A distributed architecture proposal for e-voting

Uma proposta de arquitetura distribuída para votação eletrônica

João Marcos Soares [*1] **and Rafael Oliveira Vasconcelos** [†1]

[1] *Universidade Federal de Sergipe, Departamento de Computação, São Cristóvão, SE, Brasil.*

Abstract

Manual voting processes have two main problems, the reliability of the result and the delay in counting the votes. To defraud the election, that is, change the amount of votes of the candidates, an attacker can replace ballots with votes for other candidates with ballots with votes for the candidate to be benefited. In this scenario, the security of the electoral process strongly depends on the control of physical access to the ballot boxes. On the other hand, electronic voting systems (e-voting) are known for their agility in counting the votes, as well as having the potential to increase the security of the electoral process. However, both solutions present challenges in relation to the transparency, security and secrecy of the vote. This work presents conceptual and technological requirements for a secure electronic election, proposes a distributed solution for electronic voting, and presents its limitations and some possibilities for future work. Finally, more studies are required to implement the solution, mainly in relation to the secrecy of the vote and incoercibility.

Keywords: Election. Electronic voting system. E-voting. Consensus. Distributed systems.

Resumo

Processos de votação manual apresentam dois principais problemas: a confiabilidade do resultado e a demora na contagem dos votos. Para fraudar a eleição, isto é, alterar a quantidade de votos dos candidatos, um atacante pode substituir cédulas com votos para outros candidatos por cédulas com votos para o candidato a ser beneficiado. Nesse cenário, a segurança do processo eleitoral depende fortemente do controle de acesso físico empregado às urnas. Por outro lado, sistemas de votação eletrônico (*e-voting*) são conhecidos pela agilidade na contagem dos votos, bem como têm potencial para aumentar a segurança do processo eleitoral. Entretanto, ambas as soluções apresentam desafios em relação à transparência, segurança e sigilo do voto. Este trabalho apresenta os requisitos conceituais e tecnológicos para uma eleição eletrônica segura, propõe uma solução distribuída para votação eletrônica, apresenta suas limitações e algumas possibilidade de trabalho futuro. Por fim, são requeridos mais estudos para a implementação da solução, principalmente em relação ao sigilo do voto e incoercibilidade.

Palavras-chave: Eleição. Sistema de votação eletrônico. E-voting. Consenso. Sistemas distribuídos.

Textolivre
Linguagem e Tecnologia

DOI: 10.1590/1983-3652.2023.42204

Session:
Articles

Corresponding author:
Rafael Oliveira Vasconcelos

Section Editor:
Danierlevin Pereira
Layout editor:
Leonardo Araújo

Received on:
December 21, 2022
Accepted on:
February 3, 2023
Published on:
April 6, 2023

1 Introduction

Proposals for electronic election systems (e-voting) seek to lower the costs of an election while maintaining integrity, security, and privacy requirements (HJÁLMARSSON et al., 2018). While the efficiency and cost benefits in computerized processes are clear relative to analog processes, security care can become more complex (STALLINGS, 2017).

The literature points to security as the primary challenge in e-voting. Works bring up the theoretical risks of e-voting compared to the risks of the traditional system (LAUER, 2004; GRITZALIS, 2002). Other works point out flaws in e-voting system implementations (SPRINGALL et al., 2014; ARANHA; GRAAF, 2018).

In this sense, the advent of cloud computing and decentralized architectures arise as an attempt to deal with the imposed security challenges, mainly related to trust in the system. With a decentralized architecture, the parties involved need to enter into an agreement on the values that will be committed.

*Email: joao.soares@dcomp.ufs.br
†Email: rafael@dcomp.ufs.br

Sharing responsibilities can improve the trust in the system. To this end, it will be presented how distributed computing systems resolve conflicts and how this can be leveraged for a decentralized architecture.

The advantage of the aforementioned tool is to spread the responsibility of the system among several entities, adding redundancy. This paper seeks to present the requirements and challenges that a distributed architecture of choice must possess. Initially it was planned that the proposed format would use blockchain as a platform. However, it was realized that certain exigencies of electoral systems make the pure application of blockchain not ideal. Blockchain alone does not provide the authentication mechanisms necessary for e-voting. If the network requires participants to be identified, a central entity is needed. In addition, part of the secrecy in blockchain (more specifically in the Bitcoin network) comes from the fact that users are not authenticated and therefore can assume numerous identities.

Blockchain also does not solve the problem of secret storage of votes (which will be presented in Section 2.1.9 and discussed at the beginning of Section 3). That is, there is no way to guarantee that the vote stored in the chain will only be available at the end of the election. Thus, the benefit of blockchain is restricted to distributed storage and immutability. For distributed storage, a traditional distributed database can solve the problem. A similar result to immutability can be achieved if the system enters into consensus on vote ciphers before disclosure (see Section 3.3.2).

1.1 Objectives

The goal of this paper is to present a proposal for a decentralized and distributed architecture for "small electronic elections", where the number of responsibilities of a central entity should be as few as possible. For the purposes of this research, "small electronic elections" will be understood as the internal elections of public and private institutions, student organizations, and unions.

In order to achieve the general objective, the following specific objectives have been outlined:

- Present the requirements of a secure electronic election;
- Propose a distributed election model based on the raised requirements;
- Analyze the main challenges imposed by this model;
- Analyze its advantages, disadvantages and viability.

1.2 Methodology

For the development of this work, exploratory research on e-voting was carried out in the literature. After that, concepts about distributed systems and cryptography were studied. Finally, an architecture based on the studied themes was proposed.

1.3 Related work

For the present study, articles on electronic voting were used as a basis. Gritzalis (2002) points out a set of principles that the development of election systems should comply with, based on constitutional principles of democratic countries. In addition, the author discusses whether e-voting should be used as a complementary means to the traditional process or conducted independently.

Lauer (2004) lists the possible risks of e-voting, both Internet voting and the use of technology, to support the traditional election. Risks at the physical and network levels are analyzed and recommendations are made on how to mitigate these risks.

The work of Springall et al. (2014) analyzes the security of Estonia's online election system, using code analysis, in-person observations, and intrusion tests on replicas of the system as a basis. The results show that there are security flaws in the architecture and that the design of electronic voting systems is difficult, as it must ensure confidence in the result while maintaining the secrecy of the vote.

A similar work is done by Aranha and Graaf (2018), but in the context of Brazilian elections. The authors present the electoral process in Brazil and discuss points of failure involving the electronic ballot box. It is discussed that one of the main points of failure is in the source code validation process, as it is not guaranteed that the source code that is contained in the ballot box is the same

as that audited by the stakeholders. Moreover, the source code auditing process itself is limited.

The use of cloud computing for electronic elections is explored in the work of Zissis and Lekkas (2011). Also in this paper, the use of electronic processes in governments is reviewed, and a set of threats are listed, such as malicious code on the client, message interception, and their respective security measures.

2 Background

In this section, the issues that serve as a basis for the paper will be presented. Section 2.1 presents the principles necessary for the development of a secure electronic election system. Section 2.2 presents the concept of distributed systems and consensus algorithms. Section 2.3 introduces concepts of time-based encryption.

2.1 Requirements for a secure electronic election

The first step in building the architecture of a secure election system is to define the requirements for it to be considered secure and effective. Several papers in the literature (ZISSIS; LEKKAS, 2011; SAMPIGETHAYA; POOVENDRAN, 2006; HARDWICK et al., 2018) use the work of Gritzalis (2002) as a reference for requirements' mapping.

Gritzalis (2002) defines a series of design fundamentals that should be followed when developing e-voting systems. They derive from constitutional principles in democratic countries. These principles sometimes complement and sometimes contradict each other. It is therefore impossible to design a system (whether traditional or electronic) that completely satisfies all of them. The principles are listed below.

2.1.1 Principle of isomorphism to the traditional process

The electronic electoral process must guarantee the characteristics of the traditional universal suffrage electoral process. For this, the following premises are defined:

1. Every voter has the right to participate in the electoral process;
2. The definition of the individual as a voter must be found and controlled by law;
3. The technology to cast the vote must be accessible to all voters;
4. E-voting should be considered an alternative means of voting;
5. The democratic principle (i.e, every voter has the right to participate in the electoral process) leads to the need for adequate public infrastructure (free internet points, etc).

These definitions were made for national elections. For more restricted electoral processes, such as those at a university, the law mentioned in item 2 of the above premises can be replaced by internal regulations. The important thing is that the rules defining who can vote exist. Consequently, in item 5, the important thing is to ensure that voters have access to the necessary infrastructure to exercise their political will. It is also through this principle that it is necessary for the voting system to ensure that the voter is authenticated to vote.

2.1.2 Principle of voting eligibility

Voters must be able to register and authenticate themselves in order to cast their vote. This principle does not conflict with the previous principle, since in order to be considered a voter, the individual must meet the criteria laid out in the rules defining who can vote in those elections, and it also prevents a voter from being able to vote more than once.

2.1.3 Principle of incoherence

It must be guaranteed that the vote cannot be bought, and that the voter cannot be extorted. This principle can only be achieved if the voter is unable to prove that he or she voted for a particular proposal. In addition to the technical requirements for this, the public voting structure (e.g. polling booth) must ensure that the person cannot be coerced outside the system.

2.1.4 Principle of freedom of decision

The voter's freedom of decision can be violated if any party or candidate advertisement is broadcast during the casting of the vote. In the traditional electoral process, no advertising is permitted at the polling station. Similarly, in an electronic voting process, no election advertising should be permitted on the voting site or generally on the system interface.

2.1.5 Principle of the invalid voting option

The voter should have the freedom of preference preserved. This means that if none of the available options appeals to him, he should be able to cast an invalid (null) vote.

2.1.6 Principle of equality among candidates

The interface of the system should not favor or disadvantage candidates. This principle is similar to the principle of Freedom of Decision, but with a focus on candidates. The positioning of ballots in a traditional election and the display of candidates on a computer screen should provide equality among candidates.

Moreover, all candidates should have equal access to the transparency of the election. That is, they should all have access to the same tools and information to check and audit the correct functioning of the election process.

2.1.7 Principle of equality among voters (One voter, one vote)

The principle of Equality Among Voters states that all votes should have the same value. From this premise, three pieces of information are derived. The first is that there should be no system of weights where one individual's vote is worth more than another. The second is that a voter can cast one, and only one, vote. Finally, Phillips and Spakovsky (2001) argue that in the context of electronic voting, access to and familiarity with the technology may give a certain group of voters an advantage when casting a vote. Also, for this reason, access to the tools to cast the vote should be universal, as discussed in the Isomorphism principle (Section 2.1.1). On the other hand, for people who for whatever physical reason (comorbidity, geographical distance) cannot attend a traditional polling place, the e-voting system can help make the principle of Equality Among Voters true.

As aforementioned in Section 2.1, Gritzalis (2002) claims that it is apparently impossible to guarantee the principle of "One voter, One vote". However, one possibility to guarantee such requirement is ensuring the e-voting system to register that the voter cast the vote. Thus, the system can prevent a voter from casting multiple votes. On the other hand, concerning impaired people or people unfamiliar with the technology, the organizing institution may have some campaigns or initiatives to assist those people.

2.1.8 Principle of secrecy

This principle is linked to the principle of incoherence and freedom of decision (Section 2.1.4). Secrecy is a necessary requirement to ensure free political decision. Once cast, the vote should be irreversible and not even the voter himself should be able to recover his decision.

In traditional electoral systems, secrecy is guaranteed by physical barriers, however, Gritzalis (2002) believes that in e-voting systems secrecy can be compromised, and that the following requirements must be explicitly met:

- Secrecy of the vote must be guaranteed in the casting, in the transport (in this case by network), in the reception and counting;
- None of the actors involved (organizers, candidates, and voters) should be able to make the connection between a voter and a vote;
- There should be a well-defined separation between the registration and authentication processes and the vote-casting and collection processes;
- No voter should be able to prove that he or she voted for a particular candidate.

It is pertinent to point out that in any remote voting system, there is no guarantee that the individual will not be coerced at the time of casting the vote, for this reason, when designing e-

voting systems, external protective measures should be discussed, such as voting booths (GRITZALIS, 2002). As mentioned, these principles are contradictory since some measures create a conflict with the Equality principle, which may exclude those people who were unable to move to the voting center. Thus, the organizer should provide some assistance program for those with special needs, as stated in Section 2.1.7.

2.1.9 Principle of unmonitored voting

During the electoral process, third parties cannot monitor voting. In traditional elections, this principle is enforced with measures that ensure that the ballot box is only opened at the time of vote counting. In an electronic voting context, it must be ensured that the computed votes can only be read after a certain moment, which indicates the end of the reception of votes. If this does not occur, the entity storing the votes may perform monitoring.

2.1.10 Principle of transparency

The e-voting election process should be as transparent as possible and the faith in the system as low as possible. That is, in an ideal situation, all actors involved (organizers, candidates, and voters) should be able to understand how the election takes place. Gritzalis (2002) emphasizes that faith in the system is an inherent factor in electronic voting, since an ordinary citizen has no knowledge about how computer systems work. Although the common voter does not know how computer systems work, the e-voting system should be audited by reliable third parties, as it happens in the Brazilian electoral system, for instance.

Therefore, it is important that the electronic electoral system be transparent and that the processes involved be explained, even if in a high-level language. This characteristic must accompany the system itself.

2.1.11 Principle of verifiability and accountability

Verifiability is the ability of voters, candidates, organizers, and independent observers to verify that the electoral process occurs in the correct manner. Accountability relates to the system's ability to produce logs and monitoring of all election operations, so that any individual can search for inconsistencies.

However, the voter's ability to verify that his or her vote has been counted and counted completely contradicts the principle of Secrecy (Section 2.1.8) and Incoherency (Section 2.1.3). In general, electoral systems (both traditional and electronic) make a trade off and give up some of the verifiability and accountability to maintain the secrecy of the vote.

2.1.12 Principle of simplicity

The principle of Simplicity states that the system should be as simple as possible. Simplicity, then, coupled with the principle of Transparency (Section 2.1.10), seeks to ensure that the system is user-friendly, that is, simple, accessible, and understandable to the user. However, due to principles such as Security and Reliability (Section 2.1.13), the system's source code is unlikely to be simple, given the inherent complexity of the system.

2.1.13 Principle of reliability and security

This principle permeates several other principles mentioned above. Reliability means that the election result is truly a reflection of the will of the electorate, so the system must ensure that the election result is in accordance with the votes cast and that these have not been modified and are true to what the voters want. No votes cast should be allowed to be excluded, and invalid votes should be counted, in accordance with the Invalid Vote Option principle. Just as in Secrecy (Section 2.1.8), Reliability and Security should also be enforced in all stages of the voting process (i.e., casting, transportation, reception and counting).

Security refers to ensuring the secrecy, integrity, and availability of the system. The process of registration and authentication of actors must be done in a secure manner and the system must be available and protected against denial-of-service attacks, intentional or not, since the unavailability of

the system hurts the right of the voter to cast his vote.

In a sense, Security counteracts the Simplicity principle since security processes inevitably end up increasing the complexity of the system.

Based on the principles of Reliability, Security (Section 2.1.13), Verifiability and Accountability (Section 2.1.11), Gritzalis (2002) proposes the following requirements:

- There must be hardware and software certification processes;
- All infrastructure as well as system functionality should be verifiable and comply with Kerckhoff's requirement in order to remain secure even if it is open-source (KATZ; LINDELL, 2020; DACHSELT; SCHWARZ, 2001);
- All operations should be monitored and generate system logs, except those that might break the secrecy of the vote;
- The infrastructure should be open for inspection by authorized agents;
- Voters and candidates should be assured that there were no malpractices during the process.

2.2 Consensus in distributed systems

A distributed system is a collection of independent computers that cooperate to solve a problem that cannot be solved individually (KSHEMKALYANI; SINGHAL, 2008). More precisely, the entities of a distributed system have the following characteristics:

- Independent physical clocks: The clocks in the network are not necessarily synchronized; this becomes an obstacle for managing the order of the exchanged messages. This problem was the motivation for Lamport (1978) work that gave rise to logical time theory;
- Independent Memories: Entities do not share memory, i.e., all communication depends on messages;
- Autonomy and heterogeneity: Processes have different speeds and can run on different operating systems, with different implementations.

Since the communication medium is the most important aspect of the network, it is necessary to classify the errors that can occur during message transmission. The worst type of error in this context is the byzantine failure, defined by Lamport (1983). In this fault, nodes can show any arbitrary error in the message exchange, including purposeful alteration of the content. In a variation of this problem, the Byzantine failure with authentication, the same errors can occur, but the messages can be verified through signatures.

One of the major problems in distributed systems is the consensus among the nodes. That is, how a network of nodes agrees on a certain value. In the context of Byzantine failures, the best known proposed algorithm is Practical Byzantine Fault Tolerance, or PBFT, proposed by Castro, Liskov, et al. (1999). In this algorithm, it is possible to guarantee consensus on up to $\frac{n-3}{3}$ malicious nodes, where n is the total number of nodes. Another consensus algorithm proposed later is Proof-of-Work.

Blockchain is an append-only data structure, meaning that the only modification allowed is the addition of more data. Data is entered in blocks, so that each block has a hash of the previous block (NAKAMOTO, 2008).

Since the network has no central entity, the nodes need to decide together which one will add the next block to the chain. The most intuitive solution is a lottery. However, this form of random selection is susceptible to attacks and requires some node to assume a centralizing role (ZHENG et al., 2017).

The Proof-of-work algorithm (DWORK; NAOR, 1993) is then used, where nodes need to prove that they have performed certain work to generate a new block and add it to the network. For this algorithm, work can also be interpreted as time expenditure, expressed in CPU cycles. In later work, the idea of the algorithm is extended so that the node can spend other resources, such as space or digital coins (DZIEMBOWSKI et al., 2015; GAŽI; KIAYIAS; ZINDROS, 2019).

2.3 Time-Based encryption

An interesting problem in the area of cryptography is the study of how to create a way to block access to a piece of information for a period of time, and after that period the information becomes available

without requiring an action from the person who blocked it. The problem is also sometimes described as "sending a message to the future".

Some approaches have been proposed to deal with this problem. One line of research proposes models that assume a trusted third party is responsible for releasing decryption keys at the correct time (RIVEST; SHAMIR; WAGNER, 1996; BONEH; BOYEN; GOH, 2005; CHEON et al., 2008). Another line of research uses the idea that the receiver of the message needs to solve an expressive, but not impossible, computational problem to unravel the cipher (BONEH; NAOR, 2000; UNRUH, 2015). The difficulty of the problem defines the approximate time for the cipher to be revealed, and the problems themselves are chosen so that the computational effort cannot be parallelized.

Liu et al. (2018) defines that to have complete time-based encryption, the three criteria must be met simultaneously:

- **Non-interactivity**: The sender of the cipher is not needed for decryption;
- **Independence**: There should be no dependency on a trusted third party. That is, the issuer does not need to depend on a trusted third party to store the decryption keys until the time of the cipher opening arrives.
- **Resource savings**: Interested parties in revealing the cipher should not be forced to spend computational resources for this purpose. This means that a receiver should only wait until the planned release time of the cipher to discover the message, and all stakeholders should discover the message at approximately the same time regardless of computational power.

Liu et al. (2018) proposes a solution that theoretically meets all three requirements. The authors introduce the idea of a reference computational clock, based on the state of a public, constantly iterative distributed computation. As an instance of this clock, the authors use the Bitcoin network. Added with the reference clock, a witness encryption scheme is used (GARG et al., 2013)

3 Proposal

Following the principles proposed by Gritzalis (2002) described in Section 2.1, this section presents a distributed architecture for e-voting. The goal of this architecture is that the election participants depend as little as possible on a central authority to conduct the election. The election should be managed in a distributed manner, even among adversarial parties, as proposed in Table 1:

Table 1. Transfer of responsibility in a decentralized model

Action	Responsibility in centralized model	Responsibility in decentralized model
Manage Voters	Central Authority	Central Authority
Managing Candidates	Central Authority	Central Authority, candidates, observers
Casting Votes	Voter	Voter
Storing Votes	Central Authority	Candidates, observers
Perform counting	Central Authority	Candidates, observers
Auditing the system	Voters, candidates, observers	Voters, candidates, observers

Source: Author.

In this work, three types of roles played by the network nodes are defined:

Voters: these are the nodes responsible for casting votes. Following the Isomorphism Principle (Section 2.1.1), voters must be authenticated to cast the vote. It should be possible for the voter to ensure that his vote has been computed, but it should not be possible for him to generate evidence of the content of his own vote (Principles of Verifiability and Incoherence, respectively). At the end of the election, voters should be able to verify and audit the vote count.

Hosts: are the nodes that store the votes. Candidates (or parties) must play the role of hosts. Assuming that an election has at least two candidates, then there must be at least two hosts. Other individuals, entities or observers with an interest in being hosts can also play the role.

Certificate Authority: Node responsible only for issuing and verifying digital certificates. It is the central entity (physically it can be distributed) with reduced power when it comes to issuing and counting votes. It is an identity and authorization query node for the other nodes. Before the election,

it is necessary that all future participants (voters, candidates, other interested parties) have public keys registered with the Certificate Authority. Throughout the paper it will be abbreviated to CA.

Two major challenges arise with this model:

1. How should the hosts come to consensus on vote storage?
2. What mechanism will prevent hosts from reading votes before the voting is over?

The first problem can be addressed using the literature on consensus in distributed systems, inevitably capturing its advantages and disadvantages. Importantly, in this model, although the vote is secret, the actors involved must be properly authenticated, even if there are operations that protect the user's identity, complete anonymity is undesirable for the solution, because it would break the principle of Equality – i.e., "One voter, one vote" – among voters once the system wouldn't even know if a particular individual casted a vote. As a consequence, one might vote more than once, breaking the principle of Equality (Section 2.1.7). However, complete anonymity wouldn't impose that votes have different weights. This means that the messages exchanged must be authenticated in one way or another. That is, among the possible failure models in distributed systems, the goal is for the proposed model to be Byzantine failure tolerable with authentication.

The second problem is more complex. In other contexts, when an actor needs to store information in an untrusted repository, that information is encrypted; to decrypt it, an action by the actor himself or another actor authorized by him is required. The problem arises in the context of an election, when at a certain point in the electoral process, the vote needs to be disassociated from the voter. In this case, it is necessary that the encryption protecting the vote be broken only in the counting and result stage, without action from the actor that encrypted it, which in this context is the voter. Ideally, this encryption system needs to fulfill three properties simultaneously:

- The voter does not need to be present for decryption;
- The system cannot depend on a third party who will keep the decryption keys;
- The parties interested in decrypting the cipher cannot have a significant advantage over each other according to their computing power. That is, all parties must be able to decrypt the vote at approximately the same time and never before the start of the vote counting and result stage.

Therefore, this encryption needs to be time-based. For the proposed model, it is defined that the encryption mechanism is based on the work of Liu et al. (2018), because the three properties above are analogous to the properties of his proposal. However, other approaches are also valid such as a time-lock encryption where the central authority keeps the secret decryption key until the end of the voting process. The main drawback of such approach is that the CA is a central actor that hold all the secret decryption keys and may be attacked (e.g., it may suffer a Denial of Service attack or have all the decryption keys leaked).

3.1 Rules, constraints, and notations

To serve as a basis for building and understanding the proposed model, it is necessary to define the set of rules and assumptions of the proposal:

- The system consists of a distributed set of nodes;
- The notation for the asymmetric key pair of a node n is (PK_n, PR_n), where PK_n is the public key and PR_n is the private key. The notation $E(m, k) = c$ means that message m is encrypted with key k and generates a cipher c. The notation $D(c, k) = m$ means that cipher c is decrypted with key k and generates message m;
- Nodes can participate in an event called Election. During an Election, the following rules exist:
  - An election is composed of a question q, which can be answered with a proposal p, which may be a blank or null vote. The set of proposals is P;
  - Each host can issue only one proposal p;
  - Each voter may cast only one vote v, to show support for only one proposal of any kind. The notation for extracting a proposal from a vote is $fv(v) = p_n$. The set of votes computed from an election is V and $V*$ is the set of valid votes (i.e., blank and null votes aren't considered valid votes) computed from the same election;
  - The number of candidates must be less than or equal to the total number of hosts;

  - The number of candidates must be greater than 1.
- Nodes are required to perform the voting steps over a computer network (i.e., online);
- The nodes use the NTP (Network Time Protocol) to synchronize the clocks;
- Message exchanges between nodes are done with end-to-end encryption.

3.2 Phase one: election setup

An election is commonly divided into three stages (Figure 1): a preparation, where the characteristics of the election are defined; the vote casting period, which occurs sometime after the close of the first stage; the vote counting, which can occur immediately after the election ends. The divisions between these time periods will be called t_0, t_1, t_2.

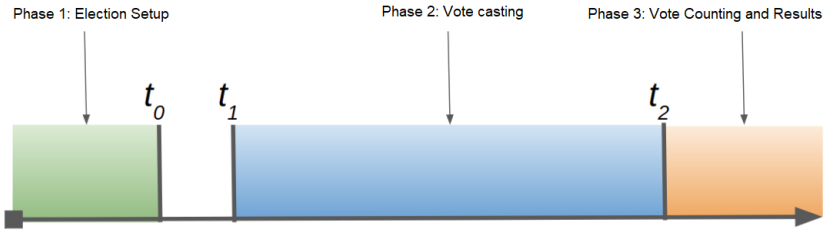

Figure 1. Election timeline.
Source: Author.

Step 1 (Creating the election): An election is created from any node in the network. The election is a 6-tuple e:

$$e = \langle q, t_0, t_1, t_2, f, bs \rangle$$

The first element is the question q. The question q represents the question that is to be answered by the election. Examples of questions are *"Which party should run the student center?"* or *"Should the United Kingdom leave the European Union?"*. The second is a future timestamp t_0, which indicates a maximum deadline for nodes to submit proposals. The third is a future timestamp t_1, which indicates the start of receiving votes. The fourth is another future timestamp t_2, indicating a maximum time limit for votes to be received. Although this is a distributed system, it is not necessary to use a Lamport's (1978), logical clock scheme because there is a tolerable clock synchronization difference such that using NTP is sufficient. The fifth element is the function $fw : V \longrightarrow P$ that defines the winning proposal based on the set of votes. For example, let A be the set of votes on a proposal p_n. That is, $v \in A$ if $fv(v) = p_n$. If it is decided that the winning proposal is the one with a simple majority, then $fw(V) = p_n$, if $|A| > 0.5|V|$. The sixth element is a parameter $bs \in \mathbb{N}$ e $bs < |E|$, where E is the set of voters, which indicates the size of the disclosure blocks. Hosts must disclose votes in groups to balance secrecy and verifiability. This process is detailed in Section 3.3.2.

The initiating node must make e public to the other nodes. To ensure the authenticity of the proposal, the node must sign the message.

Step 2 (Proposal and Host Registration): Upon learning of an election e, any node n can issue a proposal and become a candidate before time t_0. To do this, in addition to proposition p_n, it must define a host address h_n from which it will receive and store votes. The candidate must then make public p_n and h_n and sign the message containing both values to guarantee its origin. Others interested in maintaining a storage node (i.e., host) can also perform the same process, but without generating the proposal.

3.3 Phase two: voting and storing

Step 1 (ID Generation): With the arrival of t_0, the first phase is closed and, at this point, a voter node n has the information about the election with a copy of the structure e, a list of proposals $p_1, p_2 ... p_n$ and a list of host addresses $h_1, h_1 ... h_k$, so that $n \leq k$. The voter then needs to generate a temporary ID that meets the following requirements:

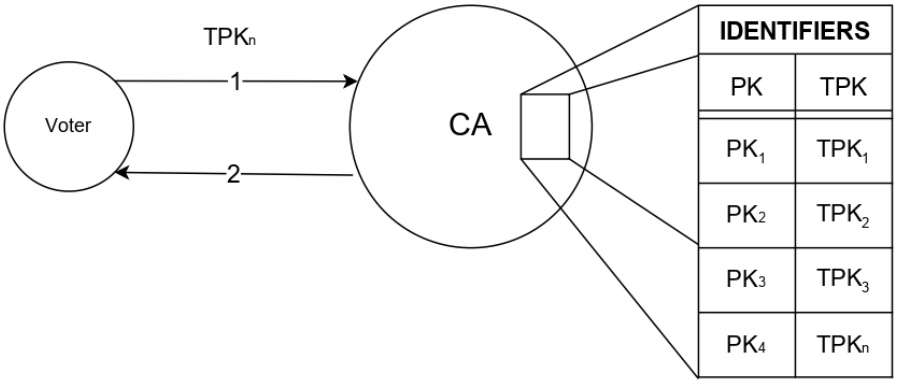

Figure 2. Step 1: Sending the temporary keys to the CA.
Source: Author.

- it must be possible for the CA to verify that the ID was generated by a valid voter and that this ID is the only one generated by that voter for that election. That is, each voter must have one and only one ID;
- It should not be possible for other participants in the election to associate an ID with a voter;
- Anyone should be able to verify whether an ID is valid for that election.

To comply with the third requirement, the voter must generate a pair of keys $I = (TPK_n, TPR_n)$ that will serve as a temporary identification and register it with the CA TPK_n, (message 1 in Figure 2). The CA must check if any identification of that voter already exists for that election and respond to the voter if the registration was done successfully (message 2 in Figure 2).

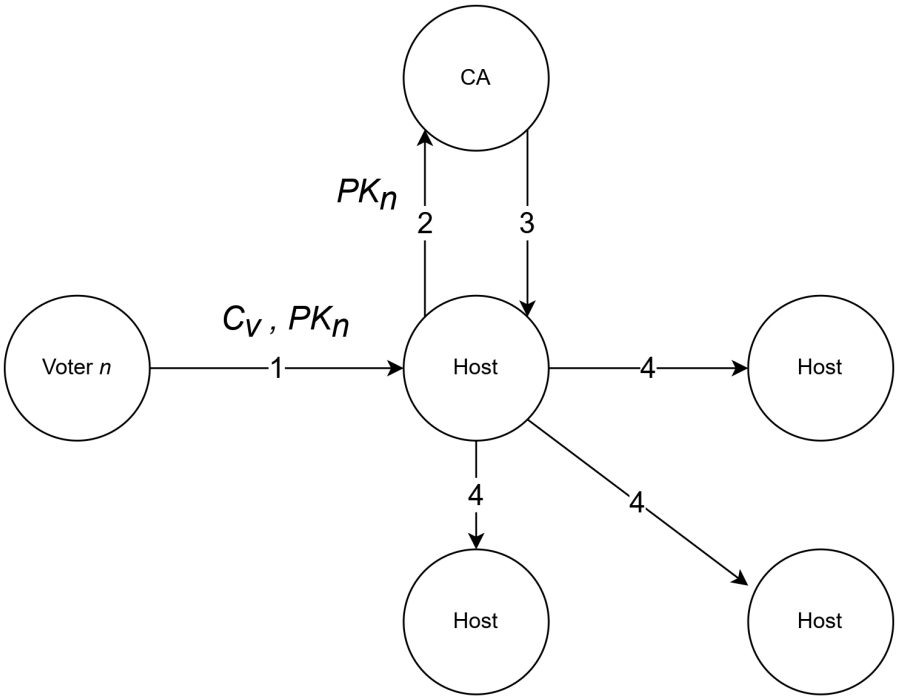

Figure 3. Step 2: message exchange sequence for vote submission.
Source: Author.

Step 2 (Sending the vote): After time t_1, the voter can cast his vote v, encrypted, and following the requirements described at the beginning of this section. To do so, he must use a time-based encryption scheme, grounded in the work of Liu et al. (2018). The cipher c_v represents the cipher of the vote after encryption.

The voter must send c_v, signed with I, to a host h_i any of the election (message 1 in Figure 3). The host should check with CA whether the public key of I is valid (message 2 in Figure 3). If it is

not, the message should be ignored. If it is valid, the host must share the signed cipher c_v with all other hosts, using a consensus algorithm (message 4 in Figure 3).

3.3.1 Consensus

As discussed at the beginning of this section, the consensus between hosts needs to be Byzantine fault tolerant with authentication. In addition, the requirements that the algorithm must meet in the context of distributed election will be defined.

The first requirement is that if the voter sends a signed cipher c_v to a host h_i, it must receive confirmation of receipt by another host $h_{j,j\neq i}$. This is done to assure the voter that there has been a consensus and that the message has been distributed. The second requirement is that each time a host receives a signed cipher c_v, it must verify the validity of the signature's public key with CA. In addition, the host needs to verify whether this is the first vote of the voter or if there is already another vote agreed upon consensus. This requirement serves as host verification that the message originated from a valid voter. Any consensus algorithm that meets these requirements, for example PBFT, can be used in this phase.

3.3.2 Confirmation and display of votes

After the consensus is performed, some host $h_{j\neq i}$ must send a confirmation of receipt to the voter. When the voter receives the confirmation from a host $h_{j\neq i}$, the vote submission operation is completed.

After a certain bs amount of vote ciphers are received, the hosts must publicly disclose these ciphers in a table, along with the accompanying TPK public key values. The key and cipher values cannot be linked with each other and therefore the order of the columns must be random (see Figure 4). The disclosure of these tables in this way is done for two reasons that seek to meet the verifiability principle. The first is that the ciphers must be public before they are opened. Moments before the cipher is decrypted, everyone will have a copy of the vote set. The second reason is that if the cipher set is accompanied by a set of temporary public keys, even if in random order, it is possible for anyone to check with the CA whether the keys are valid.

3.3.3 Phase summary

The complete sequence of actions for this phase is:

1. The voter generates $I = (TPK_n, TPR_n)$;
2. The voter sends a registration message from I to CA;
3. CA sends a confirmation message to the voter;
4. The voter sends the c_v message, signed with I to a host h_i;
5. The host checks with CA whether TPK_n belongs to a valid identity;
6. If the answer is positive, the host sends in groupcast c_v to the other hosts;
7. All other hosts $h_{j\neq i}$ perform the verification of TPK_n with CA;
8. At least one host $h_{j\neq i}$ sends the acknowledgement to the voter and to h_i;
9. The voter waits to receive confirmation from at least one host by a fixed timeout or as a parameter specified at election creation. If this does not occur, the voter repeats the send to a host $h_{j\neq i}$
10. h_i waits to receive confirmation from at least one $h_{j\neq i}$ by the specified timeout to accept the vote. If this does not occur, the vote is discarded.
11. For any host, after a number of bs votes accepted, a table is published with a column of TPK and a column of c_v, both in random order.

3.4 Phase three: counting the votes

After timestamp t_2 arrives, vote collection must be terminated. At this point, each host has already released tables with the consensus vote figures, and all interested parties can access them. If t_2 arrives and the number of votes is less than bs, the hosts release a smaller table. Following what was proposed at the beginning of this section, after time t_2, the ciphers are undone, and the winning proposal can be extracted from the set of votes using the $fw : V \longrightarrow P$ function defined at the time of election creation. Therefore, the counting is not the responsibility of only one actor, the verification

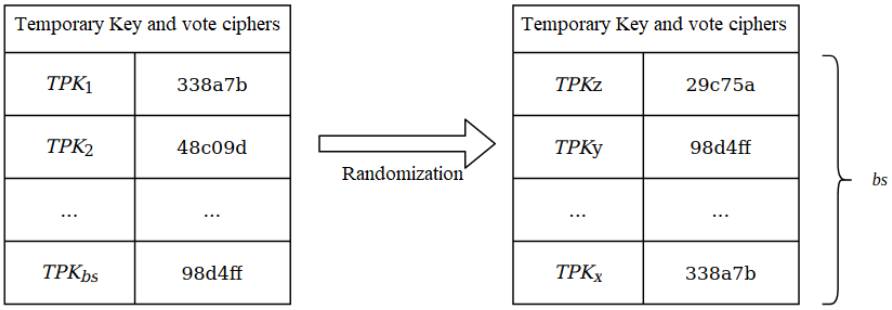

Figure 4. Phase 2: Randomization of the voting cipher table
Source: Author.

and counting of the votes can and should be performed by multiple entities.

4 Final considerations

A proposed architecture for a distributed electronic election was presented in Section 3. The goal of this architecture is to study the feasibility of a system with reduced central entity responsibility and the technological requirements needed for this.

In the work by Hjálmarsson et al. (2018), the voter can retrieve his vote using an identifier, which makes the system more adjustable. The present work has shown that the literature does not consider this action recommendable, as it makes one of the principles of secure election completely impossible and allows the voter to be coerced (see Section 2.1.3). It is also discussed in this paper how the vote can be secretly stored until the time the election is over, which was not presented in Hjálmarsson et al. (2018).

The proposal of this paper also has an advantage in protection against coercion over the architecture of Hardwick et al. (2018). The author presents a system where, in order to evade coercion, the voter may change his vote after posting. This does not guarantee voter security and goes against the recommendation in the literature.

4.1 Problems and possible solutions

When distributing the election process, advantages are gained but new problems arise. In this section, some of these problems are described and possible solutions discussed.

4.1.1 Malicious CA

The potentially dangerous dependence on a central entity was one of the motivations for this work. The central entity was therefore reduced to a Certificate Authority, responsible only for managing the identities of the participants. Even so, if the CA is compromised, the entire election can become untrusted. If the CA is malicious and wants to benefit a specific host, the moment the host queries the CA about the validity of a temporary public key it may return the voter's original identity. Also, if the host sends the tables with the non-randomized columns to CA, it can relate a table to a vote.

In this case, the threat mitigation is similar to that of a central entity in a traditional election. Voters and parties should be guaranteed to participate in CA audit processes and all actions that cannot compromise the identity of voters should be monitored and logs generated. More specifically, it must be guaranteed that communication from the CA to the hosts must be only those described in the architecture and that there will be no parallel communication channels.

One possibility is to have a distributed CA through a consensus algorithm in such a way that the commitment of a small portion, which depends on the consensus algorithm, of the set of CAs would not invalidate the election, similarly to what happens with a blockchain network (DZIEMBOWSKI et al., 2015; GAŽI; KIAYIAS; ZINDROS, 2019; NAKAMOTO, 2008).

4.1.2 Coercivity in online election

In the presentation of the principle of Incoherence, it is said that the secrecy of voting must be provided even outside the system. Gritzalis (2002) states that an inherent problem with electronic voting systems where the vote can be cast over the Internet is that there is no guarantee that the voter will not be coerced at the time of casting. That is, even if a perfect electronic system exists, the voter can be coerced out of it.

The solution to this problem is also the same as in traditional electoral systems: a physical structure with a voting booth that isolates the voter at the moment of casting the vote. In the context of smaller elections, where the outcome of the election is not as impactful or the power of coercion of the stakeholders is reduced, this issue can be relaxed. However, it is important to note that this will always be a point of failure.

4.1.3 Consensus in two-party systems and coalitions

One of the crucial issues in distributed systems without central authority is consensus. The Practical Byzantine Fault Tolerance (PBFT) algorithm by Castro, Liskov, et al. (1999) guarantees consensus on up to $\frac{n-3}{3}$ malicious nodes, where n is the total number of nodes. In other words, for a network to tolerate at least one malicious node using PBFT, it needs at least four nodes in the network. Proof-of-Work in Bitcoin guarantees consensus as long as one entity does not control more than 50% of the computational power of the network. For this algorithm, a network with two nodes is possible, but they need to maintain a constant balance of computational power, which is highly unlikely when the nodes are adversarial.

These limits make consensus impossible for completely bipartisan systems. Furthermore, suppose (a completely possible scenario) that in a multi-party system the parties can make alliances so that if a party realizes that it has no chance of winning, it will offer support to another party with better chances. Inevitably the two parties most likely to win will form two large coalitions, reducing the problem to a two-party system. Consensus in this case is only possible if it is guaranteed that there will be completely impartial entities in the network.

4.1.4 Solution complexity

The use of asymmetric encryption, time-based encryption, consensus algorithms and moderately hard functions makes understanding the architecture non-trivial. This is against the principle of Simplicity (Section 2.1.12). On the other hand, security mechanisms are inevitably complex (STALLINGS, 2017). This problem can be minimized by manuals and descriptive texts that present an overview of election processes. Also, due to the inherent complexity, one possibility is to have specialists from society and audit/comptrollership institutions acting as surrogates, just as it happens in the Brazilian electoral system where institutions such as universities, specialists, Federal Police, Federal Prosecution Service, National Congress, and political parties audit the electoral system.

4.2 Future work

The first step is part of the requirements listed in Section 2.1. To meet the principles of Transparency (Section 2.1.10), Security (Section 2.1.13), and Verifiability (Section 2.1.11), it is necessary that professionals in the field and other researchers perform critical and independent analysis of this work, uncovering possible unmapped flaws and validating processes and technologies.

The second step is to implement a system based on the proposed architecture and conduct performance and security analyses. Other researchers should also audit the implementation. Also in this implementation, the feasibility of the time-based encryption proposed by Liu et al. (2018) should be analyzed.

Another future work is to study to see if there is a better way to anonymize the voter. In the ideal scenario, an identifier needs to be generated that simultaneously meets the following requirements:

- The identifier was generated from the set of valid voters, without specifying which voter;
- A voter cannot generate two identifiers.

It should be studied if it is possible to meet both requirements simultaneously.

5 Conclusion

More theoretical studies on the development of distributed e-voting systems should still be done before an implementation. In particular, better solutions related to vote secrecy and incoercibility should be proposed. Despite the division of responsibilities, the election can still be manipulated if the central entity is malicious.

Time-based encryption presents potential as a solution for distributed vote storage, however, more studies on feasibility and performance need to be done.

The literature on distributed consensus is already established, yet it should be noted that in two-party systems consensus is impaired, both in more traditional solutions and in blockchain-based consensus algorithms. Despite the potential of distributed architecture, much work still needs to be done.

References

ARANHA, Diego F.; GRAAF, Jeroen van de. The Good, the Bad, and the Ugly: Two Decades of E-Voting in Brazil. *IEEE Security Privacy*, v. 16, n. 6, p. 22–30, 2018. DOI: 10.1109/MSEC.2018.2875318.

BONEH, Dan; BOYEN, Xavier; GOH, Eu-Jin. Hierarchical identity based encryption with constant size ciphertext. In: SPRINGER. ANNUAL International Conference on the Theory and Applications of Cryptographic Techniques. [S.l.: s.n.], 2005. p. 440–456.

BONEH, Dan; NAOR, Moni. Timed Commitments. *Advances in Cryptology – CRYPTO 2000*, p. 236–254, 2000. Available from: https://doi.org/10.1007/3-540-44598-6%5C_15.

CASTRO, Miguel; LISKOV, Barbara, et al. Practical byzantine fault tolerance. In: 1999. OSDI. [S.l.: s.n.], 1999. v. 99, p. 173–186.

CHEON, Jung Hee et al. Provably Secure Timed-Release Public Key Encryption. *ACM Trans. Inf. Syst. Secur.*, Association for Computing Machinery, New York, NY, USA, v. 11, n. 2, May 2008. ISSN 1094-9224. DOI: 10.1145/1330332.1330336. Available from: https://doi.org/10.1145/1330332.1330336.

DACHSELT, F.; SCHWARZ, W. Chaos and cryptography. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, v. 48, n. 12, p. 1498–1509, 2001. DOI: 10.1109/TCSI.2001.972857.

DWORK, Cynthia; NAOR, Moni. Pricing via Processing or Combatting Junk Mail. *Advances in Cryptology - CRYPT0 '92*, p. 139–147, 1993. Available from: https://doi.org/10.1007/3-540-48071-4%5C_10.

DZIEMBOWSKI, Stefan et al. Proofs of space. In: SPRINGER. ANNUAL Cryptology Conference. [S.l.: s.n.], 2015. p. 585–605.

GARG, Sanjam et al. Witness encryption and its applications. In: PROCEEDINGS of the forty-fifth annual ACM symposium on Theory of computing. [S.l.: s.n.], 2013. p. 467–476.

GAŽI, Peter; KIAYIAS, Aggelos; ZINDROS, Dionysis. Proof-of-stake sidechains. In: IEEE. 2019 IEEE Symposium on Security and Privacy (SP). [S.l.: s.n.], 2019. p. 139–156.

GRITZALIS, Dimitris A. Principles and requirements for a secure e-voting system. *Computers & Security*, Elsevier, v. 21, p. 539–556, 2002. Available from: https://doi.org/10.1016/S0167-4048(02)01014-3.

HARDWICK, Freya Sheer et al. E-Voting With Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). [S.l.]: IEEE, 2018. Available from: https://doi.org/10.1109/Cybermatics%5C_2018.2018.00262.

HJÁLMARSSON, Friorik et al. Blockchain-Based E-Voting System. In: IEEE 11th International Conference on Cloud Computing(CLOUD). [S.l.]: IEE, 2018. Available from: https://doi.org/10.1109/CLOUD.2018.00151.

KATZ, Jonathan; LINDELL, Yehuda. *Introduction to modern cryptography*. [S.l.]: CRC press, 2020.

KSHEMKALYANI, Ajay D.; SINGHAL, Mukesh. *Distributed Computing: Principles, Algorithms, and Systems*. 1. ed. USA: Cambridge University Press, 2008. ISBN 0521876346.

LAMPORT, Leslie. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, v. 21, p. 558–565, 1978. Available from: https://doi.org/10.1007/s10623-018-0461-x.

LAMPORT, Leslie. The weak Byzantine generals problem. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 30, n. 3, p. 668–676, 1983.

LAUER, Thomas W. The risk of e-voting. *Electronic Journal of E-government*, Citeseer, v. 2, n. 3, p. 177–186, 2004.

LIU, Jia et al. How to build time-lock encryption. *Designs, Codes and Cryptography*, v. 86, p. 2549–2586, 2018. Available from: https://doi.org/10.1007/s10623-018-0461-x.

NAKAMOTO, Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Available from: https://bitcoin.org/bitcoin.pdf.

PHILLIPS, Deborah M.; SPAKOVSKY, Hans A. von. Gauging the Risks of Internet Elections. *Communications of the ACM*, ACM, v. 44, p. 73–85, 2001. Available from: https://dl.acm.org/doi/10.1145/357489.357512.

RIVEST, Ronald L; SHAMIR, Adi; WAGNER, David A. Time-lock puzzles and timed-release crypto. Massachusetts Institute of Technology. Laboratory for Computer Science, 1996.

SAMPIGETHAYA, Krishna; POOVENDRAN, Radha. A framework and taxonomy for comparison of electronic voting schemes. *Computers & Security*, Elsevier, v. 25, p. 137–153, 2006.

SPRINGALL, Drew et al. Security Analysis of the Estonian Internet Voting System. In: PROCEEDINGS of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014. (CCS '14), p. 703–715. ISBN 9781450329576. DOI: 10.1145/2660267.2660315. Available from: https://doi.org/10.1145/2660267.2660315.

STALLINGS, Willian. *Cryptography and Network Security: Principles and Practice*. Global Edition. Edinburgh Gate, Harlow, Essex CM20 2JE, England: Pearson, 2017.

UNRUH, Dominique. Revocable quantum timed-release encryption. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 62, n. 6, p. 1–76, 2015.

ZHENG, Zibin et al. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *IEEE International Congress on Big Data (BigData Congress)*, p. 557–564, 2017. Available from: https://doi.org/10.1109/BigDataCongress.2017.85.

ZISSIS, Dimitrios; LEKKAS, Dimitrios. Securing e-Government and e-Voting with an open cloud computing architecture. *Government Information Quarterly*, Elsevier, v. 28, p. 239–251, 2011. Available from: https://doi.org/10.1016/j.giq.2010.05.010.

Author contributions

João Marcos Soares: Conceptualization, Data curation, Investigation, Methodology, Validation, Writing – original draft, Writing – review and editing; **Rafael Oliveira Vasconcelos**: Methodology, Project administration, Supervision, Resources, Validation, Writing – original draft, Writing – review and editing.