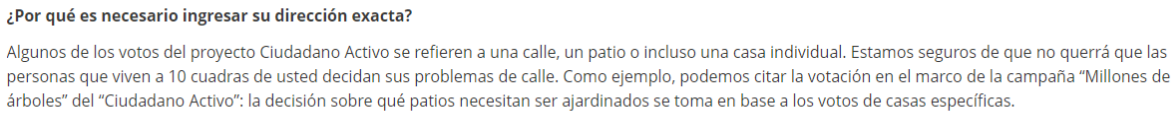


Image 3-18 Active Citizen FAQ

3.5.1.2 Free vote

You have complete control of the choices you pick in the application.

It is important to note that there are some guides and articles in some polls that "introduce" the citizens into the topic that it is being discussed. Depending on how these articles are written, they can be influencing the reader or not (Gritsenko & Indukaev, 2021). Nevertheless, this is not the subject of this survey.

3.5.1.3 Equal and unique vote

First, to access the platform, the user must complete a registration process. To validate this registration and confirm that the user is a Muscovite the user must enter his phone number and type the secret code he receives via SMS:

Image 3-19 Active Citizen register phone number

[26] https://ag.mos.ru/faq#section0

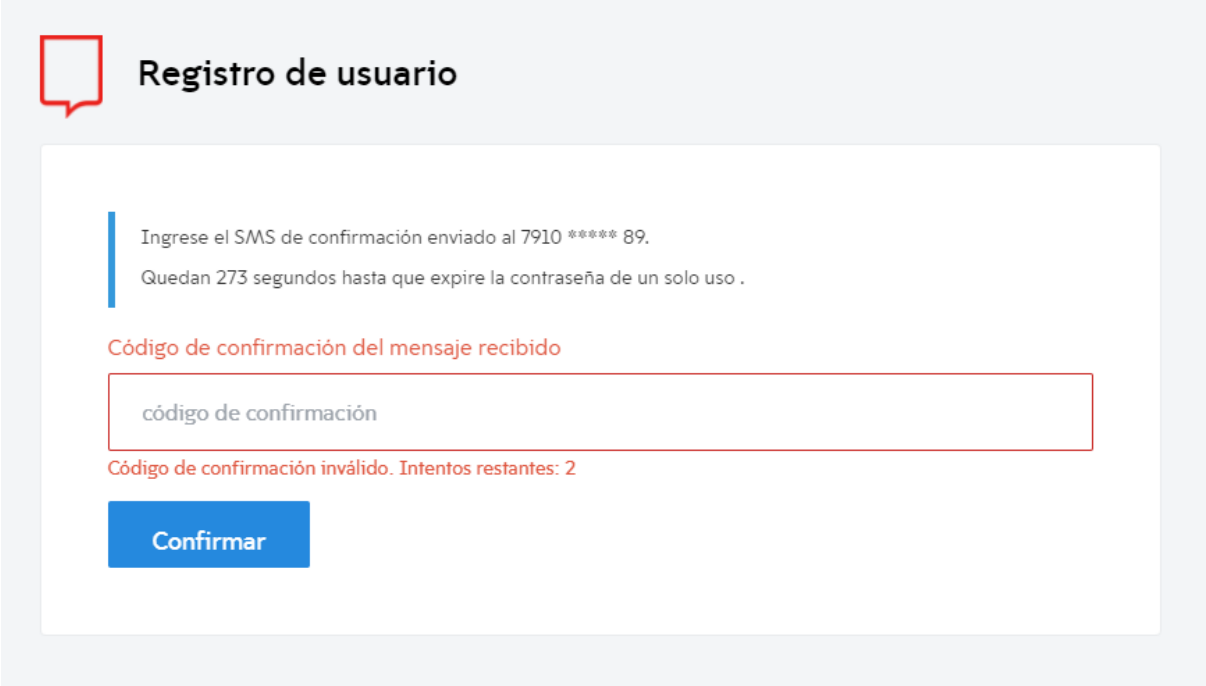


Image 3-20 Active Citizen SMS code verification for registry

This kind of registration process brings us to the following question: can a user register twice if he/she owns more than one phone number? If so, this vulnerates the equal and unique vote principle. However, as mentioned earlier, some polls require to introduce the voter address, so in these cases it is more likely to not vulnerate the unique vote principle (Gritsenko & Indukaev, 2021)

3.5.1.4 Direct vote

Each user is the unique owner of their account, as long as their accounts are not attacked or phished, and each account has access to the real polls. This means that every user can directly cast their votes into the polls they want to participate in.

3.5.1.5 Secret vote

When each voter is registered, a unique and random identifier that represents that voter is created. This ID is used to vote and is public for everyone, but the correspondence between the real person and the ID is kept secret (as long as the user does not reveal it). By analyzing the contract code published in their official

GitHub[27], it can be seen that this ID is NOT generated in the blockchain smart contracts themselves, but in the Active Citizen backend/servers. This can lead to the violation of the secret vote principle if the backend is successfully attacked and all the IDs of the voters are leaked. Also, AC explicitly specifies that the user ID is always the same, so if it is leaked, all the votes that the user cast will be exposed.

So, from all of this we can infer that all the logic and relations are done in the backend, while the Blockchain layer is only used for transparency purposes.

3.5.1.6 Transparency of results

The Blockchain environment used to store the votes ensures, by nature, the results transparency. However, this property is only fulfilled as long as the results are pushed into the blockchain itself, as it is the backend who performs the main logic of the system. So, again, centralization makes the system vulnerable as is single-point failure.

3.5.1.7 Vote checking

As each user knows his own unique ID, they can use it to check directly in the blockchain how that vote was actually cast.

3.5.2 Gyeonggi-do

In March 2017, the south Korean province of Gyeonggi-do employed a blockchain system to vote on the Ddabok Community Support Project. Nine thousand residents voted using a blockchain platform developed by the south-Korean financial-technology start-up 'Blocko' that included smart contracts (Kshetri & Voas, 2018).

No further information was founded about the application used for this voting case nor the source code of the smart contracts used. Nonetheless, this company has a product named "Pikkle" in their official website[28] which may have been used for this purpose. It is a blockchain-based voting service built over the hybrid blockchain

[27] https://github.com/moscow-technologies/ag-blockchain

[28] https://en.blocko.io/services/pikkle/

AERGO that was used in the 48th Annual Korea Broadcasting Awards in which it garnered 190.000 votes from public audience in real time. It was also used in the Seoul Drama Awards, where another 620.000 votes were cast in real time.

Pikkle is a free app that can be downloaded at the Play Store[29]. Users must register into the app and scan a QR code to vote. Depending on the poll, special points that recharge over time may be required. However, the app is not designed to election purposes, as registration process is made by email and any user can have easily multiple votes in the app. Nevertheless, the blockchain environment used (AERGO) may be suitable for holding elections as it is described on the website.

3.5.3 Sierra Leone

In the Sierra Leone's March 2018 elections, a Swiss blockchain start-up, Agora, provided a partial tally of election results. Agora was one of the accredited observers that provided and independent count for comparison (Kshetri & Voas, 2018).

Agora has built a blockchain ecosystem composed of five technology layers: the Bulletin Board blockchain, Cotena, the Bitcoin blockchain, the Valeda network and Votapp. Each one of these layers communicate with each other at various instances throughout the election process to provide a cryptographically secure voting environment with auditable proofs. This architecture is explained in detail in their whitepaper (Agora).

[29] https://play.google.com/store/apps/details?id=io.blocko.pikkle.apps&gl=ES

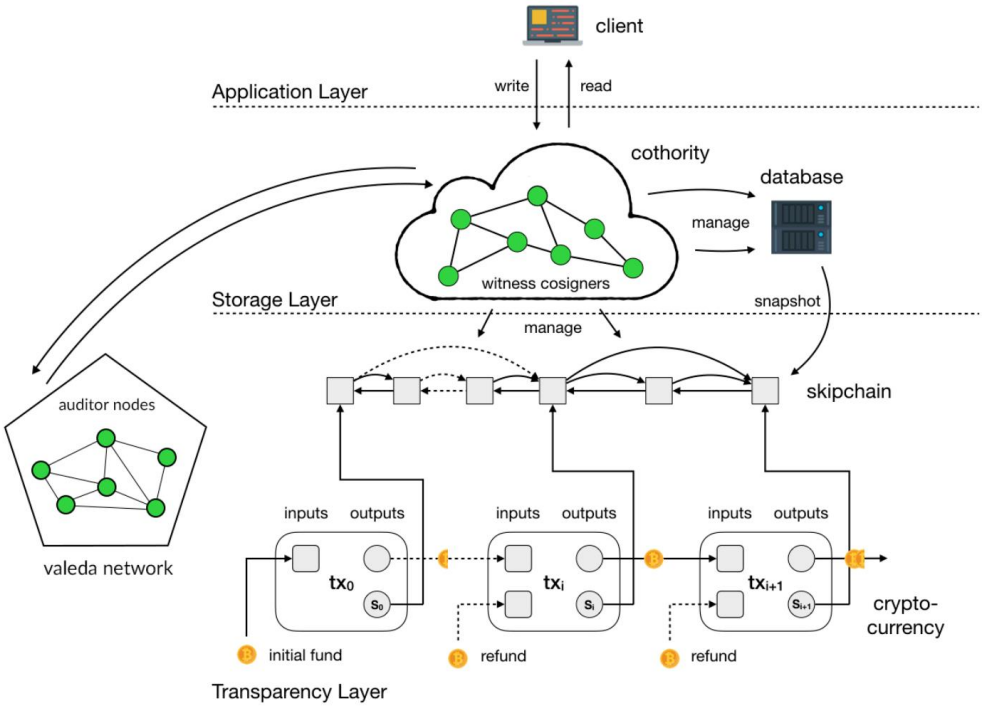


Image 3-21 Agora architecture

The whitepaper also says that in order to ensure vote anonymization each casted ballot by a voter is encrypted from the device the voter is using. When the election finishes, they run all collected encrypted ballots through a mixing network that sequentially re-encrypts the given dataset multiple times, where the correctness of each re-encryption is attested by zero-knowledge proofs. This shuffling ensures vote anonymization.

Nonetheless, it is more important to detach each system user ID from their real identity in the authentication process than the anonymization done afterwards. The reason for this is that it is indeed on the authentication process where voters identities can be filtered. In their whitepaper they state that they rely on voting external administrations to select an identity management system and to provide a mechanism to authenticate voters. Thus, at the end, it is just enough for an attacker to compromise that authentication mechanism and save all relationship between voters identities and system IDs to violate the secret vote principle.

3.5.4 Academic voting proposal for blockchain-based voting

The development of blockchain systems have not only attracted different companies and countries, but also researchers from academia.

Friðrik Þ. Hjálmarsson and Gunnlaugur K. Hreiðarsson are two researchers from the Reykjavik University in Iceland who wrote a report (Þ. Hjálmarsson, K. Hreiðarsson, Hamdaqa, & Hjálmtýsson, 2018) where they proposed a blockchain-based e-voting system for elections. Their system works as follows:

- It is built up in a private blockchain network with smart contracts.
- Nodes use the proof of authority consensus protocol in order to agree on the blockchain state.
- Nodes are managed exclusively by the government administration.
- The voter registration and authentication process are done via an external centralized app, called Auðkenni, where users log in with and ID and a PIN number. Auðkenni is an Icelandic service provider for identity verification.
- Once voters are registered, they are provided with a unique blockchain wallet/account. Voters cast their votes through their wallet via function transaction calls to the smart contract interfaces of the blockchain. Once they cast their vote and it has been validated by nodes, they receive back a transaction ID for later verification of their vote.
- The tallying phase is done on the fly in the smart contracts. When the election is over, the results are published.
- Voters can verify that their vote was correctly casted by going to their government official and presenting their transaction ID and authenticating with their user ID and PIN.

The system architecture and flow are shown in Figure 3-21:

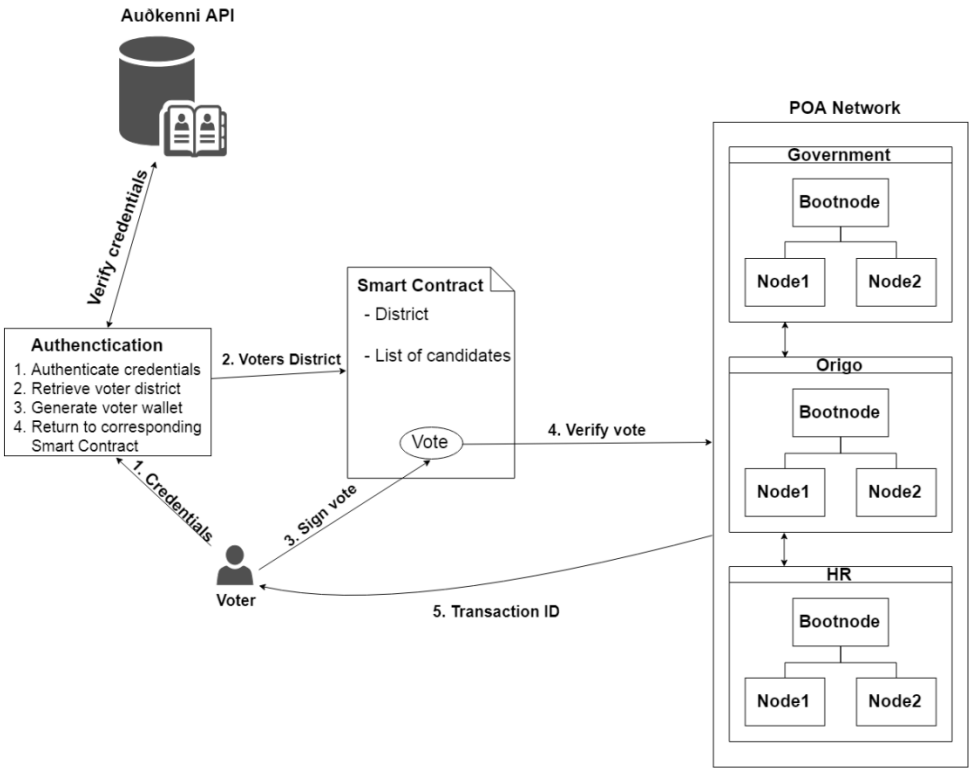


Image 3-21 Academic blockchain e-voting system architecture, obtained from (Þ. Hjálmarsson, K. Hreiðarsson, Hamdaqa, & Hjálmtýsson, 2018)

However, this system has the following flaws:

- Having government-exclusive validator nodes does not make the network decentralised enough, as a corrupt administration could easily control the majority of nodes and perform a 51% attack to the network and manipulate the election results, or directly stop the election process.
- The authentication service is a single point failure in the system as it is centralized and runs on an opaque backend server which generates the user wallets and could be easily storing the relationship between voters and wallets, thus violating the secret vote principle.
- As it can be seen in the smart contract implementation shown in their article (code excerpt in Figure 3-22), the system does not encrypt voters' ballots, instead they have a counter for each political party that increments with each vote. Thus, even if user nodes do not have permission to interact with the smart contract functions, results are

always publicly accessible as blockchain storage variables: even if they are declared as private, they can be read from the outside with external applications accessing the block they reside in.

```
function vote(uint candidate) public{
    require(!voters[msg.sender]);
    if(now >
    candidates[candidate].expirationDate){
        revert();
    }
    candidates[candidate].voteCount += 1;
    voters[msg.sender] = true;
}
```

Image 3-22 Solidity voting function for the academic proposed e-voting system

To sum up, even with its flaws, this system is an interesting and promising proposal. Therefore, the system that we propose in Section 4 is based on this system but trying to solve the problems exposed.

3.6 Main characteristics of a blockchain based voting system

Summarizing the previous sections, we have seen that all approaches have relevant issues:

- Active Citizen relies too much on the backend side, that is a single point of failure and thus vulnerable to attacks. Also, it stores the relationships between voters and system user IDs, which violates the secret vote principle.

- Pikkle is a commercial application which is not designed for classic election purposes as it allows users to register multiple times with different emails and allows, in some polls, to vote multiple times, violating the unique vote principle.

- Agora presents a sophisticated multi-layer blockchain environment which offers unique properties: auditable side-chain, persistence and security proportioned by the Bitcoin blockchain, skipchain built on top of the Bitcoin blockchain, ballot anonymization by encryption shuffling,

among others properties. However, they fail in a vital phase of an election, as they rely on an external administration: the authentication phase. By doing this, they are compromising their voter's identities, thus threatening the secret vote principle.

- The academic proposal has diverse problems: the blockchain network security relies entirely on government-controlled validator nodes, the authentication service is centralised and the results are always accessible.

Therefore, taking as a base the blockchain voting system cases studied in the previous points, it is important to extract the main characteristics that a blockchain-based voting system must have in order to preserve all voting rights:

- The blockchain network must be private and use PoA consensus in order to reduce cost and time.

- Nodes must not be controlled exclusively by government administrations.

- The authentication method must guarantee that just eligible voters can access the system. However, this method must guarantee the secret vote principle: it must be impossible for administrations to track a user back to its real identity. In order to do so, administrations must not have the possibility of storing user accounts in their backend servers along their real identities.

- Ballots must be encrypted to guarantee that results do not go public before the end of the election, thus not influencing voter choices.

4. Proposal of a blockchain-based voting system

We have seen in the previous chapter real-life voting case studies that made the most of blockchain properties in order to guarantee transparency and traceability of votes. However, most of them ended up incurring into centralization methods that may put in risk voter privacy.

In this section we are going to design a methodology and a blockchain based voting system that solves already known flaws and that complies with voting rights mentioned in previous chapters.

The system is going to be built on the Ethereum blockchain with its language, Solidity, for smart contracts programming.

4.1 Voting phases

First of all, in order to design the system, it is important to define the voting flow. However, before that, it is important to note that not all phases can be decentralized, as **we are trying to decentralize a centralized system by nature**; voter authentication and vote obtention must remain centralized as it is the central authority who determines who can vote and who provides the means to do it.

4.1.1 Election creation

The government summons new elections. In this phase, administrations organise themselves, it is determined the election date, the election duration, the political parties involved, and the eligible citizens from the census.

Once all preparations are completed, the system must be initialized with all said parameters. The system behaviour and how the elections will perform must be fully defined through smart contracts in the blockchain so anybody can modify them.

4.1.2 Voter authentication

In this phase the citizens from the country have to identify themselves in order to get access into the system. A proof of identity is required for voters, like a physical

national citizen ID or a digital certificate, like X509 certificates. The government needs a verification service to securely authenticate the provided proof of identity.

This is the only phase, along with the next one, where centralization is the only option as a central authority, like the country administration, is in charge of controlling access to the system.

4.1.3 Ballot obtention

All previously authenticated citizens now have to be authorised to vote into the system. We refer to this phase "ballot obtention".

The vote distribution must be done in a way that, when the vote is provided to a citizen, it is detached from their identity from the beginning, and it cannot be traced back.

In order to do this, a central authority creates authorised Ethereum accounts addresses and provides them to authenticated voters. These accounts are allowed to cast just a single vote into the election smart contract. As they are standard Ethereum addresses and therefore composed of 40 hexadecimal numbers, they do not have any meaning related to the voter's identity. An example of an account address would be: *0xb794f5ea0ba31114ce839613ffdba74279579263*.

These accounts can be handled by the system in two different ways:

- **As permanent accounts**: each voter is given just one account when they authenticate for the first time, and it can be used across all elections. In each new election the account receives the capability of casting just one vote, and that capability expires at the end of the election.
- **As single-use accounts**: in each new election, voters have to authenticate themselves and then they receive a fresh new account with the capability of casting a single vote. This account can be used only for that election and the account expires after that. They behave similar to traditional paper ballots.

Each one of these approaches have pros and cons. On one hand, permanent accounts, in comparison with single-use accounts, are quite handier for voters as they do not need to get issued for each new election, and cheaper in cost and time for

administrations because they do not have to organize the account distribution in each new election.

On the other hand, it is important to note that Ethereum accounts do not have a "forgot my password option"[30] so voters are fully responsible of keeping their account credentials. This characteristic may end up on many voters losing their permanent accounts and, consequently, losing access to the system. The system must issue a new permanent account for those voters, but lost accounts cannot be de-authorised from the system because, in order to preserve anonymity, the administration should not store by any means the relationship between voters and account addresses. Because of that, an attacker may claim that he lost his account but in reality, he kept it in secret, so now the attacker has two accounts, which means double vote on elections, and this violates the unique vote principle.

For this last reason, permanent accounts are unviable as they would keep increasing in numbers as lost accounts cannot be de-authorised from the system. Therefore, **single-use accounts are the way to go**.

As mentioned earlier, it is vital for accounts distribution to be done in a way where it is impossible for the central authority to save the relationship between the voter identity and the account address. This is not a trivial issue for online voting distribution, as how can a voter authentication be detached from the account address? In order to obtain a vote, the voter must be previously authenticated into the system (with a token, for example). However, once the voter is authorised, the central authority already knows who the voter is, and when an authorised account is given to the voter, a malicious authority software might store the relationship between the voter's identity and the account address the voter is going to use to vote. This may violate the secret vote principle. It must be remembered that the system must be tamper-proof by construction, and it must guarantee the voting rights even in the case of a malicious authority.

The problem here relies on the communication between the voter and the central authority, as the central authority must communicate in some way the

[30] https://geth.ethereum.org/docs/interface/managing-your-accounts

account to the voter, but the system must prevent the authority to link the voter identity with the account address being assigned..

One way to overcome this issue would be to rely on analogic methods for voter authentication and vote obtention. This process would be like ballot boxes, but in reverse. The administrations would create authorised Ethereum accounts, print them along with their keys and credentials in a QR in a wrapped paper so it cannot be seen, and store them in ballot boxes. Then, voters, after they have been authorised to do so by officials, would pick them up randomly (instead of casting a vote into the ballot box, they take it out). After they have picked up the Ethereum credentials, they would use the blockchain interface they want (such as e.g. a mobile app) to log in with the account and cast their vote. In this way, voter authentication and vote obtention would be detached from the beginning from vote casting, as it is the voter who picks up the vote he wants from the ballot box, and not a central authority who gives it to them.

Even though this approach seems to be very primitive, it can be extremely useful and a very simple way to provide both transparency and perceived transparency by voters. There are recent approaches in academia to solve this issue by means of zero knowledge proofs (ZKP) and other very sophisticated techniques that apparently solve this issue, like ring signatures, but they are out of the scope of this work (for more information on this, see (Russo, González Vasco, & Pietro Romano, 2021) and its references). In fact, voters might not trust them due to their complexity, so analogic methods may work better in a real-life scenario.

4.1.4 Vote casting

The vote casting phase runs during the election period, usually a single day, and voters can deposit their votes into their preferred option.

It is vital that votes are not traced back to their casters, neither results are available before the election finishes.

Ethereum accounts from the previous phase will be used to cast the vote into the system smart contracts, inside the blockchain.

4.1.5 Results

Once the voting phase comes to an end no one has the possibility of casting any more votes. The results and how they were tallied must be publicly available and cannot be altered. Any voter can check if their vote was correctly casted and counted. All these functions must be available through smart contracts modules which must be accessible with Ethereum accounts.

4.2 System architecture

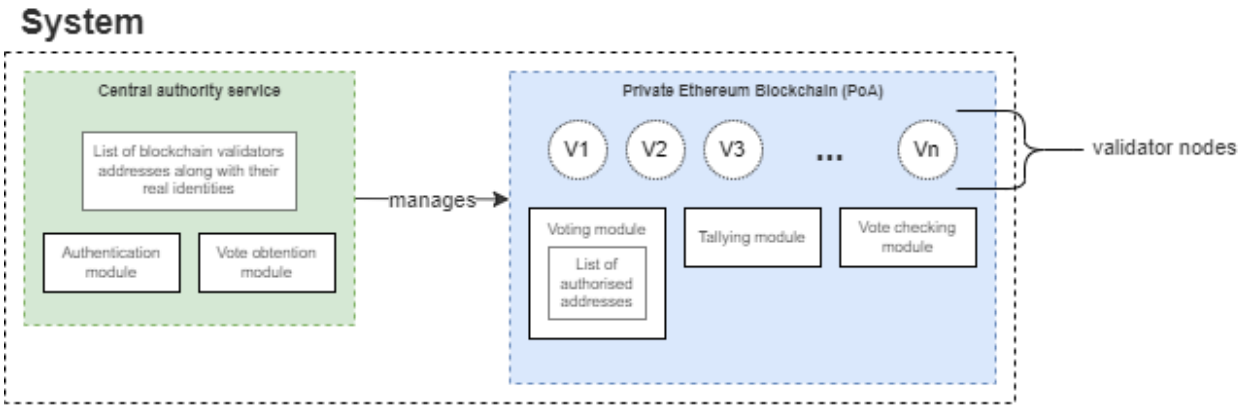


Figure 4-1 Proposed system architecture

The system architecture is composed of two separated modules, as can be seen in Figure 4-1:

- **Centralized module**: Contains functions related to voters authentication and vote obtention.
- **Decentralized module**: provides the means to vote, to tally the results and to check the casted vote.

4.2.1 Centralised module

4.2.1.1 Voter authentication and ballot obtention

It has been already explained in Section 4.1.3 how this is going to be addressed in the system: the government administration authenticates voters in person (with a

national identity card, for example), then voters themselves pick a random Ethereum address from a ballot box, and lastly they are registered by the government administration (in paper or digitally) as having already obtained a ballot. This process is done this way in order to *physically* detach voter's identities from their system identities. These actions are illustrated in Figure 4-2.

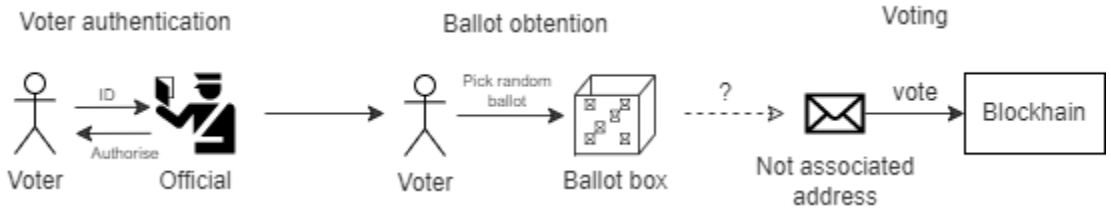


Figure 4-2 Voter authentication, ballot obtention and vote processes

However, this system exposed has a major flaw. As it is proposed right now, it records the number of total votes cast into the system, a number that must be smaller than the total citizens registered in the census (number of votes <= total of citizens). Nonetheless, there is no transparent and traceable way of knowing if those votes come from real people or not. In the system, each account is just a number as it has been detached from its real voter identity. Therefore, nothing stops a corrupted government to easily create multiple fake Ethereum accounts in secret (which no one will know they are fake), vote multiple times into the system in their favour, and then claim a fake percentage of election participation. Every democratic country publishes the election participation rate, but that figure can be faked to be higher than reality if sufficient effort and resources are put into.

Obviously, the malicious government is limited by the real election participation, as they cannot fake a participation that goes beyond the 100% of the total country census. For example, if there is a 75% of election participation (a number that is quite high, at least in Spain) then the corrupted government has a window of 25% to alter votes in their favour, which makes a big difference.

Therefore, it is vital not to leave the task of registering which voters already obtained their ballot to the good will of government administrations and register it also into the blockchain storage. Doing so will limit the damage that the government administrations can cause to the system, since if the government wants to tamper

results, it is not as simple as incrementing by one the number of citizens that obtained a ballot (which is opaque and does not give any information about which citizen identity has been faked specifically); now they have to fake entirely the identity of a real citizen by writing into the blockchain that this citizen obtained a ballot, and, if they do so they will get caught much easier than before.

Therefore, it is necessary to store the entire electoral census into the blockchain's smart contracts and assign to each identity a Boolean value that indicates if the citizen has obtained his/her ballot or not.

4.2.2 Decentralised module

4.2.2.1 Type of blockchain

As it can be seen in the diagram, the decentralized module is running on a **private blockchain**, aside from the Ethereum main net, with its own nodes, configuration, and blocks. This design decision is crucial to avoid the high costs of gas fees for performing actual transactions. In contrast, creating a private separated blockchain allows the organization (the country government) to drop the gas fees to zero.

However, making a private blockchain has important consequences. The most relevant one is that it requires miners to put the blockchain into work, and a reduced number of miners may make this blockchain more vulnerable to 51% attacks. Thus, government must make sure that the few miners the network has are all highly reliable. To do so, government administration must determine who can be a validator and who cannot. This can be done easily with a proof of authority consensus.

4.2.2.2 Consensus protocol

In order to ensure the network security, the system will not rely on proof of work consensus, as it needs many nodes to be secure and it is highly costly as exposed in Section 3.4.1. It should neither rely on proof of stake, as it does not make sense in this case as we surely do not want to limit validators to a rich minority who can afford to be one. The best consensus protocol that can be used in this case is **a proof of authority consensus**.

In this system each validator will be a known node to the government administration. **They will put their real identity as a proof that they are reliable nodes**. Validators will receive a monetary reward in exchange for their work to incentivise becoming a validator. However, if they commit any kind of fraud, this reward will not be paid and they would incur a crime imputation, a reputation loss by publishing the identities of those malicious validators and be fined with a relevant amount of money.

In this setting, several kinds of participants can be eligible as validator nodes: government administrations, legal entities, companies, and even citizens. Also, the number of validators for each category must be fixed, and they also must be assigned in a public raffle where validators are picked randomly to avoid any kind of organization between malicious peers that might compromise the network security. However, in order to fulfil this last purpose, it is also vital not to publish the validator's identities until the end of elections, when no damage can be done.

In this way, the network will not rely exclusively on government nodes, but also on citizens and legal entities nodes, increasing its security and drastically lessening the probability of tampering elections.

Also, although now validators are distributed among different types of entities, it is important to establish a distribution that does not give more than the 50% of nodes to government administrations, as it would incur the 51% attack Section 3.5.4 explained. The same applies for companies, as they could be controlled by governments and might be politically biased. However, citizens in a country tend to be more heterogeneous in ideology and cannot organise themselves as validator positions are raffled randomly and kept as secret until the end of elections. Therefore, most of the weight of validator nodes must lay on natural people. One possible and concrete distribution of validator nodes could be:

Validator nodes distribution

Government	Legal entities	Natural people (citizens)
20%	20%	60%

Lastly, Geth, the Ethereum tool that is going to be used in Section 4.3, uses an specific proof of authority consensus protocol called "Clique" (Szilágyi, 2017). Further details are going to be explained in the said section.

4.3 Prototype implementation

Once the properties of a blockchain-based voting system have been analysed and an architecture has been designed, we can propose a prototype implementation. We are going to follow the guidelines from the previous sections to implement a secure blockchain voting system that also guarantees voting rights.

4.3.1 Implementation tools

The following tools and programs are needed for this implementation:

- **Geth[31]:** this is one of the three original implementations of the Ethereum protocol written in Go. It is open-source[32] and is a command-line tool used for blockchain network management and node setup.
- **Remix[33]:** it is an open-source web-based development environment [34] (thought it also comes as a desktop application) used for smart contract development as it prepares the blockchain environment for their deployment and function calls. Remix also offers the possibility to connect to private blockchain networks, as the one that we are going to build with Geth.
- **Solidity:** it is the smart contract programming language and where the voting system logic will reside in. It is an object-oriented language where the classes are called "contracts".

---

[31] https://geth.ethereum.org/

[32] https://github.com/ethereum/go-ethereum

[33] https://remix-ide.readthedocs.io/en/latest/

[34] https://remix.ethereum.org/

- **Javascript:** there are Javascript libraries that make possible to communicate with a blockchain network, send transactions and communicate with smart contracts. Web3 is one of this APIs and it is going to be useful for testing purposes.
- **NodeJS[35]:** Javascript is a language usually executed inside a web browser, like Google Chrome or Firefox. However, there are tools, like NodeJS, which allow programmers to execute Javascript code locally (which is normally called to execute code on the server-side). With NodeJS we can build the Javascript environment and execute this code locally on the command line. Also, among NodeJS, there are other correlated programs that are needed to setup the whole environment, like the node package manager (npm[36]) which allows us to install other needed development tools.

We are going to use the following versions:

Versions

Geth	Remix	Solidity	Javascript	NodeJS
1.10.18-stable	0.23.3	^0.80	ECMAScript 2018	16.14.0

4.3.2 Blockchain network setup

A blockchain network is essentially composed of nodes. For this prototype implementation, the blockchain environment is composed of 5 nodes running Geth, with each one of them represented by a folder numerated from the number "01" to "05", as can be seen in Figure 4-1. At the same time, each node has two subdirectories: one where the nodes store their copy of the ledger, called *geth*, and

[35] https://nodejs.org/es/

[36] https://www.npmjs.com/

other where they store their account credentials, called *keystore*. This directory structure is illustrated in the Figure 4-2.

Figure 4-3 Nodes directories

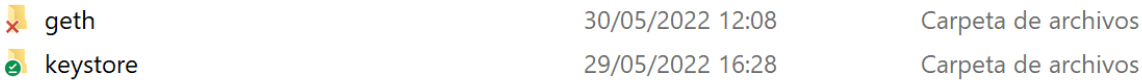


Figure 4-4 Node directory structure

The geth and keystore directories are created when executing the following Geth commands:

```
geth account new --datadir 01

geth account new --datadir 02

geth account new --datadir 03

geth account new --datadir 04

geth account new --datadir 05
```

These commands create a new account with its own credentials (a public and a private key) and store them in the keystore from the specified directory in the "--datadir" flag as can be seen in Figure 4-4. When this command is executed Geth also prompts the public key associated with the new account in the console as can be seen in Figure 4-3.

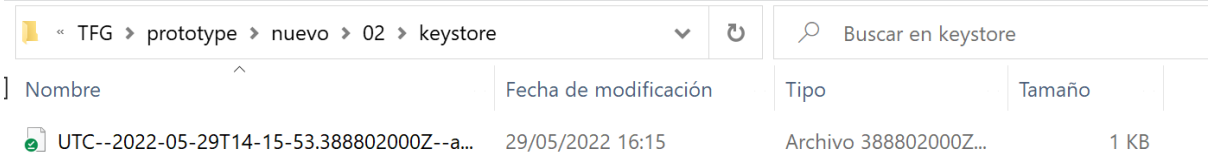


Figure 4-5 Keystore node directory

```
D:\Documents\Universidad\4º curso\TFG\prototype\nuevo>geth account new --datadir 01
INFO [05-29|20:35:00.961] Maximum peer count                       ETH=50 LES=0 total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x9d206854C19024702a1B1288cD8B8fed2DcBe4f4
Path of the secret key file: 01\keystore\UTC--2022-05-29T18-35-04.678395000Z--9d206854c19024702a1b1288cd8b8fed2dcbe4f4

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Figure 4-6 Geth new account output

These credentials are vital for nodes signing transactions and to perform any interaction with the blockchain, and as they do not have a "forget my password option" they must be stored securely. In this case, as it is a prototype, they are going to be just stored in the root folder of the project.

Once the accounts for the validator nodes have been created, the can start operating. However, all of them need a first block to start from and that makes them agree on some basic blockchain configuration parameters. This special block is called the genesis block. Its most important fields are:

- **chainID:** serves as an identifier to the whole blockchain so peers can find each other on the internet. The main net has the ID 1, but there are other side blockchains which use other identifiers. There is a list of Ethereum chain IDs[37] so new blockchains do not conflict with each other.
- **clique:** indicates that the consensus protocol used by nodes is the clique consensus. This consensus has already been mentioned in Section

[37] https://chainlist.org/

4.2.2.2. This protocol provides means to vote in new validators or vote out validators from the network based on their behaviour. This functionality is provided by overwriting the meaning of certain block fields like "miner", where the address of the proposed account to vote in or vote out is written, or "extradata" field, which is now used by current validators to express their vote towards the address proposed. However, this functionality must be restricted because validators will be assigned from the beginning in the genesis block by the government administrations and will not be modified. Also, letting validators vote in new members could compromise the security of the blockchain. Also, this protocol defends itself against malicious signers by just letting them sign one block out of every K. The paper where the protocol is explained (Szilágyi, 2017) indicates that this "K" number must be the result of the following expression: $floor(TOTAL\_VALIDATORS\ /\ 2)\ +\ 1$. This enforces majority consensus and ensures that damage is limited. For example, let us assume that there are 1000 validators in a network. In this case, each node would be able to sign one block out of every 501.

- **period:** specifies the block time of the chain in seconds. The main net has a block period of 15 seconds, but this number can be shortened or expanded.
- **difficulty:** In proof of work it specifies the difficulty or score of mining a block, but since proof of authority does not rely on computational power it is just used for containing an standalone score of the block to derive the quality of a chain.
- **extradata:** this parameter holds the list of addresses of the validator nodes. The validators addresses are next to each other and are between 65 zeros on each side. In this list we are going to write the addresses of the five previously created nodes. This field would look like this:

  0x0000000000000000000000000000000000000000000000000000000000000000000054926A2bF931C667F7046b7B32770460a1d940cfa0EC2254Be5dd6F7118E34E28BcF66D9B20CF2300fcbC3Bb013f68D3975e6c7610f29c86EA0FFF177F0Fc9fB8cd31539cf5851D6052dc02a3a929CcCC8941854475ef8E

Af6c7bf4Fa691E9f13092bdf70000000000000000000000000000000000000000000000000000000000000000000000000000

- **alloc:** allows to start the blockchain with pre-filled wallets for some specified addresses. In this case we fill the address of the administrator node (the 01) with a high initial balance.
- **gasLimit:** specifies the max amount of gas that can be consumed during function calls to smart contract. As in a real-life case this is going to be a closed system with just one smart contract that is going to be audited by security experts, the gas limit will be set to the maximum possible number that Geth allows.

Figure 4-7 Genesis block configuration

Now, to initialise nodes with the genesis block configuration, Geth provides the following command, which output can be seen in Figure 4-6:

```
geth init genesis.json --datadir 01

geth init genesis.json --datadir 02

geth init genesis.json --datadir 03

geth init genesis.json --datadir 04

geth init genesis.json --datadir 05
```

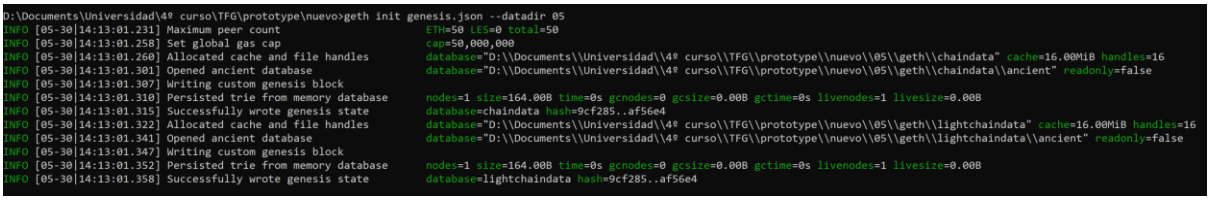


Figure 4-8 Genesis block initialisation output

Once this command is executed, it can be seen in the node directory that the previously mentioned folder "geth" has been created.

Now the blockchain nodes can start running. The initial node to setup is the *bootstrap node*, which is a special node that serves the purpose of being the node that the other nodes in the network will connect to the first time. To setup this node, the following command has been used on node 01:

```
geth --datadir 01 --networkid 1234 --nat extip:127.0.0.1 --
netrestrict 127.0.0.0/8 --mine --miner.threads=1 --miner.gasprice 0
--txpool.pricelimit 0 --http --http.corsdomain
https://remix.ethereum.org --http.api web3,eth,debug,personal,net -
-unlock 0x54926A2bF931C667F7046b7B32770460a1d940cf –password
password.txt --allow-insecure-unlock
```

The meaning of each field is:

- **--datadir:** specifies the directory of the bootstrap node, in this case the 01.
- --**networkid:** specifies the network to which connect to (must be the one specified in the genesis block, in this case 1234).
- **--nat:** indicates the IP address of the bootstrap node. It needs to know about its own IP address in order to be able to relay it to others. As it is a localhost blockchain, we specify 127.0.0.1.
- **--netrestrict:** restrict the connections to the specified network. As it is a local blockchain, we specify 127.0.0.0/8.
- **--mine**: indicates that the node is a "miner" or validator and must sign blocks.
- **--miner.threads:** indicates how many threads are used for mining. In this case we just need one, but it can be increased in real-life cases where more efficiency is needed.
- **--miner.gasprice:** indicates the price of gas. Transaction in this voting system must be gas free, so we specify 0.
- --**txpool.pricelimit:** indicates the minimum gas price limit to enforce for acceptance into the transaction pool of the node. Again, as this network needs zero gas, price limit is set to 0.
- **--http:** enables the HTTP-RPC server. This allows Remix to connect to the blockchain network for developing purposes. However, in a real-life situation it should be disabled.
- **--http.corsdomain:** comma separated list of domains from which to accept cross origin requests. For the same goal as the previous point, we specify the Remix domain.
- **--http.api:** list of APIs offered over the HTTP-RPV interface. We allow for developing purposes, however this should be reviewed for real-life purposes.
- **--unlock:** specifies the account address that is going to be used for signing blocks, in this case:

"0x54926A2bF931C667F7046b7B32770460a1d940cf".

- **--password:** the file which contains the password to unlock the specified account.
- **--allow-insecure-unlock:** Geth by default does not allow this type of unlocks as they are considered insecure, so we specify this option to enable it.

Once the command is executed, the bootstrap node will start running and looking for peers as can be seen in Figure 4-7.

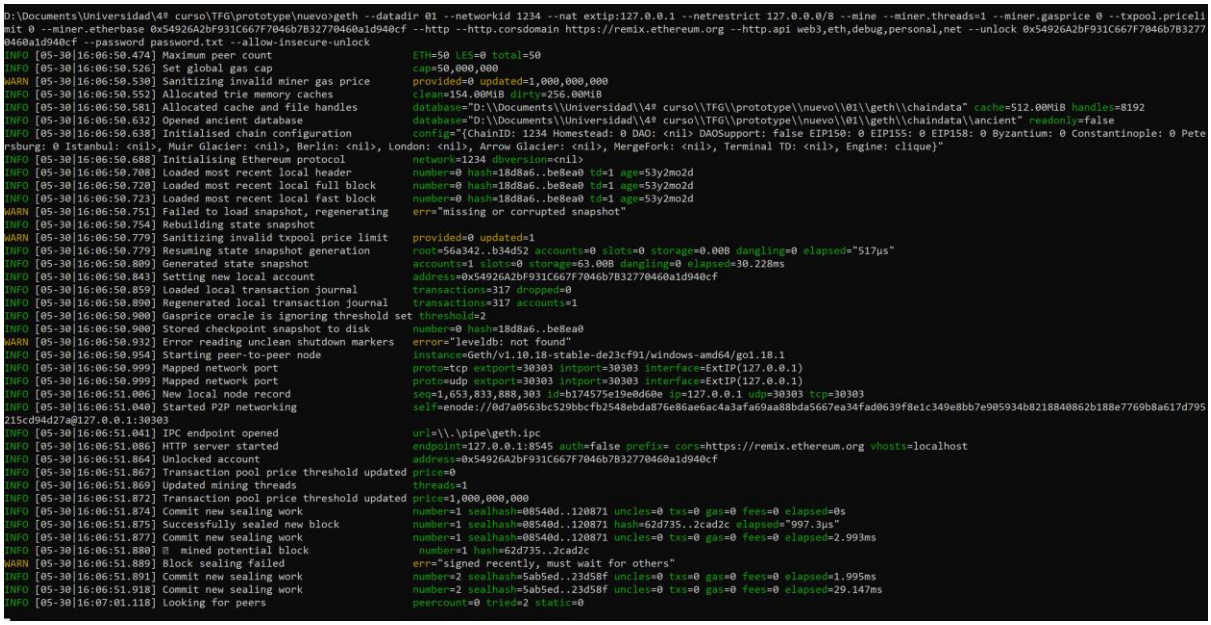


Figure 4-9 Bootstrap node running and looking for peers

Now that the bootstrap node is running, the other peers can connect to it. However, they need a "node record" from the bootstrap node. To extract this record Geth provides a Javascript console. To open it, the following command is used (this is for Windows, for other operating systems the URL to attach to may be different):

```
geth attach \\.\pipe\geth.ipc
```

The previous command opens a special console where we can introduce Javascript code. To obtain the node record the following command must be used:

```
admin.nodeInfo.enr
```

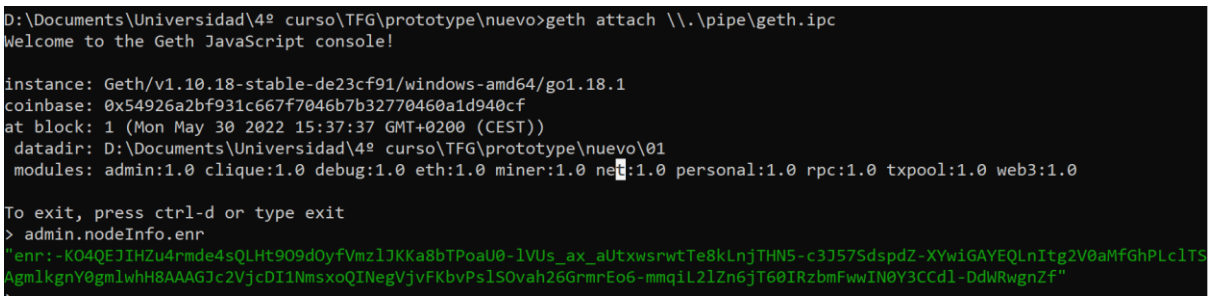


Figure 4-10 Bootstrap node record obtention

When the node record has already been obtained (Figure 4-8) it is possible to connect the other peer. To do so, the following command has been used (this one sets up node 02, but the same command is used for the other nodes but changing the –datadir, –unlock and –port flags):

```
geth --datadir 02 --ipcdisable --networkid 1234 --bootnodes "enr:-
KO4QEJIHZu4rmde4sQLHt9O9dOyfVmzlJKKa8bTPoaU0-
lVUs_ax_aUtxwsrwtTe8kLnjTHN5-c3J57SdspdZ-
XYwiGAYEQLnItg2V0aMfGhPLclTSAgmlkgnY0gmlwhH8AAAGJc2VjcDI1NmsxoQINeg
VjvFKbvPslSOvah26GrmrEo6-mmqiL2lZn6jT60IRzbmFwwIN0Y3CCdl-DdWRwgnZf"
--mine --miner.threads=1 --miner.gasprice 0 --txpool.pricelimit 0 -
-port 30307 --netrestrict 127.0.0.0/8 --unlock
0xa0EC2254Be5dd6F7118E34E28BcF66D9B20CF230 --password password.txt
--allow-insecure-unlock
```

The **–ipcdisable** flag indicates to deactivate the IPC-RPC server, as the node 01 is already using it.

Now all validator nodes are connected and the blockchain is up and running. Now it is time to deploy smart contract on it.

4.3.3 Smart contracts

The smart contracts are the ones who make possible the logic of the voting system. This implementation consists of two solidity files which are going to be explained in the following two sections.

4.3.3.1 Election.sol

It contains the Election contract who is responsible for the creation and management of the election. It is responsible for storing the political parties and regions involved and is managed by an owner, who typically is the blockchain administrator and is the one who can make modifications over it.

It contains the following storage variables:

- **owner:** address of the account which manages the election. It has permissions over special parts of the blockchain, as the function to add parties, add regions, add managers, register citizens, and authorise voters.

- **encryptKey**: public key which is going to be used to encrypt votes.

- **decryptKey:** private key which is going to be used to decrypt votes after an election ends.

- **startDataTimestamp/endDateTimestamp:** are two UNIX timestamps that indicate the range of dates where an election is on course. Before the start date, modifications to the system can be done, as adding regions, parties, managers, citizens or voters. After the start date, no

modifications can be done to these structures. They are specified in a timestamp format because in this way voters will have the guarantee that the system will be available for voting at the exact moment it is specified. If it were implemented by, for example, a lock, it could be imprecise, and the election could be delayed as someone has to lock it.

```
constructor(uint256 startTimestamp, uint256 endTimestamp) {
    owner = msg.sender;
    regionsID = new uint[](0);
    partiesID = new uint256[](0);
    startDateTimestamp = startTimestamp;
    endDateTimestamp = endTimestamp;
}
```

- **parties/partiesID:** two variables that are responsible for storing the political parties involved in the election. Parties are stored in a mapping structure that relates a party ID with a party object. Each party object has an ID, name and a list of candidates. The partiesID array stores the IDs of all parties registered in the system. This is done because it is not possible in Solidity to retrieve back the keys contained in a mapping structure, and as a consequence they can be lost. Therefore, they are stored in an array that any account can consult.

- **regions/regionsID:** it serves the same purpose as the pair parties/partiesID but with regions. The only difference is that each region is a contract with its own functions.

- **Events:** there are three events defined (PartyIdEvent, RegionIdEvent and RegionAddressEvent) which allows functions to return values to the Javascript environment.

Also, there are functions that allows users to read the state of the storage variables. These functions are: *getStartTimestamp, getEndTimestamp, getPartyById, gerRegionById, getPartiesIDList, getRegionsIDList, getAllVotesFromRegion* and *getVote*. Each function is quite self-explanatory and can be executed at any time. However, functions *getAllVotesFromRegion* and *getVoteFromRegion* can be only executed to obtain all encrypted votes after the election ended to avoid consulting results before-hand. However, as we already know, that state of a blockchain can be seen even when a variable is defined as private, so this measure just avoids improper function calls.

However, there are other functions that need some attention:

- **setDecryptKey**: sets the decrypt key so votes can be decrypted. It can be only executed by the owner of the election contract after the election starts.

- **addParty:** adds a new party to the election appending it to the "parties" and "partiesID" variables given an ID, a name and a list of candidates. The reason why the ID is given and not internally generated is explained later. It just can be executed by the owner of the election contract before the election starts.

- **addRegion:** adds a new region to the election appending it to the "regions" and "regionsID" variables. It just can be executed by the owner of the election contract before the election starts.

- **addManagerListToRegion:** adds a list of new managers to the given region. A manager is a special address which role is to mark which citizens have obtained a ballot in the region census it is authorised in before the election starts.

- **registerCitizenList:** registers a list of new citizens to a region census. It just can be executed by the owner of the election contract before the election starts.

- **citizenObtainedBallot:** marks whether a citizen obtained a ballot in a specified region. It seems that it can be executed by any address but the unique call that this function makes is done to the function citizenObtainedVote from the ElectionRegion contract which just allows managers of that region. It can be only executed when an election is on course

- **authoriseVoterList:** authorises an address to vote in a given region. It just can be executed by the owner of the election contract before the election starts.

- **castVote:** function that allows voters to vote to a party in a given region. It is specified as a string because it is encrypted. It can be only executed when the election is on course.

4.3.3.2 ElectionRegion.sol

Voting in an election is done by regions, as most countries use the D'Hondt system for proportional representation among regions of different sizes. Normally countries create various subdivisions for each region. However, we are going to keep it simple for the sake of the implementation.

Each region in an election contains the following variables:

- **owner:** the owner address of the contract, which is the Election contract as is the one who creates this "subcontract".

- **name:** the name of the region, which is a string.

- **id:** the ID of the region, which is an unsigned number.

- **isManager/managerList:** managers accounts represent the officials that are behind a ballot box at the moment of giving ballots to citizens in the real world. These two variables that store the managers from the region. The first one is a mapping structure that indicates whether an address is

manager or not. The second variable persists the managers addresses as they cannot be retrieved from the mapping structure as it was explained previously.

```
mapping(address => bool) private isManager;
address[] private managerList;
```

- **census/citizensID:** two variables that store the citizens registered in the census of the region. It follows the same structure as the previous point: a mapping structure that relates citizens IDs with a citizen object and an array for the IDs. Each citizen has a variable that indicates if it is registered (it could be consulted in the array of IDs, but in this way the consulting of registered citizens is accelerated) and a variable that indicates if the citizen obtained a vote.

```
struct Citizen {
    bool isRegistered;
    bool voteObtained;
}
mapping(string => Citizen) private census;
string[] private citizensID;
```

- **voters/votersList:** two variables that store the address of the authorised voters in the regions. It follows the same structure as the previous point: a mapping structure that relates a voters address with a voter object and an array for the IDs. Each voter has a variable that indicates if he/she is authorised to vote in that region (it could be consulted in the array of IDs, but in this way the consulting of authorised voters is accelerated), a variable that indicates whether a voter cast his/her vote and a variable that stores the encrypted vote.

```
struct Voter {
    bool isAuthorised;
    bool hasVoted;
    string vote;
}
mapping(address => Voter) private voters;
address[] private votersList;
```

- **votes:** a mapping structure that relates encrypted votes with a variable that indicates if it has been already cast. This is done to ensure that all encrypted votes are different, so no previous tallying can be done. This is explained in detail in the next section.

This contract is managed by the Election contract, so most of its functions are restricted to the address of that contract. Also, it has function to read the state of the storage variables, as previously. These functions are *getManagers, accountIsManager, citizenHasObtainedVote, citizenIsRegistered, getCensus, isAuthorised, hasVoted, getVotesFrom, getAllVotes, getAuthorisedVoters, getName* and *getID*.

Apart from the read-only function, the following ones perform other more complex tasks:

- **addManagerList**: adds a list of managers to the pair of variables isManager/managerList. It can be just executed by the owner (the Election contract).

- **registerCitizenList:** registers a list of citizens into the pair of variables census/citizensID.

- **citizenObtainedVote:** marks a citizen ID to indicate that obtained a ballot (an authorised account address). It can be just executed by a manager of the region and it can mark citizens that are registered into the system and who have not obtained previously a ballot.

- **authoriseVoterList:** authorises the given voter list into the region to perform votes. The encrypted vote that this function receives must not be repeated. It can be only executed by the owner (Election contract).

- **castVote:** allows authorised voters, who have not already voted, in that region to cast a vote. It can be only executed by the owner (Election contract).

4.3.4 Vote encryption

In order to avoid previous tallying of votes, they must be casted already encrypted. However, what key to use for encryption?

- **Symmetric encryption algorithms**: they provide a single key to encrypt and decrypt data. With these algorithms the unique option viable would be to make a system where the government administration gives each voter a unique secret key along their addresses. Each secret key would be directly related with a voter address in a database. This key would be used to encrypt their vote and, after the election, the government would use the secret key from that voter address key to decrypt his vote. However, it would introduce more complexity to the system as now voters need to receive, firstly, a unique voter address, and secondly, a unique secret key. Also, as the keys must be stored privately in a closed database so no tallying is done until the end of the election, it is a single-point failure, as losing the keys (which, if there is one per voter, it could be millions of keys) would imply to not be able to tally the votes and therefore to have to repeat elections.

- **Asymmetric encryption algorithms:** also called public key encryption, they are based on a pair of related keys. They encrypt data with a "public key" and it can be just decrypted with a "private key". This system seems more viable as it introduces more options.

- Each voter generates a public and private key, which they will keep in secret and are responsible for not losing. They encrypt their vote with the first one and cast it. At the end of the election, they must decrypt it with the other key. Just correctly decrypted votes will be counted. However, this system is quite complex, as it introduces a two-step voting process, which could end up on many voters forgetting to decrypt their vote at the end of the election, or even loosing their private key, which would incur in violating the universal vote principle. Therefore, this option is not viable.

- Government provides, along the account address, a public key to voters and store the private key in their database. Each private key would be related with a voter address, so at the end of the election all votes can be decrypted. However, this option has the same downsides as the symmetric encryption algorithms, so it is not viable.

- The government generates a public key for encrypting votes which publishes in the smart contract. Then, at the end of the election, the government publishes the private key, so everyone can decrypt all votes (as all of them are encrypted with the same public key) and tally them.

Out of three options, the third one is the most viable. However, it introduces a new problem: if everyone encrypts the same data (political parties IDs) with the same public key, then every encrypted vote to the same political party will look the same. Therefore, though no one can know which political party is who has a certain number of votes, they can still be influenced by this data. Thus, this solution, as it is right now, is not sufficient. Every encrypted text must be different to the other.

The solution to this problem would be to not directly cast votes with the ID of a political party, but to cast a vote with a number which modulo with the maximum party ID number plus one gives the party ID of the option voted. An example would be:

- We have the following five political parties:

Party	**A**	**B**	**C**	**D**	**E**
ID	564	223	943	243	12

- To vote to political party **D**, we would have to pick a number which modulo with the biggest ID plus one (the ID of the political party C, which is 943, plus one is 944) gives the ID of the political party **D**. Therefore, we pick a number that satisfies this restriction, for example, 1187.

- Now we encrypt that number with the government public key, which would give a value X, and vote with it.

- When elections end, if we want to tally results we have just to decrypt the vote with the government private key which will be published in the blockchain, make the modulo operation with the biggest ID number plus one (944) and we obtain the option that the voter chose, in this case, 243.

However, this would rely on voters choosing numbers that do not repeat themselves among all the other votes. Therefore, to ensure that all votes always have different encrypted values, we force voters to find a number that it is not already cast on the blockchain. Following the previous example, if another voter wants to vote for party **D**, he/she will have to find another number different than 1187 which modulo with 944 gives 243, for example 2131.

However, it is important to note that this system allows users to cast invalid votes, as there is no way of checking if they are voting correctly. So, all votes that do not correspond to any option presented will be considered not valid, and consequently not tallied.

With this proposed system, all votes cast are different from the rest and the government administration (and anyone else) can tally them easily.

4.4 Vulnerabilities and issues

The proposed system has the following issues and vulnerabilities:

- The major issue with this system is the authorization process, which has to be done physically in order to detach voter identities from accounts addresses. This is a job to be done by the government administration. Therefore, a vulnerability of this is that **it can lead to unauthorized voters to enter the system and corrupt an election**. It must ensure that each individual has just one authorized account address, so everyone has the same number of votes. This is a job to be done by the officials who are guarding the ballot boxes. **This can vulnerate the equal vote principle**.

- All the responsibility of keeping the account secure and for not losing it lays on voters. And, since Ethereum accounts do not have a "Forget my password option", this can lead to a huge problem if a user forgets his own password as **this threatens the universal vote principle**.

- The process of assigning accounts to the citizens must be done privately, without anyone else who is not the owner discovers the account number or password in the process. This again falls into administrations and officials hands. **This threatens the secret vote principle**.

- The process of generating the account addresses and authorising all of them into the blockchain must be done cautiously, because an

unauthorised address that falls into the ballot box will cause a voter to not be able to vote. **This threatens the universal vote principle**.

- An alliance between validator nodes to compromise network security could be catastrophic to the election. **This threatens the universal vote principle** as a 51% attack could delete votes or to directly stop all the blockchain. To ensure that this does not happen the list of validators must not be published until the end of election to avoid any kind of collusion.

- Validators must be working until the end of the election. A contract between the government and the validator must be done to ensure that the validator takes full responsibility for being active and to ensure that it does not leave with an election on course. **This threatens the universal vote principle** as, if validators leave, the network is more prone to 51% attacks.

- Any smart contract vulnerability could be exploded by attackers and cause unwanted behaviours. **This threatens the free of choice** (as votes could be manipulated if there was a vulnerability that allowed to do so) **and the universal vote principle.** Therefore, it is important to analyse and study the source code carefully to avoid any type of vulnerability, like re-entrancy attacks (for our implementation it is impossible as it does not call any external contract). Special attention should be given to authorization and voting functions.

- The private key to decrypt all votes resides on the government administrations. This is a big issue because introduces a single-point of failure component. However, as it is just one key, and not a whole database of different keys (as was proposed in one of the options of Section 4.3.4), the complexity and risk of losing it decrease. However, as

it is just one element it would be much easier to steal it, so it must be guarded securely. Also, strong, updated and secure symmetric encryption algorithms, like AES, must be used to avoid force-brute attacks.

- One downside of the system is that it is known if a certain citizen has voted or not (but not the vote). This design decision was made to not let government administrations fake votes, as to do so they would have to impersonate a real citizen. However, with traditional voting systems it is also possible to know if someone voted or not, as you see them in person in the ballot box location.

- If few people vote in one district it could be known who voted what. This problem also happens with traditional voting system, like ballot boxes in a small town. Therefore, to avoid these districts should cover as many people as possible.

4.5 Scalability

Does this implementation scales to a whole country with millions of voters? Any direct test could be done with millions of voters, but analysis can be done with data extracted from the Ethereum main net. The Ethereum main net can process around 30 transaction per second, what would mean 2.592.000 transactions per 24 hours. This is a quite low transaction rate as for a country like Spain would take roughly one week to process all votes. However, it must be taken into account that the Ethereum blockchain is proof-of-work-based, so this number is quite lower than the real number of transactions that could be done in a private blockchain.

A better comparison would be with another private blockchain, as it serves better as a reference for time between transactions. Hyperledger Fabric is a permissioned blockchain designed for business purposes that has a transaction rate of 3.500, but it can be scaled to 20.000 transactions per second, depending on the system configuration (Gorenflo, Lee, Golab, & Keshav, 2019). So, taking those numbers

into account, this system could process in eight hours (what an election in Spain lasts) between 100.800.000 and 576.000.000 transactions, which is sufficient for all countries in the world.

However, a major bottleneck resides on creating the Ethereum accounts, as it is a slower process. In a normal computer, it takes around two seconds. Scaled to 20.000.000 of accounts, it would take 11 hours to generate all of them.

4.6 Source code availability

The smart contracts proposed in this section and the testing program made with NodeJS are available in the following GitHub repository:

https://github.com/ZaderCode/blockchain_voting_system/tree/main

5. Conclusion

This project began with a question: Is blockchain technology the future of voting?

To be able to answer it, we had firstly to review what is a voting system and which properties it should fulfil that we refer to as the voting rights. Current implemented approaches have been studied to understand better how they accomplish voting rights, and what were their flaws and we saw that the blockchain properties could cover them up. However, this was not enough as a more profound analysis had to be done to see how exactly it would be implemented and if it fulfilled voting rights.

To do so, current blockchain voting system approaches have been analysed to see what they accomplished and what were their flaws. Active Citizen was one of the most direct blockchain implementations a country implemented for an election. However, it fell into centralization methods which threatened voting rights and introduced single-point failures. Gyenggi-do case was a better implementation of a decentralised voting system; however, its use was mostly commercial, so the use differs from the one studied here. Sierra Leone made a partial tally of election results with the Agora blockchain system made by the company of the same name. However, Agora relied the authentication process on external administrations, which, depending on the implementation used for authenticating users, it could threat their voting rights. Therefore, it was not enough. Lastly, we saw an academic-level voting proposal which defined a private blockchain system which accomplished decentralization and gave each voter a unique address to vote so they could then check it. However, nodes were controlled by government administrators, the authentication process was relied on external centralized applications, and the votes were not encrypted, so the tally could be done in any moment, influencing voters choice.

Lastly, considering all flaws of the blockchain cases studied, we designed our own implementation of a voting system: a proof of authority

based private blockchain which accomplished full decentralization giving citizens, legal entities, and government administrations a portion of the total validators in the net to avoid 51% attacks, which also made the blockchain completely public, so anybody could enter and validate externally the blocks in it. Also, the system proposes a viable encryption mechanism which ensures that all votes are encrypted securely and it is not possible to do tallies until the government administrator publishes the private key. However, the major flaw of the system keeps being the step between authentication process and ballot obtention, where we ended up relying on physical methods to ensure completely that voters identities and account addresses are detached from the beginning. However, to cover this drawback, we proposed a concrete methodology based on reversed ballot boxes where citizens, instead of casting a ballot into them, they take out one in the form of an authorised account address and then they use this account to vote from their devices. Lastly, we made a security analysis searching for vulnerabilities and issues in the whole system, where we saw their issues and consequences, with the authentication process and the private key storage been one of the most high-risk issues.

In conclusion, the system proposed, taking the necessary security measures, complies with voting rights and we consider that it could be a viable voting system to be applied into a real election.

6. Future work

We propose as future work guidelines the following topics:

- Designing a digitalised authentication protocol that allows eligible users to gain access into the system but without giving the centralized authority the possibility of knowing their identities inside the system. The use of zero-knowlege techniques seems a very promising approach to this end.

- Designing another methodology to avoid the central authority from casting fake votes into the system that does not publishes if a citizen obtained a ballot or not.

- High-intensive load testings could be done over this system with millions of users.

7. References

Agora. (n.d.). *Agora Whitepaper.* Retrieved May 26, 2022, from https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5f37eed8cedac41642edb534/1597501378925/Agora_Whitepaper.pdf

C. Eggers, A., Garro, H., & Grimmer, J. (2021, August 30). *No evidence for systematic voter fraud: A guide tostatistical claims about the 2020 election.* University of Chicago, Department of Political Science. Retrieved from https://www.pnas.org/doi/epdf/10.1073/pnas.2103619118

China Constitution. (2019, November 20). China. Retrieved May 8, 2022, from http://www.npc.gov.cn/englishnpc/constitution2019/201911/1f65146fb6104dd3a2793875d19b5b29.shtml

Gaudry, P., & Golovnev, A. (2019). *Breaking the Encryption Scheme of the Moscow Internet Voting.* Retrieved 05 20, 2022, from https://arxiv.org/pdf/1908.05127.pdf

Gorenflo, C., Lee, S., Golab, L., & Keshav, S. (2019). *FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second.* International Conference on Blockchain and Cryptocurrency (ICBC). Retrieved May 28, 2022, from https://ieeexplore.ieee.org/document/8751452

Gritsenko, D., & Indukaev, A. (2021). *Digitalising City Governance in Russia: The Case of the 'Active Citizen' Platform.* Retrieved 05 20, 2022, from https://www.tandfonline.com/doi/full/10.1080/09668136.2021.1946013

Gritsenko, D., & Indukaev, A. (2021). *Digitalising City Governance in Russia: The Case of the 'Active Citizen' Platform.* Retrieved from https://www.tandfonline.com/doi/full/10.1080/09668136.2021.1946013

Kshetri, N., & Voas, J. (2018). *Blockchain-Enabled E-Voting.* IEEE Software.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* Retrieved from https://bitcoin.org/bitcoin.pdf

R. Mebane, W., & Kalinin, K. (2010, April 20). *Electoral Fraud in Russia: Vote Counts Analysis using Second-digit Mean Tests*. Michigan, United States. Retrieved from http://websites.umich.edu/~wmebane/mw10B.pdf

Russia Constitution. (1993, December 12). Article 1. Russia. Retrieved from http://www.constitution.ru/en/10003000-01.htm

Russo, A., González Vasco, M., & Pietro Romano, S. (2021). *Chirotonia: A Scalable and Secure e-Voting Framework based on Blockchains and Linkable Ring Signatures.* Melbourne, Australia: IEEE International Conference on Blockchain. Retrieved from https://ieeexplore.ieee.org/abstract/document/9680518

Santamarta, R. (2022, January 18). *Finding vulnerabilities in Swiss Post's future e-voting system.* Retrieved May 15, 2022, from https://www.reversemode.com/2022/01/finding-vulnerabilities-in-swiss-posts.html

Spain Constitution. (1985, Junio 19). *BOE*. Retrieved from Ley Orgánica 5/1985, de 19 de junio, del Régimen Electoral General: https://www.boe.es/buscar/act.php?id=BOE-A-1985-11672

Szilágyi, P. (2017, March 06). *EIP-225: Clique proof-of-authority consensus protocol.* Retrieved from https://eips.ethereum.org/EIPS/eip-225#attack-vector-malicious-signer

Þ. Hjálmarsson, F., K. Hreiðarsson, G., Hamdaqa, M., & Hjálmtýsson, G. (2018). *Blockchain-Based E-Voting System.* 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). doi:10.1109/CLOUD.2018.00151

University of Cambridge. (2022, 05 20). *Cambridge Bitcoin Electricity Consumption Index*. Retrieved 05 20, 2022, from https://ccaf.io/cbeci/index/comparisons