Towards a Fully Mobile, blockchain-based Electronic Voting system using Non-Fungible Tokens

Ricardo Lopes Almeida[1], Fabrizio Baiardi[2], Damiano Di Francesco Maesa[3], and Laura Ricci[4]

[1, 2, 3, 4]Dipartimento di Informatica, Università di Pisa, Italia
[1]Università di Camerino, Italia

September 17, 2024

Abstract

Research in electronic voting systems has been constant and fruitful since the 1970s, when the introduction of commercial cryptography provided the tools and methods for such critical operations. Yet, save for a few exceptions for testing purposes, no remote electronic voting system has been employed systematically.

A significant breakthrough occurred in 2009 with the introduction of the decentralised computational paradigm through Bitcoin, the world's first cryptocurrency, and the distributed ledger technology that supports it. Researchers soon started applying decentralised concepts and using distributed ledger's features to propose e-voting systems from a decentralised approach, which revealed itself superior right from the start. As a consequence, the field moved to use the new paradigm soon after, and new proposals employing the latest distributed ledger features are still appearing.

This article characterises the evolution of e-voting research through the years and presents a novel architecture based on smart contracts and Non-Fungible Tokens (NFTs), a recent addition to the distributed ledger ecosystem. We explore the inherent advantages of this new concept, as well as its development framework, to present the architecture of a fully mobile, decentralised electronic voting system using NFTs as the main vote abstractor.

1 Introduction

Modern democracies are critical to technological advances, as they provide citizens with levels of comfort and security that allow them to divert their time and energy away from basic survival towards research and innovation. This is a transversal effect across society, with all sorts of mundane activities getting progressively easier and even automated, with each technological advance. Yet, the exercise that keeps these same modern democracies running smoothly still resists innovation. Most elements that characterize a modern society - banking, farming, food processing, education, healthcare, etc - have improved thanks to technology. All except voting. Except for a few minor upgrades on the voting act itself, the fundamental mechanics remain almost unchanged.

From dropping clay shards in a pot, to selecting an option in a touchscreen, there has not been a change significant enough to bring this process to the same levels of practicality of other activities. Votes submitted in an electronic medium can be counted faster, but other than that, current electronic voting still limits voters geographically and does not contribute to make elections easier or cheaper to organize.

This is not an indication of a lack of interest in the research community. In fact, the modernization of voting systems has been an active area of research in general and the digitalization of voting systems has been researched actively since the advent of commercial cryptography. A landmark article by Diffie and Hellman in 1976 [34] established the basis for commercial cryptography, a research area that was confined to governmental (i.e., military) applications until then. This publication spearheaded the development of several cryptographic tools, particularly schemes for symmetrical and asymmetrical encryption systems, as well as cryptosystems that were made available for the public based on diverse approaches to intractable problems that are trivial to solve in one direction but computationally unfeasible in the opposite. The *Rivest-Shamir-Adleman (RSA)* [92] cryptosystem, which relies on the factorization of large prime numbers, and the *ElGamal* [36] cryptosystem, which uses the intractability of computing discrete algorithms to establish cryptographic operations, are among the most popular ones.

In the specific context of e-voting, these advancements were followed by approximately three decades years of e-voting research based on a centralized, server-client paradigm that produced a significant number of academic proposals that used cryptographic tools such as encryption, digital signatures, one-way hash-function, etc. to create secure and transparent e-voting systems.

This was the state of e-voting research until 2009. At the end of that year, blockchain broke into mainstream through Bitcoin, the first cryptocurrency, introduced through a landmark paper [82]. After Bitcoin's solidification as the first truly digital currency, others followed, improving not only the currency feature, but adding support to different applications. Six years later, the Ethereum blockchain [33] was launched and with it the concept of blockchain virtual machine, a functional abstraction of the cumulative resources available in the network nodes, which could be used to run code synchronously - referred to as Smart Contracts - in a subset of the network nodes. The virtual machine abstracted by the Ethereum network was named *Ethereum Virtual Machine (EVM)* [6]. Though Bitcoin already offered some capacity to execute scripted code in a distributed manner, this was not its main feature and thus was quite limited in that regard, Ethereum was the first blockchain to offer explicit support for Smart Contracts and since more and more blockchains have been launched with similar capabilities. As a consequence, Smart Contracts became almost ubiquitous in the blockchain ecosystem. Other examples of blockchains with smart contract support created after Ethereum include *Cardano*, *Solana*, *Tron*, *EOS*, *Polkadot* and *Flow* [29].

The applicability of blockchain to the e-voting context was clear from early on, and e-voting research migrate to this new approach to take advantage of these features. The first academic articles detailing e-voting systems using a blockchain appeared in 2016 and, since then, the interest in this new research approach has increased steadily. Blockchain is still a technology under development, and new features based on this concept have been introduced at a fast pace in recent years.

This article presents a fully mobile, blockchain based electronic voting system that explores the concept of Non-Fungible Tokens (NFTs), one of the newest features put forward by the Smart Contract enabled Ethereum network. We discuss this concept in greater detail in Section 3.

The rest of this paper is organized as follows: a state of the art regarding the evolution of e-voting systems is presented in section 2. Section 3 introduces the central element in the solution presented, mainly from a technical point of view. Section 4 begins by detailing a generalist approach to the problem at hand, followed with a description of the two approaches considered. A security analysis of the proposed solution follows with Section 5. The paper concludes with section 6 where the main conclusions and open problems are discussed.

2 State of the Art

The history of e-voting systems research details an evolution process that was triggered by the first proposals resulting from the "commercialisation" of cryptography, up to the latest decentralised proposals using blockchain as the technological basis.

Cryptography has been used for some time but mainly within a military context, where secure communications are a critical element for battlefield success. Cryptography informally entered the public domain through the publication of [34], which proposed the computation of logarithms over a finite field with a prime number of elements as the intractable problem required to produce cryptosystem. This was the first proposal for a non-military, non-proprietary cryptosystem, and soon after, other articles detailing alternative solutions and exploring other sources for intractable problems followed, such as [96], [27], [36], and [92], which proposed the first commercial cryptosystems based on symmetrical and asymmetrical encryption keys. The applicability of these cryptographic tools in an e-voting context was fairly clear and the first e-voting proposals based on these concepts soon followed. These early systems are characterised by cryptographic tools derived from research on commercial cryptography, such as blind signatures [24], Mix-Nets [25], Homomorphism in threshold cryptosystems [96] and cryptographic proofs [46].

Research in e-voting systems progressed towards the establishment of a classification criteria that were then used to compare proposals from a security standpoint. Authors implemented criteria such as *accuracy*, *privacy*, *eligibility*, *verifiability*, *convenience*, *flexibility*, *mobility* and *robustness* using techniques obtained through research in commercial cryptography. A simple example that illustrates this process is the usage of asymmetrical encryption keys to encrypt voter data, thus protecting the *privacy* of the voter. A proposal that uses such scheme can claim that it establishes voter *privacy*. Yet, a formal definition of such criteria has notorious absent from related literature. [83] was among the first to attempt such characterisation, with subsequent publications, such as [43], [10], [59], [65], [69], [58], [8], and [26], continuing this trend. These articles followed the rationale that the more secure a system is, i.e., the more security criteria it implements, the more it can assure a user that he/she can trust his/her choice to it. Over time, these cryptographic tools became a fixture in all e-voting proposals in this initial 30-year window of research in centralised e-voting systems. This is a limiting paradigm since it constrains the whole system by establishing a single point of attack or failure, while also reducing system scalability, due in great part to the demand of a large amount of resources, such as primary and secondary storage, computational power, network bandwidth, etc., to implement these criteria.

Scalability is an important characteristic that can hinder a wide adoption of the proposed system. Contemporary elections can go from simple exercises, where the system is expected to process up to a thousand votes at one point, to national-wide events that require the processing of millions of votes instead. The relationship between the scalability of a system and the amount of available resources is quite evident in the analysed literature. Proposals from this era confirm that the ones that satisfy the most security criteria also establish a computationally complex and demanding system that is often limited to small-scale elections. As an example, in [30], [85], [57], and [84], this trade-off was shifted towards security and transparency at the expense of scalability; hence, these systems are all limited to small-scale elections, as the authors declare.

On the other side of this spectrum, proposals such as [15], [16], [89], [60], [87], [86], [73] or [80] present scalable systems that are significantly simpler than the ones considered in the last paragraph, but their adoption in large-scale elections results in a substantial sacrifice in security and transparency since these implement the lowest number of security criteria considered.

Several e-voting systems were evaluated in a real-world scenario. For this case, we are only interested in systems that address the criterion of *mobility*, i.e., a voting system that does not restrict voters geographically, in part because these proposals did not follow any of the academic ones that preceded them. Therefore, there was little interest in electronic proposals that limited their users to traditional polling places, since they infuse a degree of privacy and security that derives solely from the surrounding election apparatus. The few notable exercises in recent history were run in Canada in 2013 [47], Estonia in 2005 [52], Switzerland in 2005 [17], Norway in 2011 [38], France in 2012 [90], and Australia in 2015 [48].

The main discovery from this analysis is an overlap between academic proposals and real-world systems, or, more precisely, the lack thereof. Private companies developed most of the real-world remote solutions indicated by being awarded government development contracts. As a consequence, specific technical details are kept private as patented intellectual property, with any technical information about them available only from review reports, which makes any further analysis a difficult endeavour. The complexity and resource hungriness of centralised solutions are characterised by the requirement of government-scale investments to achieve minimal implementation. In large-scale elections, the scalability effort quickly grows out of the capacity of small organisations and individuals, leaving governments as the only organisations with enough resources to implement them. This generation of e-voting systems proved that the concept is sound and that it is possible to use electronic means to establish elections with greater accuracy and security for their participants.

A shift towards a decentralised approach to this issue initiated with proposals that used blockchain merely for transport and record, in part as a consequence of the "imposition" of using Bitcoin's blockchain, since it was the only mature public blockchain available at the time. Some proposals [110], [31], [19], [70], [95], [108], [35] or [12] used a script function available in Bitcoin transactions to add voting information to the blockchain data, namely the *OP_RETURN* function, which receives an 83-byte wide string as input, and adds it to the transaction metadata as the function's output (depends on the version of the Bitcoin protocol. The last one extended the size of the allowed output to 83 bytes. Earlier versions are more limited in size). Hence, it was used to write non-transactional data directly to the blockchain. This mechanism needs to go around the limited functionalities offered by early blockchain solutions whose scope of operations were limited to cryptocurrency transactions only. Furthermore, this method is infeasible for large-scale use because a Bitcoin transaction necessarily involves exchanges worth a considerable amount of money, as well as being notoriously hard-to-scale blockchain due to its high block rate. Bitcoin adds a new block every 10 minutes, which limits severely the rate of operations that this blockchain can withstand.

The popularisation of public blockchains attracted interest from other platforms, which triggered the development of software frameworks used to create and deploy custom blockchains with proprietary access control and offering more flexibility to applications. As such, researchers deployed custom-made solutions in customisable frameworks such as Hyperledger Fabric, Quorum, and Multichain. Some proposals, [63], [7], [23], [20], [109], [61], [81], [40], [62], [49], [56], [74], [111], [5], [51], [101] or [75] adopted private blockchains tailored to the voting application. As a drawback, the increased flexibility is paid for by a lack of network support. It is difficult to establish a privately accessible network with enough active nodes that can establish a satisfactory level of redundancy.

A significant breakthrough arrived with the introduction of the smart contract through the Ethereum blockchain, specifically through the implementation of a *Touring-complete* processing platform, named *Ethereum Virtual Machine*

(EVM), that can execute code scripts in a decentralised fashion by splitting and distributing the instructions through the active nodes in the network. This opened up new research avenues in the e-voting environment. A few years after Ethereum's debut, the first proposals used smart contracts to establish the logic of the system in a decentralised approach, such as [76], [64], [32], [44], [54], and [79].

Some researchers did base their solution on the Ethereum network, but they did not resort to smart contracts; instead, they explored the increased flexibility and additional application support of this network, employing similar methods as the ones in earlier proposals. Examples of this strategy can be found in [50], [103], [55], [67], [97], and [22].

The same time period saw several blockchain-based proposals for electronic voting systems in a real-world scenario. Small organisations are putting forward these proposals. But unlike the centralised approach, these solutions have significant overlap with the academic proposals considered. Among these real-world examples, we cite *Follow My Vote* [37], *TiVi* [99], *Agora* [2], and *Voatz* [102]. The remarkable difference between the nature of approaches regarding their real-world applications is a strong indicator of the potential of blockchain in this scenario. Real-world blockchain-based e-voting solutions follow the academic approach much closer than their centralised counterparts.

The real-world solutions indicated are end-to-end applications, i.e., they are ready to be used in an election, as long as their limitations are properly addressed (mostly related to scalability). But there are other proposals published in the public domain as protocols that can be used to set up an e-voting systems instead. These are not complete solutions, as the ones indicated thus far, but instead "recipes" that, if followed, can be used to establish a secure and transparent e-voting system. [68] presents a concise summary of the most relevant Ethereum based protocols in existence. Most of the logic employed in these protocols is already abstracted through smart contracts already deployed and publicly available in the Ethereum blockchain, such as the *MACI (Minimal Anti-Collusion Infrastructure)* protocol [21], *Semaphore* [91], *Cicada*, and *Plume*. It is important to notice that none of these protocols employs NFTs as an abstraction of votes as well. There are references to NFTs in the protocol description, but these are used to exemplify how the protocols handles ownership of digital objects or using NFT ownership as means to verify the identity of a voter, but never as the main data element carrying the voter's choices.

To finalise this section, we attempted to review any e-voting proposals that were using NFTs explicitly in the voting process. Any usage of NFTs in any capacity in a remote voting system was considered relevant, yet our search was unable to find a single complete proposal that combined both. For this purpose, we consulted the main academic databases, namely *Google Scholar*, *Science Direct*, *IEEE*, and *ACM*, using a broader search term at first, namely, "e-voting" and "NFT," as well as with expanded acronyms and other variations, without much success. The closest article to a NFT-based e-voting system we were able to find was [94], where the authors use SoulBound NFTs, a special case of non-transferable NFTs that can potentially be used for identification purposes

[107], to circumvent the need for a trusted third party to implement *eligibility*, i.e., if a voter is allowed to vote in a given election or not. The proposed system only interacts with these NFTs during the voter validation phase. The article does not provide enough technical details to determine exactly how blockchain is used in the remainder of the voting process, but it is clear with regard to where and when they use these NFTs.

Two other proposals, namely [18] and [1], do mention Non-Fungible Tokens, but more so as a product of their literature and technology review and not as an integral component in their solution. To conclude this search process, [3], [9], and [106] produced extensive surveys around the potentials and usage of NFT, as well as providing a list of future challenges where this technology can be determinant. [3] does mention a potential application of NFTs in governance applications, but without specifying voting or even elections in any capacity. [106] listed challenges limited to purely digital applications, namely gaming, virtual events, digital collectibles, and metaverse applications. [9] provided a broader survey and identified a larger and more specified set of potential applications for NFTs, but none of them related to governance or e-voting.

As far as we were able to determine, no proposals were submitted thus far using NFTs as an integral element of an e-voting system.

3 Introduction to Non-Fungible Tokens (NFT)

Non-Fungible Tokens (NFTs) followed cryptocurrencies as another example of digitally unique constructs, which also contribute to the concept of *digital scarcity*, but with functional differences. As implied in the name, NFTs are not fungible, i.e., unlike Bitcoin or Ethereum cryptocurrency tokens, which are interchangeable and can be transacted as fractions of a unit, every NFT is digitally unique and can only be transferred whole. A user can have only a fraction of a Bitcoin in an account (0.56 BTC for example), but the same is not valid for NFTs. Either a user owns it whole, or not. Just as well, users can exchange any cryptocurrency tokens among themselves

The NFT concept is transversal to all blockchains, but since the concept was introduced through the Ethereum chain, the NFT standard is regulated by two *Ethereum Improvement Proposals (EIP)*, namely EIP-721 and EIP-1155. These proposals define a set of base characteristics (variables and functions) that a smart contract needs to implement to conform to the standard. These requirements are abstracted in the respective *Ethereum Request for Comments (ERC)* standards, ERC-721 [42] and ERC-1155 [41]. It is possible for someone to define an NFT outside of these standards since they are not legally enforceable in any fashion. Yet, since the inception of this concept, the vast majority of published NFts follow this standard since this gives them a level of default interoperability that is hard to achieve otherwise. If a given NFT implements the standards indicated, other users and developers have the guarantee, due to technological requirements that are "forcibly" implemented through the usage of these standards, that the variables and functions defined in the interface are implemented in the NFT contract, similar to what already happens with interfaces in object-oriented programming paradigms.

The application potential of this technology regarding digital collectibles has in itself motivated the creation of NFT-centric blockchains, such as Flow [53], which were developed towards overcoming aspects that make NFT mechanics too expensive, both in gas spent, resources allocated and execution time, in more general purpose blockchains such as Ethereum for example, as well as online marketplaces dedicated solely to the commercialization of NFTs (e.g., OpenSea [88]) as long as the NFT smart contract implements the standards indicated.

Unlike cryptocurrencies, NFTs can store metadata onchain. Yet, because of the high cost associated with writing operations, in most cases, to optimise cost, most of the metadata stored in a NFT is a URL that can be automatically resolved to an off-chain resource, typically an image, a video, or any other type of digital file. This is the most common approach with artistic NFTs and even most digital collectibles [100].

3.1 NFTs vs. Cryptocurrencies

In a blockchain context, NFTs represent objects, while cryptocurrencies are variables. Though most NFTs produced thus far represent **digital** objects, there are no strict requirements in that regard. NFTs can easily be used to represent physical object, or more correctly, to establish ownership of that physical object in a digital distributed ledger, but so far the emphasis has been in keeping everything in the digital realm. The amount of cryptocurrency owned by an account is determined by either a balance value stored in the governing contract, which is the case for the majority of ERC-20 (the standard that regulates cryptocurrency contracts in the Ethereum network) tokens, or by determining the Unspent Transactional Output (UTxO) value associated with the account, which is the strategy used by Bitcoin and a few other cases. Adding data to a blockchain using only cryptocurrencies is a challenge in itself, since these do not provide a direct mechanism to write arbitrary data into a block. Before the NFT standard, researchers looking to use a cryptocurrency-based blockchain for alternative applications had to be creative in that regard. One of the more popular approaches is using transactional metadata to add extra data to the transaction that gets written into a block.

Bitcoin was extensively explored in this regard, since it was the sole blockchain in operation until Ethereum was introduced in 2015. Bitcoin added new functionalities somewhat periodically, and its 0.9.0 version introduced the *OP_RETURN* instruction to its execution set. When run, this instruction always writes its input, unchanged, into the transactional data that gets written into the blockchain [11]. This provided researchers with an alternative to sending arbitrary data to a blockchain that was not conceived with this functionality in mind.

NFTs and smart contracts provide much more efficient and flexible means to achieve the same result. As indicated, NFTs are defined as digital objects, i.e., a pre-defined data structure that can contain several internal parameters, or even other objects if needed. When an NFT gets minted, its metadata is written into a block in the chain. Depending on the actual implementation, this data can be changed afterwards, but typically these operations are restricted to the NFT owner, i.e, changes to the NFT metadata require a digitally signed transaction, which can only be produced by the owner of the account that is currently housing that NFT.

3.2 NFTs in Contract-based and Account-based blockchains

For contract-based blockchains such as Ethereum, NFT metadata is always stored relative to the contract that implements the NFT itself, with information indicating which user "owns" it (typically through an address-to-NFT-identifier mapping). For account-based blockchains such as Flow, NFTs are always and uniquely stored in an account-based location, which can be the minting contract account (address to where the NFT contract was deployed), another contract, or a user account.

In a contract-based blockchain, the information of every NFT minted by a contract is always visible in the deployed contract code, i.e., by checking the deployment address of that contract, which leads to the block where the NFT contract is written, it is possible to check the NFT details or metadata for all NFTs minted by that contract, even the ones that were "burned." This means that if the contract gets destroyed, all NFT ownership information is lost permanently. For example, in Ethereum, Solidity contracts can be programmed with a self-destruct function that ensures this result.

3.2.1 Token Burning

Token ownership in a blockchain is abstracted by the ability of a user to access the account containing that token. This essentially consists of the capability of that user to generate a valid digital signature that can sign a transaction to transfer that token to some other location, which implies the user owning or controlling the private encryption key required to generate this signature. Deriving an account address from a private encryption key is a trivial operation, but the opposite is computationally infeasible. The ownership mechanism of a blockchain is implemented based on this asymmetrical relationship.

Up to the point of this writing, there is no known private encryption key from which it is possible to derive a *zero address*, i.e., an account address composed solely of zeroes (0x00000...), which implies that no one "controls" that address. No one has the private key that can be used to sign transactions to move tokens out of this address to another location. As such, this address has been used historically to "burn" tokens, which consists of transferring a token (NFT or cryptocurrency) to an unrecoverable address. Once a token goes into the *zero address* account, no one can move it out of it due to the lack of a valid private key that can sign the required transaction. Therefore, "burned" tokens are considered as if they were destroyed, when in reality they are simply stored in an account that no one is able to control [6].

In account-based blockchains, NFT metadata can be checked by anyone as long as the NFT remains in the minting contract. If the token is moved into an user account, it becomes inaccessible, unless its owner uses the access control features provided by the blockchain to delegate access to that resource. Section 4.3.1 provided a more detailed explanation on the mechanism used to provide this kind of access control over digital objects. This contract-to-user decoupling also means that destroying the contract does not necessarily mean the destruction of all NFTs minted up to that point. In this particular case, only NFTs still stored in the contract account are destroyed. Tokens already in a user account are safe from destruction.

This approach is interesting and has enough application potential to an e-voting system to warrant a more in-depth exploration. The ability of an NFT to store data directly on the chain while simultaneously maintaining close ownership of the digital object used to modulate this data makes NFTs an interesting candidate for an abstraction of digital voting ballots. This article presents a general idea of an NFT-based e-voting system, followed by two specific implementations based on each approach indicated in Section 3.2 regarding the storage of NFT metadata. Both approaches have advantages and disadvantages that can be analysed to determine which paradigm provides an optimal voting system.

4 General Approach to a NFT-based e-voting system

4.1 General Approach

The e-voting system proposed in this article uses smart contracts to establish an NFT-based framework that uses these tokens as main abstractors of votes, as in data that establishes the choices of a voter in an election in a non ambiguous fashion. We use the features offered by the NFT standards indicated in Section 3 to establish mechanisms for transporting these tokens along the system while simultaneously protecting both the identity and the choices of a voter in a transparent and secure way. Fig. 1 presents a general diagram of what is intended with this proposal. The diagram is agnostic to which type of metadata storage strategy used.

The proposed system is not a fully centralised one, as it can be stated from Fig. 1, mainly due to the intrinsically restrictive nature of elections, which typically establish *a priori* rules to determine who is eligible to vote or not. The ample nature of these rules (age limit, professional background, nationality, membership in an organisation, etc.) requires the existence of a trusted third party to define and enforce them. Our solution not only does not eliminate this element but also assumes its existence as well as incorporating it into parts of the voting process.

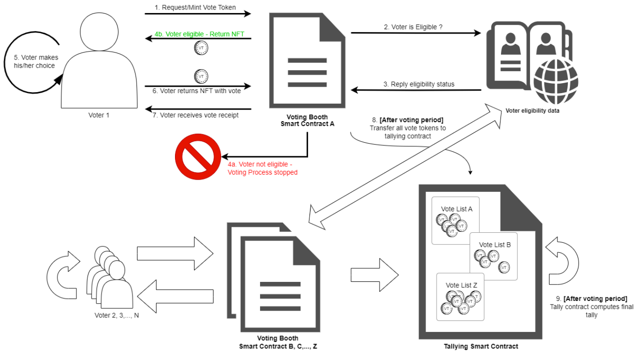
1. Request/Mint Vote Token
4b. Voter eligible - Return NFT
VT
5. Voter makes his/her choice
6. Voter returns NFT with vote
7. Voter receives vote receipt
Voter 1
Voting Booth Smart Contract A
2. Voter is Eligible ?
3. Reply eligibility status
Voter eligibility data
8. [After voting period] Transfer all vote tokens to tallying contract
4a. Voter not eligible - Voting Process stopped
Voter 2, 3,..., N
Voting Booth Smart Contract B, C,..., Z
Vote List A
Vote List B
Vote List Z
Tallying Smart Contract
9. [After voting period] Tally contract computes final tally

Figure 1: General architecture of an NFT based e-voting system.

Voting Process

1. The process begins with a voter interacting with one of the voting booth smart contracts in the available set. Providing multiple points of entry into the system via the creation of multiple but functionally identical voting booth smart contracts increases availability and security by multiplying the effort that an adversary has to endure to prevert the system to his or her favor by a similar factor.

   The interaction is abstracted by calling a public function in the contract that receives the required inputs from the voter. The process starts by infering over the voter's eligibility to participate in the election.

2. The voting booth smart contract requires a valid eligibility status returned before minting a Vote NFT. The list of eligible voters needs to be provided and managed by an external third party that regulates the election, due to the sensitive nature of most elections. The actual implementation of the authentication step is also a research point identified in this solution. For a detailed explanation of the approaches to consider, please refer to Section 5.3.2, where two potential approaches that achieve the same result are detailed. In each alternative considered, the output from an eligibility query is always a boolean value determining if the voter is allowed to continue.

3. The flow of the process splits in this step depending on the response obtained from the previous one:

   a. The voter is not eligible for the current election. In this scenario, the process stops, and the voter is informed of the reason for this interruption.

   b. The voter is cleared for the current election. The voting booth smart contract proceeds with minting a Vote NFT and transferring it to the voter.

4. The voter makes his or her choice by editing the NFT's metadata accordingly. Editing a NFT's metadata is a relatively simple operation, but it still requires a level of technological knowledge that may be out of reach for most people. As such, this step of the process is simplified by abstracting the metadata edition with a contract function. The same function also applies the necessary encryption layers, which are in place to preserve voter privacy as well as to enable the submission of additional votes that replace the previously submitted ones. From a storage point of view, Vote NFTs are stored under the voting booth contracts during the election period, where they can be replaced if the voter changes his or her mind, and move to the tally contract once this period ends. This process also removes the unidirectional link between the voter and his or her submitted NFT, which disables the multiple vote casting feature as well. Please refer to Section 5.2 for additional details on this feature.

5. After submitting his or her choice into the Vote NFT's metadata, the voter returns the token back to the voting boot smart contract. We make no assumptions about the storage location of this token at this point. Any transfer operations should be perceived from a token ownership standpoint.

   The voting booth smart contract stores all Vote NFTs whose encrypted metadata contains the choice of a voter.

6. The submission of a valid vote triggers the return of a receipt to confirm the success of the operation thus far. This receipt is the hash of the transaction used to return the Vote NFT to the voting booth smart contract.

7. The process described thus far has been replicated through several voting booth smart contracts during the election period. Using smart contracts to establish this mechanism also allows for automatic temporal triggers. These are used to divide the election period from the tally period, using a temporal condition to switch the functionalities available in the voting booth contracts.

   During the election period, voting booth contracts accept and validate voter requests, as well as process Vote NFTs for successful voter requests. The end of the election period triggers the start of the tally period. During it, voting booth contracts refuse voter requests by default, either to mint a Vote NFT or to submit a previously minted token, while they transfer all stored tokens thus far to a tally contract.

   With all Vote NFTs are under the control of the tallying smart contract, the counting can begin once the data is ready, i.e., all encryption layers and randomizing elements (salt) are removed from the vote data.

   Once finished, the final tally is retrieved from this contract through a public function. The tally results remain stored in this contract for future reference and auditing purposes.

The general approach described morphs into two different systems depending on the type of data storage paradigm employed by the blockchain used to implement this system. Sections 4.2 and 4.3 detail the two options in greater detail.

4.2 Contract-based Approach

The contract-based specification of the general system is summarized in Fig. 2.

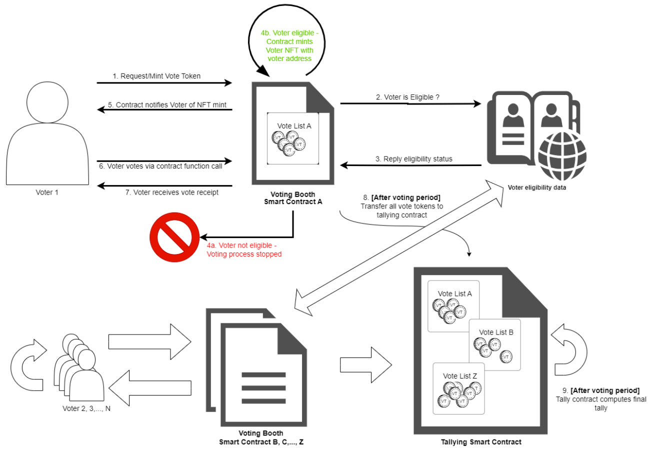
4b. Voter eligible - Contract mints Voter NFT with voter address
1. Request/Mint Vote Token
5. Contract notifies Voter of NFT mint
6. Voter votes via contract function call
7. Voter receives vote receipt
Voter 1
Vote List A
VT
Voting Booth Smart Contract A
2. Voter is Eligible ?
3. Reply eligibility status
Voter eligibility data
8. [After voting period] Transfer all vote tokens to tallying contract
4a. Voter not eligible - Voting process stopped
Voter 2, 3,..., N
Voting Booth Smart Contract B, C,..., Z
Vote List A
Vote List B
Vote List Z
Tallying Smart Contract
9. [After voting period] Tally contract computes final tally

Figure 2: Contract-based version of the e-voting system proposed.

The crucial aspect to take into account with this version is how contract-based blockchains store smart contract-related data. All data related to a smart contract, from token ownership records to token metadata (in the case of NFT-generating smart contracts), is stored "into" the contract, i.e., in a block array referenced from the block where the contract was deployed. The details of how this storage is processed are beyond the scope of this document, but this fact exposes a security issue that requires addressing since contract-based blockchains are quite common and we intend to implement this version of the solution in Ethereum, perhaps the best example of such blockchain.

Considering the visibility of NFT data in this context, additional measures need to be used to protect voter privacy. Encrypting the data using the public key from an asymmetrical encryption key pair is the most obvious solution and one we intend to explore to mitigate this issue. Regardless of the simplicity of the approach, there are several options on how to employ this encryption layer that require careful consideration first. We identified four potential approaches to keeping vote data private in a public, contract-based blockchain, while addressing ownership exchanges of the data objects used to encapsulate votes:

- **Each voting booth smart contract uses a different public key**

from an asymmetric encryption pair to encrypt any sensible information before setting it as the NFT's metadata. Smart contract code is transparent once it is published in the blockchain, so it is not possible for the contract to store the private key from that pair without invalidating the encryption itself.

To maintain a level of decoupling, voting booth smart contracts can only encrypt data while keeping decryption functionalities limited to the tallying contract. This is not a limitation of the technology but a security strategy. The transparency argument also applies to the tally contract, which means that we cannot store any private encryption keys in it without invalidating the encryption scheme. As we have established in Section 4, our solutions assume the existence of a trusted third party, which can be used to store the private pairs of the encryption keys used, under reasonable safety. This encryption problem can be solved by implementing the decryption function in the tally contract only, but in such a way that it requires the private encryption keys to be provided as inputs to the function as well as the data to be decrypted. Fig. 3 exemplifies the process described thus far.

The system is more secure if each voting booth smart contract uses a different encryption key, but this security comes at the cost of increased system complexity. Decrypting the data requires multiple inputs, and a single erroneous byte in the transmission of one of the private keys can invalidate the whole election.

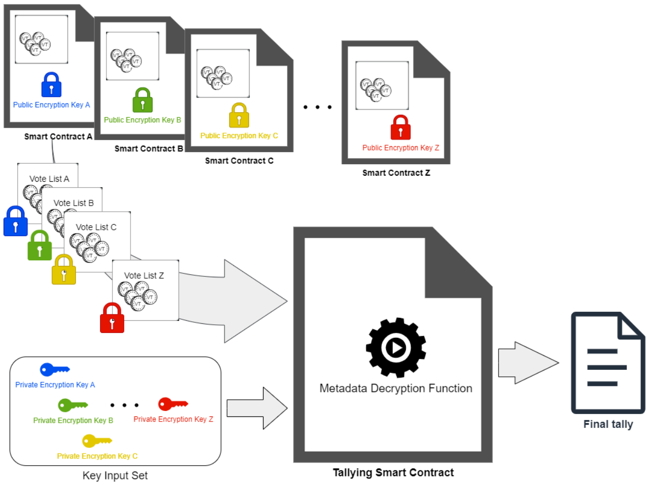

Figure 3: Metadata encryption scheme using multiple encryption key pairs.

- **All voting booth smart contracts use the same public key from an asymmetrical encryption pair to protect NFT metadata.** This approach is functionally similar to the previous one but logistically simpler because all NFT metadata is encrypted with the same key.

In this case, only one encryption key pair gets generated, with the public key freely distributed and used over the whole system and the private key safely guarded by a trusted third party. As with the previous case, voting booth smart contracts can only encrypt data, with the decrypting responsibilities falling solely on the tallying contract and the correct input of the private encryption key from the trusted third party.

In this alternative, the trade-off benefits system complexity (or lack thereof) to the detriment of security. It is logistically simpler to keep a single key secure than multiple ones, but if the key is compromised, the whole system gets affected. Fig. 4 illustrates this approach.

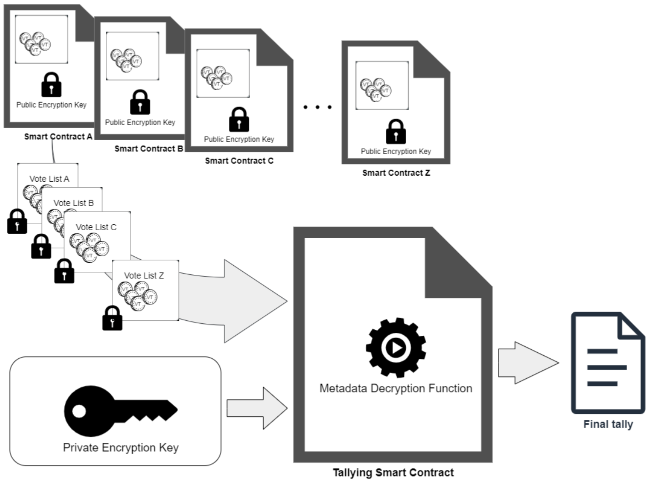

Figure 4: Metadata encryption scheme using a single encryption key pair.

- **Use a threshold cryptosystem and homomorphic properties** to keep vote data encrypted throughout the whole process. If a threshold cryptosystem is used to encrypt the vote data, we can use homomorphic operations directly on the encrypted data to calculate a (still encrypted) final tally, the only element requiring decryption in this scenario [13] [93].

- **Use Adi Shamir's Secret Sharing scheme** [**96**] to split a private key D from an asymmetrical encryption key pair into n pieces. Shamir's secret sharing scheme requires that only k of these pieces be able to reconstruct D. This scheme also requires the use of a *(k, n)* threshold cryptosystem, where $n = 2k - 1$. With such a scheme, an adversary needs to obtain at least k pieces to be able to subvert the system while being able to use a single encryption key for all. As long as an adversary is unable to obtain as many as $k - 1$ pieces, reconstructing D is still computationally infeasible. This alternative, though more complex than any of the encryption-decryption schemes considered to this point, allows for the best of both worlds in this regard: an adversary needs to subvert multiple elements in the system to be able to corrupt the system while still using a single key to encrypt all data in the system. This scheme effectively combines the advantages of the first two discussed with a minimal increment in solution complexity.

4.3 Account-based Approach

Fig. 5 displays the proposal system under an account-based blockchain environment.

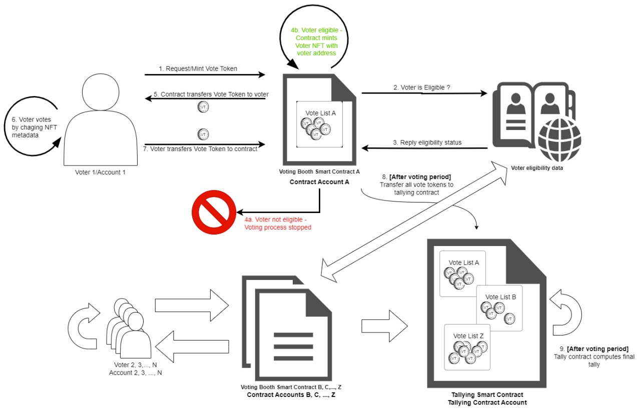
4b. Voter eligible - Contract mints Voter NFT with voter address
1. Request/Mint Vote Token
5. Contract transfers Vote Token to voter
6. Voter votes by chaging NFT metadata
7. Voter transfers Vote Token to contract
Voter 1/Account 1
Vote List A
VT
Voting Booth Smart Contract A
Contract Account A
2. Voter is Eligible ?
3. Reply eligibility status
Voter eligibility data
8. [After voting period] Transfer all vote tokens to tallying contract
4a. Voter not eligible - Voting process stopped
Voter 2, 3,..., N
Account 2, 3, ..., N
Voting Booth Smart Contract B, C,..., Z
Contract Accounts B, C, ..., Z
Vote List A
Vote List B
Vote List Z
Tallying Smart Contract
Tallying Contract Account
9. [After voting period] Tally contract computes final tally

Figure 5: Account-based version of the e-voting system proposed.

The main difference with this approach is that, in this case, the Vote NFT is always stored under the account of the element that holds the token at a given point in the process, i.e., the Vote NFT is actually moved through distinct storage locations throughout the process, which are abstracted and referenced as accounts, as opposed to the contract-based approach, where the only changes are to the internal mappings of the NFT-generating contract, typically indicating which address controls which NFT. From a voter's perspective, he or she should not detect any changes in the voting experience.

4.3.1 Account-based Blockchain to use

For this case, we considered the Flow blockchain for two main reasons:

1. Flow [53] was developed by the creators of CryptoKitties [45], one of the first examples of an NFT-generating smart contract in the Ethereum network. Besides establishing one of the first real-world examples of digital collectibles, the CryptoKitties smart contract established a series of game-like characteristics designed to attract users to experiment with the concept.

The success of this initiative exposed the potential behind NFT technology as well as how limited and difficult it really was to scale the Ethereum blockchain. The network experienced difficulties in responding to an unusually high volume of transactions due to the rapid popularity of CryptoKitties.

Flow was developed as a scalable, resource-based blockchain. Like many public blockchains developed in recent years, it also developed Cadence [98], an interpretative programming language used to create smart contracts, as well as expanding the functionalities of this blockchain through automated Cadence-written scripts and transactions.

2. Flow is a good example of an account-based blockchain. Given the resource-oriented nature of this blockchain as well as its claims of scalability, there was some emphasis on data storage when planning this blockchain, which is very useful to the system we intend to implement.

Account model in Flow Flow blockchain implements accounts as a record of a specific state of the blockchain defined by:

- An unique address consisting of an 8 byte or 16 hex-encoded character string that can optionally be preceded by **0x** to indicate the format used (optional).
- A set of public keys that are required for any interaction with that account, as well as to verify the validity of digital signatures. Unlike Ethereum and other blockchains, Flow supports multiple public keys in a single account, which enables multi-user-controlled accounts. Keys in the public set can be created with different weights to allow multi-signature schemes that also allow for non-homogeneous distribution of account control [66].
- Smart contract code, which is saved in a specific dedicated area of the storage space. A single account can store multiple contracts.
- The main storage where all digital objects (resources) are stored.

Fig. 6 illustrates this organization.

An important fact to retain from Fig. 6 is that contracts in this model are stored in a distinct area from general-purpose storage and, as such, have different access rules. Like with any other smart contract-capable blockchain, smart contracts in Flow can be read by anyone, even though they are stored in an account-specific storage area.

How storage works on Flow Flow stores data as digital objects, which are referenced as resources. This aspect is central to the functionality of this blockchain, so much so that its developers define the programming paradigm imposed by Cadence as a "resource-oriented paradigm" [66].

Storage in the Flow blockchain is account-based, which means that all digital objects permanently stored in this blockchain are referenced by the controlling

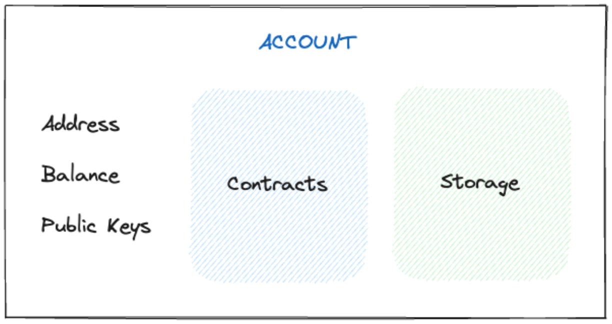

Figure 6: Overview of the account model used in the Flow blockchain. Source: [66]

account. The internal mechanics of the Flow blockchain ensure that only the owner of an account can alter its storage, namely, read, write, and manipulate the access control (permissions) associated with data objects in storage. Other than contracts, Cadence is used to write executable scripts and transactions. Functionally, these are similar, but with one exception: modifications to the state of the blockchain (add or remove resources, token transfers, etc.) can only be performed through transactions, and these, as is the rule in all blockchains, need to be digitally signed by the account owner. Scripts are "lighter" versions of transactions that do not require a digital signature to be executed and are restricted to reading data only. This guarantees that only the owner of the account is able to manipulate the data in it.

Similar to all other blockchains, storage space in Flow is not free, though it is arguably cheaper than in other popular blockchains such as Ethereum. The amount of storage allocated to an account is proportional to the balance of the FLOW token (Flow's native cryptocurrency, from here referred to in all capitals to distinguish it from the main blockchain) of the account in question, establishing a rate of 100MB of storage space per FLOW token, i.e., a user with a balance of 1 FLOW in his/her account can store up to 100MB of data in storage [66]. If an operation (transaction or smart contract function) stores data above the allowed capacity of an account based on the current balance, the transaction fails and all state changes are reverted, just like with a regular smart contract exception.

Flow also establishes a base value of 100 kB of storage space that remains available in perpetuity. This establishes two consequences:

1. A user needs to pay to create an account. The "permanent" storage costs 0.001 FLOW, according to the rate mentioned above, and has to be paid during account creation, though these tokens actually end up in the main account balance.

2. Flow accounts require a minimum balance of 0.001 FLOW at all times

to pay for this permanent storage, therefore Flow accounts cannot be completely emptied [66].

Storage domains Each account in Flow has three domains that define its storage space: */storage*, */public*, and */private* [66]. These are indicated as UNIX-style paths to reinforce the notion that they are mainly used to indicate where a digital object is stored in the storage space allocated to a specific account. This account storage is only used to save digital objects as well as references (akin to memory pointers. Please refer to Section 4.3.1 for additional details), used to delegate read access to stored objects without risking their modification or deletion from other users and/or smart contracts.

All resources stored in an account's storage go into the */storage* domain by default. Only the account owner(s) have access to this domain, for both write and read privileges. This means that any resource transferred into an account, either through a transaction or a functionally equivalent smart contract function, becomes private by default as soon as it is stored. To allow for external access, the controlling user must move that resource to either the */public* or */private* domains. Data in the */public* domain is available to everyone. The */private* domain, despite the name, actually sits somewhere between the */storage* and */public* domains regarding access. Data stored in this domain is only available to the owner and to users previously authorised by the owner.

Capabilities *Capabilities* are tools used in the Flow blockchain to delegate access to resources (data objects). An account controller creates *capabilities*, which are akin to a memory pointer in that instead of referencing a memory position, they reference a specific resource stored in the account where the *capability* was created. *Capability* creation changes the state of the blockchain; therefore, it needs to be done in a transaction digitally signed by the account owner to be executed. These requisites restrict the creation of *capabilities* to the owner of the resource in question.

Flow distributes *capabilities* among account holders using a messaging system. Once created, the account owner publishes the *capability* to an account address to delegate control of the target resource. The *capability* is delivered to the inbox of the address indicated, to be retrieved at a later point by the target account owner.

Interfaces, a recurring concept in both general-purpose programming languages, Solidity and Cadence, are used to restrict the visibility of a resource. Visibility in this context means the internal parameters, events, and functions that remain available once access to the resource is delegated through a *capability*. Cadence uses interfaces to define mandatory standards for classes, similarly as in Solidity, but with an additional layer of access control for future instances of that class in storage.

5 Security Analysis and Additional Features

5.1 Data Encryption

The solution proposed is based on publicizing data in a publicly accessible ledger. In order to achieve and maintain voter privacy, data published in this medium needs, at least, a layer of encryption needs to be applied to it before entering the public structure. This encryption must provide two functionalities: prevent other users from determining the voting options of a specific voter or voters, and prevent the calculation of the final tally before the intended process step.

Asymmetrical encryption schemes are an obvious, simple and efficient solution for this problem. Our proposed system maintains three essential actors: voters, voting booth contracts, and the tally contract. Each actor can create a pair of asymmetrical keys and distribute the public key, but for our specific case, this task is limited to the smart contracts, i.e., voting both and tally. This removes some complexity from the voter perspective and deposits it into programable software scripts, which allows for an automation of this process. As such, each voting booth contract and the tally contract generate a pair of asymmetrical encryption keys, with the public key in each one provided to a voter alongside with the Vote NFT used to contain their election choices.

Section 5.2 presents a more detailed exposition on how the encryption scheme is implemented in this protocol.

5.1.1 Data Obfuscation

Vote data is double encrypted with the tally contract key and the respective voting booth contract key, i.e, the public encryption key provided by the voting booth contract that mints the Vote NFT requested by a voter. A double encryption scheme using a pair of keys from a larger set increases the solution space for the resulting ciphertexts, which makes a statistical analysis of the encrypted votes harder, but not completely impossible. In any election, the universe of choices for a given question is finite. If no additional obfuscation methods are employed, especially if any encryption efforts are performed with a small number of encryption keys, the solution space for generated ciphertexts may be small enough to warrant the use of statistical tools that can provide relevant information about the final result before the end of the election.

For example, consider an election that requires a voter to chose one of four choices: *A, B, C, or D*, a simple encryption scheme that advances the selected option ten positions in the alphabet, and with only one encryption layer, it is easy to understand that the solution space is simply *K, L, M, N.* Determining the final tally without decrypting the vote data becomes a trivial operation. Adding "salt", i.e., a piece of random data, to the plaintext before encrypting it solves this issue because it extends the solution space so that statistical analysis of the encrypted data becomes infeasible.

The value of salt in itself is irrelevant since it gets discarded after decrypting the "salted" data. The difficulty of obtaining true random values in a blockchain, which is purely deterministic, is well known [6], but for our purpose, a pseudo-random number is sufficient since we are not using it to generate critical system information, such as encryption keys or other cryptographic element. Using portions of a hash digest is a popular option to obtain pseudo-random values in a simple and fast fashion. Popular hash functions, such as the ones in the *keccak* family used in the Ethereum blockchain, can be applied to another pseudo-random piece of data, such as the current UNIX timestamp for example, to obtain these salt values.

Salt values are appended to the vote information before encrypting it with the public key from the tally contract (inner encryption layer).

5.1.2 Private Key Management

The universe of key pairs is equal to the number of smart contracts used to support the election,which are all the voting booth smart contracts plus one, the tally one. The transparent nature of these contracts prevents any sensitive information from being published in their code, namely, writing the private keys in the corresponding contracts defeats the whole encryption effort since anyone can check the contract source code and decrypt data stored under it. We solve this problem by relying on the thrusted third party that we are already using to determine voter eligibility.

5.2 Multiple Vote-Casting

Multiple vote-casting is a feature used to combat vote coercion by a coercer, here defined as someone that can influence how a voter casts his or her choice either through positive, (money offers, gifts, etc) or negative (blackmail, threats of violence, etc.) reinforcement, by allowing voters to cast multiple votes in an election and counting only a single one. Vote coercion is an undesirable behaviour for any voting system, electronic or not, since it defeats the basic purpose of the exercise. This approach was implemented in e-voting solutions such as the Estonian i-Voting system [72], the Norwegian e-Vote project [38], and even TiVi, a blockchain-based e-voting solution [99]. In all the examples indicated, a voter can submit multiple votes electronically during the election window, and only the last one gets counted. The Estonian i-Voting system, which was deployed in Estonia's 2005 national elections in parallel with a traditional paper ballot system, prioritised the physical vote over any electronic one as well.

The idea behind this approach is to remove incentives from coercers. Even if they are able to "convince" someone to vote against their will, voters always have the opportunity to replace the coerced vote later on or, in the Estonian case, vote physically if they are unable to do so freely from an electronic platform.

We intend to implement a similar, NFT-adapted feature in our proposed solution. The increment in solution complexity is negligible since we already benefit from the fact that we operate with digitally unique NFTs as vote abstractors. In our approach we considered a strategy based on the replacement of submitted Vote NFTs, i.e., once submitted, the metadata of a Vote NFT cannot be altered, but a submitted token can be replaced by another cast at a later stage, as long as it happens within the voting window defined.

To implement such feature, we need to ensure the storage strategy used allows for future replacements of submitted tokens while, simultaneously, ensuring voter and vote privacy, i.e., that any voter information, as well as their choices in the current election, are protected. To ensure privacy along the whole process, multiple layers of encryption are applied to the sensible data, namely, voter specific information and ballot choices. Data encryption is performed at the smart contract level, i.e., invoking contract functions. Public encryption keys for this purpose are shared within the contract code.

Vote NFTs are stored at the contract level using an hash map, a popular storage strategy in distributed environments. The hash value used to index Vote NFTs is calculated before the submission of the Vote NFT, at the user level, where his or her personal information is available and can be prepared such that it produces a unique indexing hash. The usage of already unique identification numbers in this process, such as a National ID number, a Tax Revenue Service ID, Social Security Numbers, etc., ensures this outcome.

The full process flow for this feature is represented in Fig. 7.

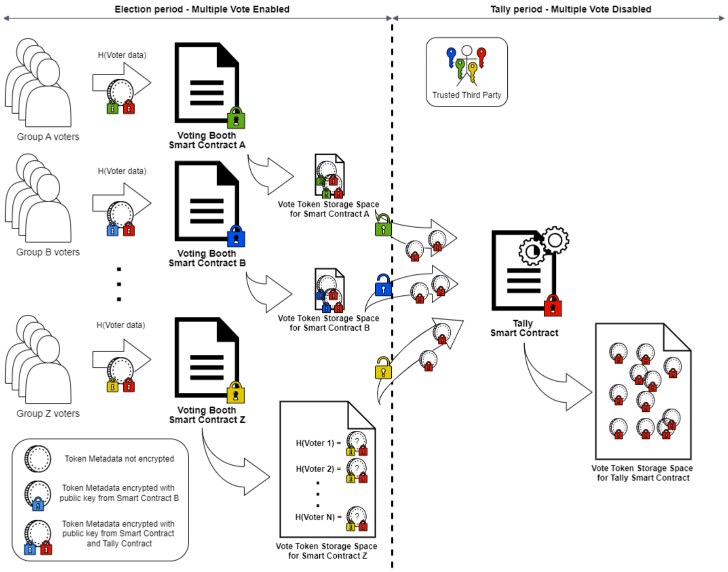
Election period - Multiple Vote Enabled
Tally period - Multiple Vote Disabled
Group A voters
H(Voter data)
Voting Booth Smart Contract A
Group B voters
H(Voter data)
Voting Booth Smart Contract B
Group Z voters
H(Voter data)
Voting Booth Smart Contract Z
Vote Token Storage Space for Smart Contract A
Vote Token Storage Space for Smart Contract B
H(Voter 1) =
H(Voter 2) =
H(Voter N) =
Vote Token Storage Space for Smart Contract Z
Token Metadata not encrypted
Token Metadata encrypted with public key from Smart Contract B
Token Metadata encrypted with public key from Smart Contract and Tally Contract
Trusted Third Party
Tally Smart Contract
Vote Token Storage Space for Tally Smart Contract

Figure 7: Implementation logic to allow for multiple vote casting.

The submission of a Vote NFT also includes a hash digest that is unique for all voters and can be used to retrieve that Vote NFT cast previously by a voter without revealing its contents. During the election period, successfully submitted Vote NFTs are stored under the respective voting booth smart contract. The actual storage location for these tokens is dependent of the blockchain architecture considered but it is not critical for this explanation.

Privacy is assured via double encryption of the vote data by a pair of keys supplied from the voting booth contract and tally contract. The private encryption keys required to decrypt all the voting data have to be securely stored during the whole exercise. A sensible and sensible approach is to entrust them to the same trusted third party that ensures the integrity of the list of eligible voters, but other options may be of consideration.

During the election window, users can request and submit multiple Vote NFTs, each replacing any one previously submitted. Once this window terminates, the tenuous link between the tokens and their owners is broken upon sending them to the tally contract. This operation also removes the layer of encryption that could be used to divide the tokens per voting booth, thus achieving anonymity between the voters and their choices. Vote NFT tokens are stored under the tally contract under no specific order.

The final tally is determined by removing the last layer of encryption from the tokens metadata and computing the final result. Another potential, more private approach, consists in computing the tally homomorphically and decrypt only the final result. This approach extends the privacy related to the contents of the actual votes, but given that all links to the voters who cast them are already removed, the gains with such approach are negligible considering the added complexity and resource consumption.

5.3 Security Analysis and Assessment of Threats

The security of e-voting systems has been the subject of academic research, and it has coalesced around the definition of security criteria, such as voter privacy, transparency, accuracy, etc. These criteria are met through the implementation of, typically, cryptographic features that increase the security of the system against dishonest actors. A simple example is using encryption to protect the information in a vote to achieve vote privacy. The number and type of criteria used for this characterization differ from publication to publication, but some attempts have been made over the years to normalise this exercise. In one of the latest publications dedicated to this effort, [4] presented a literature survey that established a series of five criteria deemed "minimal" as the base set from an analysis of a number of publications that employed such characterization to determine the security offered by the proposed solution.

Alongside with this analysis, we also include an assessment of potential threats included in each criteria considered, namely, concrete attacks that have been employed on previous e-voting systems and how the proposed system is able to protect itself and/or nullify them. [112] includes a detailed list of potential threats affecting e-voting system without differentiating between any of the architectural paradigms considered. We adapted this list to our architecture and repeated the threat analysis from the reference point of a decentralised e-voting system.

The following analysis is going to focus on the following security criteria:

1. Accuracy - A voting system is accurate if it does not allow for changes to a vote after submission, changes to the final tally, or invalid votes.
2. Eligibility - Voting systems that allow only registered or valid voters to submit votes implement eligibility.
3. Privacy - A private voting system is able to remove all the links between personal information related to the voter and the vote submitted.
4. Verifiability - A voting system is considered verifiable if it allows for independent confirmation of the submission of a vote by the voter.
5. Robustness - A robust voting system is able to prevent and even withstand the actions of dishonest voters to prevent the successful and accurate completion of the voting exercise.

The remainder of this section analyses the e-voting solution detailed in this proposal according to the framework for minimal security criteria presented in [4].

5.3.1 Accuracy

Encoding votes into NFT metadata ensures that this information is written into the blockchain. In both approaches considered, once the NFT is written into a block and is part of the chain, the contents of a vote become immutable due to the computational infeasibility of modifying data on the chain.

The final tally is the output from a smart contract function, which adds transparency to the process but does not negate the possibility of an erroneous result or detect any invalid votes. A smart contract function can be used to ensure the correct format of a vote before counting it. Assuming a deterministic computation of the final tally, this means that changes in it imply unauthorised changes to the number of Vote NFTs submitted. An expansion of the previous reasoning negates this scenario as well, since such a change to the number of votes submitted, as well as the number of Vote NFTs stored, implies adding and/or removing information from the blockchain, which we have already stated is computationally infeasible.

Threat Assessment The accuracy of an e-voting system can be affected by adversarial activities such as:

- Wiretapping
- DNS attack
- Malicious code on client
- Hardware modification, substitution and interception
- Software modification, deletion, edition, trojan horses, information leaks, trapdoors and viruses

Basing our solution on a blockchain immediately nullifies many of the threats considered above. The immutable and transparent nature of blockchain make threats based on an adversary being able to alter the system's software and/or hardware irrelevant, as well as any threats from wiretapping communication channels. Software is deployed mainly as smart contracts, which prevents unwanted and undetectable changes to the code used for the core properties of the system. Blockchains also notoriously abstract most hardware characteristics from its overall workings, so, as long as nodes execute protocol code as intended, any hardware alterations do not pose a threat.

The most serious threat from this list is the existence of malicious code on a client. The core of the system runs on a distributed, public network, but the submission of individual votes requires an interface that is served from the client side. Realistically, an adversary can replace this interface with a different one without the user noticing it. The risk here is the loss of private information, such as private encryption keys and personal information. If an adversary is able to, for example, control the interface used by someone else to submit votes, such as controlling an unlocked and previously authenticated device (theft, blackmail, threats of violence, etc.), installing remote control software in a device without the owner's permission, etc., he or she can submit a vote under another person's identity. It is very hard to protect the system against such deep level of control. To counter this threat, we provide a multiple-vote casting feature, but in the end, it is the voter that needs to detect an erroneous submission from his or her platform, which is eased by providing voting receipts and an history of interactions with the smart contracts that process the submissions, and provide a replacement vote according to the voter's wishes, as well as reporting this attempt to the relevant authorities. As indicated above, the actions that an adversary has to undertake to be able to cast a vote as someone else can be considered criminal offenses with penalties foreseen in most legislative frameworks.

5.3.2 Eligibility

The sensible nature of elections prevents the full decentralisation of an e-voting system, at least regarding the presence of a trusted authority that determines the individual eligibility of each voter based on whatever rules exist to regulate a given election (e.g., age, nationality, professional status, etc.). Unless an election is free, i.e., without any regulations regarding who can vote, there is a requirement for a mechanism to implement limitations on the set of voters that are eligible to participate in it. For example, it makes sense to exclude teachers and other non-student school workers from an election used to select a representative from the student body.

For our proposal, we consider two approaches to solving this problem in a decentralised fashion:

- using a centralised database to store eligibility information and access it from the decentralised e-voting system using an oracle. Any oracle used for this effect replies to a query in boolean format, i.e., if a voter is (*true*) or is not (*false*) eligible to participate in the election based on the data returned from the centralised database.

- using a cryptographic accumulator, used to prove membership. Instead of storing all the eligible voter data, it can be "accumulated" instead into a root value. We can use the accumulator's properties to infer if a given piece of data related to a specific voter is present in the root value or not, thus also obtaining a boolean reply in this regard, like with the previous method.

Another possible approach is the usage of SoulBound NFTs for identification purposes, a strategy used in [94]. But unfortunately, these NFTs are still in their infancy regarding their applicability in other areas, and this approach is omitted from the presented solution. Sections 5.3.2 and 5.3.2 provide additional details regarding the two approaches considered in determining voter eligibility in our solution.

Oracle Accessible Centralised Eligibility Database The basis of our eligibility problem derives from the limitations of current blockchain implementation in storing large quantities of data. Though it is technically possible to do so, storing even a small list of eligible voters directly on the chain is very inefficient, both from an economic (due to potential gas costs) and resource management perspective. This issue is actually pertinent across several application domains, and, as a consequence, several strategies to allow blockchain to access external data, i.e., off-chain data, have been proposed.

[39] presents an exhaustive study on these strategies, which are used to establish communication protocols to enable the exchange of data with external data sources or even other blockchains. Among these, oracles stand out as one of the most promising in terms of flexibility and ease of implementation. Blockchain oracles are implemented as smart contracts that use their functions to query data from the outside, namely by requesting it from an external source, similarly to a regular API. Due to this perception, oracles are often seen as "blockchain middleware," decreasing the gap between blockchains and the external world. A simple example of oracle use can be found in crypto exchange platforms that allow for trade between different cryptocurrencies. The value of each tradeable cryptocurrency is currently very volatile and often dependent on speculative factors. Since blockchains themselves do not store the exchange rate of their native token relative to a fiat currency, which is the parameter to consider in inter-blockchain trades, this information is kept instead in "traditional" centralised sources that maintain APIs that can be used to retrieve this information remotely. Oracles are often used in these cases to retrieve the exchange rate of the tokens involved (external information) and provide a fairer exchange of tokens (internal operation).

Oracles fit quite well in our solution. Eligibility data is stored in a conventional, centralised database, and the eligibility state of a given voter is obtained by invoking a function in the smart contract that implements the oracle. Oracles are relatively easy to construct, but they do increase the centrality of the solution while potentially creating a point of failure in the system. External data sources are not subjected to the blockchain protocol and data redundancy that ensure its integrity, and thus can be corrupted with less effort. Therefore, using an oracle in this fashion also requires a trusted third party to ensure the integrity of the external data, which moves this solution away from a purely decentralised one.

Cryptographic Accumulators Cryptographic accumulators, also known as set accumulators, are cryptographic primitives that use unidirectional hash functions to represent a set of elements through an *accumulator value*, a single and constant-size value that can be used to determine if a given element belongs to the set without revealing the set contents at any point of the verification process. A hash function $y = H(x)$ is such that it receives an input x with arbitrary length to produce a fixed-sized output y, also known as *hash digest.* The unidirectionality of the hash function derives from the fact that:

1. For all $x \in \{0,1\}^*$ it is computationally easy to calculate $y = H(x)$.
2. For all $y \in \{0,1\}^n$ it is computationally infeasible to find $x \in \{0,1\}^*$ such that $H(x) = y$.

A desirable hash function is also one that provides collision resistance, i.e., it is computationally infeasible to find two input elements, $x_1, x_2 \in \{0,1\}^*$, such that $H(x_1) = H(x_2)$, which enables hash digests to be used as data fingerprints, a strategy commonly used to ensure data integrity during communications, software certification during distribution, etc. Among the various use cases of cryptographic accumulators, we highlight *anonymity enhancement* and *identity management* as the most pertinent for our case. Set accumulators can be static or dynamic, depending on their ability to add and remove elements from the original set without requiring the re-calculation of the *cumulative set* [71]. For our particular context, static accumulators are preferable. By publishing the *cumulative value* of the set of eligible voters using a static accumulator, this means that the list of eligible voters "is closed," meaning that, for better or worse, no voters can be added or removed after this publication without changing the *cumulative value* published.

[14] introduced set accumulators as, in a simplistic fashion, the hash digest of the cumulative value of the hash digests of the elements of the set. To determine the *cumulative value* of a set accumulator, one begins by determining the hash digest of every individual element. From there, hash digest pairs are concatenated and re-hashed to obtain a digest of the hashed pair. This process continues until one value remains: the *cumulative value.* Fig. 8 illustrates this process.

An independent verifier can verify the membership of an element by obtaining a minimum of partial results and recalculating the *cumulative value*, starting

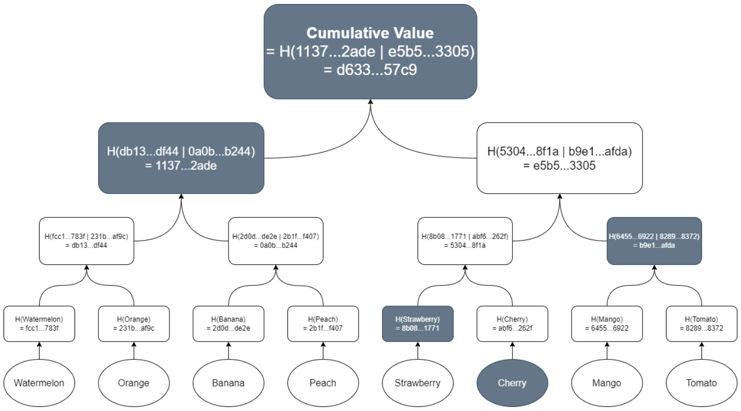

Figure 8: Simple example of an accumulator to represent a set of strings.

with the element whose membership he or she desires to verify and combining the successive hashes with the ones provided. If the *cumulative value* obtained matches the one published, the unidirectionality and collision resistance of the hash function used guarantee, with reasonable certainty, that the element does belong to the original set. Taking the example presented in Fig. 8 into consideration, if an independent verifier wishes to determine if "Cherry" belongs to the original set, he/she only requires knowledge of the *cumulative value* and the partial calculations indicated in grey in the figure. From there, the verifier can use the hash function used to determine this value to re-calculate the *cumulative* one without requiring any knowledge about the remaining elements of the original set or needing to re-hash every element. Providing only the intermediary hashes to the independent verifier allows for the remaining set elements to remain secret since it is computationally impossible to determine which strings originated from the hash digests provided.

Cryptographic accumulators present a solution that benefits from the advantages of storing relevant data directly in the blockchain while keeping that data to a minimum of the required storage space [104]. This approach is not novel: in [28], the authors use accumulators to optimise a UTXO (*Unspent Transaction Output*)-based blockchain by storing the *accumulated value* of the UTXO set in the blockchain instead of the whole set. [105] apply cryptographic accumulators to authenticate users in an *IoT (Internet Of Things)*-based blockchain, similarly to our proposed strategy.

Our approach is to use a set accumulator to obtain the *cumulative value* of all eligible voters, and then use membership functions to determine if a given voter is eligible for the exercise based on the output from the membership function. This eliminates the need to access external data sources while keeping the additional data stored in the chain to a minimum.

Threat Assessment

- Impersonation
- Falsification of messages
- Spoofing
- Man-in-the-middle Replay attacks

5.3.3 Privacy

The way voter privacy is achieved in our system depends on the approach considered, given how important the storage mechanism is to the criteria. But both approaches are able to achieve it.

In a contract-based approach, the link between Vote NFT data and the entity that created it is maintained solely in the data structure that maps a specific, unique Vote NFT to an address. The privacy of a voter depends on how hard it is to determine the identity behind these addresses, which is an issue transversal to all blockchains. It is possible to use statistical tools that analyse blockchain transactions to increase the probability of discovering who is controlling an address, but these often rely on the existence of a significant number of transactions with that address. If a voter uses a specific address solely for voting purposes, this threat diminishes greatly. Additionally, in this approach, vote metadata is always encrypted when the NFT gets stored in the contract. So even if the identity of the voter behind an address gets revealed somehow, the adversary would still have to decrypt the metadata somehow to have any impact on the voting exercise. The pseudo-anonymity of blockchain operations plus the added layer of encryption to the sensible data ensure voter privacy in this system.

Account-based blockchains simplify this feature by assigning private storage domains that ensure that any sensitive operations, namely the voting act itself, occur in a personal space protected cryptographically. To gain access to such space, an adversary needs to discover the private counterpart of the asymmetrical encryption key pair used to set up the account, i.e., by breaking an encryption cypher. If the strength of the keys used is sufficiently high, this task should be computationally infeasible, thus also achieving voter privacy within the account-based approach.

5.3.4 Verifiability

Implementing this solution in a public blockchain allows us to benefit from transparency regarding the verification of data once it gets written into a block, which in our particular case is in the form of NFT metadata. As with any other NFT published thus far, their contents are open to consultation for anyone, namely, the image, video or audio clip, text, etc. that got encoded into that NFT. Unfortunately, this level of transparency also defeats any attempts to keep voter data private, thus negating the *Privacy* feature described in Section 5.3.3. But as it was also indicated in this section, any sensible data written into the public blockchain (regardless of the storage approach considered) needs to be protected to implement even the weakest form of voter privacy. This opens up two interesting and practical approaches to implementing vote verifiability:

1. If a vote is published after being encrypted using public encryption keys whose private counterparts are being carefully controlled by a trusted third party, as suggested in Section 4.2, a voter can verify the correct submission of his or her vote by comparing the ciphertext in the metadata for the Vote NFT associated with his or her address (this is valid for both contract and account-based storage approaches) to the result of a re-encryption of his or her original vote. The voter is unable to decrypt and verify the data directly in the Vote NFT metadata because that would require knowledge of the corresponding private encryption keys, which are kept secure by a trusted third party, but he or she can easily replicate the encrypted data and verify that it matches the one written in the chain. But this approach also requires the use of "salt," as in a random integer inserted into the vote data and known only to the voter, to prevent an adversary from executing the same exercise. This strategy mimics others used to establish secure communications over insecure channels, as detailed in [78] and [77].

2. A simpler and more practical alternative is to "fingerprint" vote data with a hash digest, using all data present in the Vote NFT metadata, for example, and including it as another parameter in the Vote NFT. Like the previous approach, voters can ensure that their vote remains unchanged after submission by re-creating the vote and re-hashing it. If the digest obtained this way matches the one written on chain via the Vote NFT's metadata, the encoded vote is exactly the same as the one just reproduced. This also opens the possibility of an adversary using this method to break the privacy of voters. If an adversary knows the format used and sufficient personal data from a voter, he or she can determine the voter's choices by obtaining the hash digest written in the submitted Vote NFT metadata, by brute force if needed. As such, the use of "salt" or any other parameter exclusive to the knowledge of the voter needs to be used to ensure vote verifiability without compromising vote privacy.

5.3.5 Robustness

A system based on a blockchain benefits directly from the security derived from the cryptographic methods used to establish basic data integrity in that chain. The effort required from an adversary to change the value of a vote (as metadata from an NFT) or the final tally is the same as manipulating a block of transactions to attempt double-spending or any operation that tries to change data into a block already in a chain: computationally infeasible.

A potential source of corruption can be envisioned if an external database is used to determine eligibility (through an oracle, for example). Any off-chain elements are, by definition, weak points of a decentralised system since they represent an exception in the paradigm. This threat can be mitigated by the use of hash digests of the list of voters and their data, published once a valid and final list is achieved, and verified periodically to detect unwanted modifications. .

6 Conclusion

Electronic voting systems are still an active area of research and a playground from which some important cryptographic concepts have been derived. It has evolved around the technology that supports it, which has transitioned this research area to a new age regarding the fundamental approach considered, thanks to its recent integration with blockchain technology.

This proposal presents a decentralised, remote electronic voting system that takes advantage of the properties of Non-Fungible Tokens, as well as the Flow blockchain and its Resource-Oriented Paradigm. The organisation and features offered by this new approach support the design of a private, secure, transparent, verifiable, and mobile voting system whose proof of concept is to be derived from the strategy presented in this text.

This also proves the suitability of the NFT concept as a transport mechanism within public networks.

This paper defines a concrete road map to produce a working prototype to prove the implementation of the set of security criteria in Section 2, as a strategy to achieve a truly secure and remote electronic voting system. Another important aspect that determines the usability of this proposal is scalability; specifically, what are the limits of the supporting blockchain regarding the scope of operation of this system.

References

[1] Samuel Agbesi and George Asante. “Electronic Voting Recording System Based on Blockchain Technology”. In: *Proceedings of the 12th CMI Conference on Cybersecurity and Privacy*. Copenhagen, Denmark, 2019. ISBN: 978-1-72812-856-6. DOI: 10.1109/CMI48017.2019.8962142.

[2] Agora. *Agora, Bringing our voting systems into the 21st century*. Tech. rep. www.agora.vote, 2021, p. 41. URL: https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5f37eed8cedac41642edb534/1597501378925/Agora_Whitepaper.pdf.

[3] Omar Ali et al. “A Review of the Key Challenges of Non-Fungible Tokens”. In: *Technological Forecasting and Social Change* 187 (2023), pp. 1–13. ISSN: 00401625. DOI: 10.1016/j.techfore.2022.122248.

[4] Ricardo Lopes Almeida et al. “Impact of Decentralization on Electronic Voting Systems: A Systemic Literature Survey”. In: *IEEE Access* 11 (November 2023), pp. 132389–132423. ISSN: 21693536. DOI: 10.1109/ACCESS.2023.3336593.

[5] Syada Tasmia Alvi et al. “DVTChain: A Blockchain-based decentralized mechanism”. In: *Journal of King Saud University - Computer and Information Sciences* 34 (9 2022), pp. 6855–6871. ISSN: 22131248. DOI: 10.1016/j.jksuci.2022.06.014.

[6] Andreas M. Antonopoulos and Gavin Wood. *Mastering Ethereum, Building Smart Contracts and Dapps*. 1nd. OŔeilly, 2018. ISBN: 978-1-491-97194-9.

[7] Ahmed Ben Ayed. “A Conceptual Secure Blockchain-based Electronic Voting System”. In: *International Journal of Network Security & Its Applications (IJNSA)* 9 (3 May 2017), pp. 1–9. ISSN: 09752307.

[8] Fabrizio Baiardi et al. “SEAS, a secure e-voting protocol: Design and implementation”. In: *Computers & Security* 24 (8), pp. 642–652. ISSN: 01674048. DOI: 10.1016/j.cose.2005.07.008.

[9] Hong Bao and David Roubaud. “Non-Fungible Tokens: A Systematic Review and Research Agenda”. In: *Journal of Risk and Financial Management* 15 (5 May 8, 2022), pp. 1–9. ISSN: 1911-8074. DOI: 10.3390/jrfm15050215.

[10] Ahmad Baraani-Dastjerdi, Josef Pieprzyk, and Reihaneh Safavi-Naini. “A Practical Electronic Voting Protocol Using Threshold Schemes”. In: *Proceedings of the 11th Annual Computer Security Applications Conference*. New Orleans, Lousiana, USA: IEEE Computer Security Press, 1995.

[11] Massimo Bartoletti and Livio Pompianu. “An Analysis of Bitcoin OP_RETURN Metadata”. In: *Lecture Notes in Computer Science* 10323 LNCS (2017), pp. 218–230. ISSN: 16113349. DOI: 10.1007/978-3-319-70278-0_14.

[12] Silvia Bartolucci, Pauline Bernat, and Daniel Joseph. “SHARVOT: secret SHARe-based VOTing on the blockchain”. In: *WETSEB’18: IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*. 2018, pp. 1–5.

[13] Josh Benaloh. “Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret”. In: *Advances in Cryptology - CRYPTO ’86* (1986), pp. 251–260.

[14] Josh Benaloh and Michael de Mare. “One-Way Accumulators: A Decentralized Alternative to Digital Signatures”. In: *Advances in Cryptology - EUROCRYPT ’93 Proceedings*. Springer-Verlag, 1993, pp. 274–285.

[15] Josh Benaloh and Dwight Tuinstra. “Receipt-Free Secret-Ballot Elections”. In: *26th Annual ACM Symposium on Theory of Computing*. Montreal, Canada, 1994, pp. 544–553.

[16] Josh Benaloh and Moti Young. “Distributing the Power of a Government to Enhance the Privacy of Voters”. In: *Proceedings of the 5th ACM Symposium on the Principles of Distributed Computing*. 1986, pp. 52–62.

[17] Nadja Braun Binder et al. "International Standards and ICT Projects in Public Administration: Introducing Electronic Voting in Norway, Estonia and Switzerland Compared". In: *The Estonian Journal of Administrative Culture and Digital Governance* 19 (2 2019), pp. 8–22.

[18] Stefano Bistarelli et al. "An E-Voting System Based on Tornado Cash". In: *Lecture Notes in Computer Science* 13782 (2022), pp. 120–135.

[19] Stefano Bistarelli et al. "An End-to-end Voting-system Based on Bitcoin". In: *Proceedings of the 32nd ACM SIGAPP Symposium on Applied Computing*. 2017, pp. 1836–1841. ISBN: 9781450344869.

[20] Nur Sakinah Burhanuddin et al. "Blockchain in Voting System Application". In: *International Journal of Engineering and Technology* 7 (4 2018), pp. 156–162. ISSN: 2227524X.

[21] Vitalik Buterin. *Minimal anti-collusion infrastructure*. URL: https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413 (visited on 01/05/2024).

[22] Linh Vo-Cao-Thuy et al. "Votereum: An Ethereum-based E-voting system". In: *2019 IEEE-RIVF International Conference on Computing and Communication Technologies*. Danang, Vietnam, 2019, pp. 1–6. ISBN: 9781538693131.

[23] Marwa Chaieb et al. "Verify-Your-Vote: A Verifiable Blockchain-based Online Voting Protocol". In: *Lecture Notes in Business Information Processing*. Limassol, Cyprus, 2018, pp. 16–30.

[24] David Chaum. "Blind Signatures for Untraceable Payments". In: *Advances in Cryptology* (1983), pp. 199–205. ISSN: 00200255.

[25] David Chaum. "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability". In: *Journal of Cryptology* 1 (1 1988), pp. 65–75. ISSN: 09332790.

[26] David Chaum et al. "Secret Ballot Elections with Unconditional Integrity". In: *Cryptology ePrint Archive* (270 2007), pp. 1–33. ISSN: 00029114.

[27] Chaum1981. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". In: *Communications of the ACM* 24 (1981), pp. 84–90. ISSN: 00010782.

[28] Huan Chen and Yijie Wang. "MiniChain: A lightweight protocol to combat the UTXO growth in public blockchain". In: *Journal of Parallel and Distributed Computing* 143 (2020), pp. 67–76. ISSN: 07437315. DOI: 10.1016/j.jpdc.2020.05.001.

[29] cointree.com. *Top 9 Smart Contracts Platforms (Layer-1 blockchains)*. URL: https://www.cointree.com/learn/smart-contract-platforms/ (visited on 11/09/2022).

[30] Lorrie Faith Cranor and Ron K. Cytron. "Sensus: A Security-Conscious Electronic Polling System for the Internet". In: *Proceedings of the Thirtieth Hawaii International Conference on System Sciences* (1997), pp. 561–570.

[31] Jason Paul Cruz and Yuichi Kaji. "E-voting System Based on the Bitcoin Protocol and Blind Signatures". In: *IPSJ Transactions on Mathematical Modeling and Its Applications* 2016-MPS-107 (7 2016).

[32] Gaby G. Dagher et al. "BroncoVote: Secure Voting System using Ethereum's Blockchain". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. Funchal, Madeira, Portugal, 2018, pp. 96–107.

[33] Chris Dannen. *Introducing Ethereum and Solidity*. Apress, 2016.

[34] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22 (6 Nov. 1976), pp. 644–654.

[35] Tassos Dimitriou. "Efficient, Coercion-free and Universally Verifiable Blockchain-based Voting". In: *Computer Networks* 174 (February 2020). ISSN: 13891286. DOI: 10.1016/j.comnet.2020.107234.

[36] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms". In: *Advances in Cryptology - Crypto '84* (1984), pp. 10–18.

[37] Adam Kaleb Ernest. *Follow My Vote*. 2021. URL: https://followmyvote.com (visited on 2021-05-26).

[38] Jordi Barrat i Esteve, Ben Goldsmith, and John Turner. *International Experience with E-Voting: Norwegian E-Vote Project*. Assessment Report. Washington, U.S.A: The International Foundation for Electoral Systems, June 2012. 188 pp.

[39] Shahinaz Kamal Ezzat, Yasmine N. M. Saleh, and Ayman A. Abdel-Hamid. "Blockchain Oracles: State-of-the-Art and Research Directions". In: *IEEE Access* 10 (5 2022), pp. 67551–67572. DOI: 10.1109/ACCESS.2022.3184726.

[40] Nazim Faour. "Transparent E-Voting dApp Based on Waves Blockchain and RIDE Language". In: *Proceedings of the XVI International Symposium on Problems of Redundancy in Information and Control Systems (Redundancy 2019)*. 2019, pp. 219–223. ISBN: 9781728119441. DOI: 10.1109/REDUNDANCY48165.2019.9003336.

[41] Ethereum Foundation. *ERC-1155 Multi-Token Standard*. URL: https://ethereum.org/en/developers/docs/standards/tokens/erc-1155/ (visited on 11/29/2023).

[42] Ethereum Foundation. *ERC-721 Non-Fungible Token Standard*. URL: https://ethereum.org/en/developers/docs/standards/tokens/erc-721/ (visited on 11/29/2023).

[43] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. “A Practical Secret Voting Scheme for Large Scale Elections”. In: *Workshop on the Theory and Application of Cryptographic Techniques*. Ed. by Jeniffer Seberry and Yuliang Zheng. Gold Coast, Queensland, Australia: Springer-Verlag, 1992, pp. 244–251.

[44] Francesco Fusco et al. “Crypto-voting, a Blockchain based e-Voting System”. In: *IC3K 2018 - Proceedings of the 10th International Joint Conference on Knowledge Discovery Knowledge Engineering and Knowledge Management*. Vol. 3. 2018, pp. 223–227. ISBN: 9789897583308.

[45] Roham Gharegozlou. *Introducing Flow, a new blockchain from the creators of CryptoKitties*. URL: https://medium.com/dapperlabs/introducing-flow-a-new-blockchain-from-the-creators-of-cryptokitties-d291282732f5 (visited on 2022-06-20).

[46] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: *Society for Industrial and Applied Mathematics (SIAM) Journal on Computing* 18 (1 1989), pp. 186–208.

[47] Nicole J. Goodman. “Internet Voting in a Local Election in Canada”. In: *The Internet and Democracy in Global Perspective: Studies in Public Choice*. Vol. 31. Springer, Cham, 2014. Chap. 1, pp. 7–24. ISBN: 978-3-319-04351-7.

[48] J. Alex Halderman and Vanessa Teague. “The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election”. In: *Lecture Notes in Computer Science* (9269 Mar. 2015), pp. 35–53. ISSN: 16113349.

[49] Gang Han et al. “Blockchain-Based Self-Tallying Voting System with Software Updates in Decentralized IoT”. In: *IEEE Network*. Vol. 34. 2020, pp. 166–172. DOI: 10.1109/MNET.001.1900439.

[50] Freya Sheer Hardwick et al. “E-Voting with Blockchain: An E-Voting Protocol with Decentralization and Voter Privacy”. In: *2018 IEEE International Conference on Internet Things*. Halifax, Nova Scotia, Canada, 2018, pp. 1561–1567.

[51] Ch Anwar ul Hassan et al. “A Liquid Democracy Enabled Blockchain-Based Electronic Voting System”. In: *Scientific Programming* 2022 (2022). ISSN: 10589244. DOI: 10.1155/2022/1383007.

[52] Sven Heiberg, Arnis Parsovs, and Jan Willemson. “Log Analysis of Estonia Internet Voting 2013-2015”. In: *Lecture Notes in Computer Science* (9269 2015), pp. 19–34. ISSN: 16113349.

[53] Alexander Hentschel, Dieter Shirley, and Layne Lafrance. *Flow: Separating Consensus and Compute*. URL: http://arxiv.org/abs/1909.05821 (visited on 06/22/2022).

[54] Friðrik þ. Hjálmarsson et al. “Blockchain-Based E-Voting System”. In: *IEEE International Conference on Cloud Computing.* July 2018, pp. 983–986. ISBN: 9781538672358.

[55] Jen-Ho Hsiao et al. “Decentralized E-Voting Systems Based on the Blockchain Technology”. In: *Lecture Notes in Electrical Engineering.* Vol. 474. 2018, pp. 305–309.

[56] E-Voting System using Hyperledger Sawtooth. “Vivek S. K. and Yashank R. S. and Yashas Prashanth and Yashas N.” In: *Proceedings of the 2020 International Conference on Advances in Computing, Communication and Materials (ICACCM 2020).* 2020, pp. 29–35. ISBN: 9781728197852. DOI: 10.1109/ICACCM50413.2020.9212945.

[57] Kenneth R. Iversen. “A Cryptographic Scheme for Computerized General Elections”. In: *Advances in Cryptology* (LNCS 576 1992), pp. 405–419.

[58] Rui Joaquim, André Zúquete, and Paulo Ferreira. “REVS - A Rocust Electronic Voting System”. In: *IADIS International Journal of www/Internet* 1 (i 2003), pp. 47–63.

[59] Wen-Shenq Juang and Chin-Laung Lei. “A Secure and Practical Electronic Voting Scheme for Real World Environments”. In: *IEICE Transactions Fundamentals* (1997).

[60] Wen-Shenq Juang, Chin-Laung Lei, and Horng-Twu Liaw. “A Verifiable Multi-Authority Secret Election Allowing Abstention from Voting”. In: *The Computer Journal* 45 (6 2002), pp. 672–682. ISSN: 00104620.

[61] Kashif Mehboob Kahn, Junaid Arshad, and Muhammad Mubashir Khan. “Secure Digital Voting System based on Blockchain Technology”. In: *International Journal of Electronic Government Research* 14 (1 2018). ISSN: 1548-3886.

[62] Christian Killer et al. “Provotum: A Blockchain-based and End-to-end Verifiable Remote Electronic Voting System”. In: *Proceedings of the 2020 IEEE Conference on Local Computer Networks (LNC).* Vol. November. 2020, pp. 172–183. ISBN: 9781728171586. DOI: 10.1109/LCN48667.2020.9314815.

[63] Kevin Kirby, Anthony Masi, and Fernando Maymi. *Votebook, A proposal for a blockchain-based electronic voting system.* Tech. rep. New York University, Sept. 2016. 14 pp.

[64] Ali Kaan Koç et al. “Towards Secure E-Voting Using Ethereum Blockchain”. In: *6th Symposium on Digital Foresinc and Security.* 2018, pp. 1–6. ISBN: 9781538634493.

[65] Wei-Chi Ku and Sheng-De Wang. “A secure and practical electric voting scheme”. In: *Computer Communications* 22 (3 1999), pp. 279–286. ISSN: 01403664.

[66] Dapper Labs. *Cadence documentation.* URL: https://cadence-lang.org/docs/ (visited on 01/06/2024).

[67] Wei-Jr Lai et al. "DATE: A Decentralized, Anonymous, and Transparent E-Voting System". In: *Proceedings of the 2018 1st IEEE International Conference in Hot Information-Centric Networking*. 2018, pp. 24–29. ISBN: 9781538648704.

[68] Odysseas Lamtzidis. *The State of Private Voting in Ethereum*. 2023. URL: https://odyslam.com/blog/state-of-private-voting/ (visited on 01/05/2024).

[69] Byoungcheon Lee and Kwangjo Kim. "Receipt-free Electronic Voting through the Collaboration of Voter and Honest Verifier". In: *Proceedings of JW-ISC 2000*. Okinawa, Japan, 2000, pp. 1–8.

[70] Kibin Lee, Joshua I. James, and Tekachew Gobena Ejeta. "Electronic Voting Service Using Block-chain". In: *Journal of Digital Forensics, Security and Law* 11 (2 2017), pp. 123–136. ISSN: 15587223.

[71] Matteo Loporchio et al. "A survey of set accumulators for blockchain systems". In: *Computer Science Review* 49 (100570 2023), pp. 1–17. ISSN: 15740137. DOI: 10.1016/j.cosrev.2023.100570.

[72] Ülle Madise and Tarvi Martens. "E-voting in Estonia 2005: The first practice of country-wide binding Internet voting in the world". In: *Proceedings of the 2nd International Workshop on Electronic Voting*. Ed. by Robert Krimmer. Bregenz, Austria, 2006, pp. 15–26.

[73] Emmanouil Magkos, Mike Burmester, and Vassilis Chrissikopoulos. "Receipt-freeness in Large-Scale Elections without Untappable Channels". In: *IFIP Advances in Information and Communication Technology* 74 (2001), pp. 683–694. ISSN: 18684238.

[74] Aanchal Mani et al. "College Election System using Blockchain". In: *ITM Web of Conference* 44 (2022), pp. 1–5. DOI: 10.1051/itmconf/20224403005.

[75] Raphael Matile et al. "CaIV: Cast-as-Intended Verifiability in Blockchain-based Voting". In: *2019 IEEE International Conference on Blockchain and Cryptocurrency*. Seul, South Korea, 2019, pp. 24–28. ISBN: 9781728113289.

[76] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. "A Smart Contract for Boardroom Voting with Maximum Voter Privacy". In: *Lecture Notes in Computer Science*. Vol. 10322 LNCS. 2017, pp. 357–375.

[77] Ralph C. Merkle. "Protocols for Public Key Cryptosystems". In: *Secure Communications and Asymmetric Cryptosystems* (1980), pp. 122–134.

[78] Ralph C. Merkle. "Secure Communications Over Insecure Channels". In: *Communications of the ACM* 21 (4 1978), pp. 294–299.

[79] Johannes Mols and Emmanouil Vasilomanolakis. "ethVote: Towards secure voting with distributed ledgers". In: *International Conference on Cyber Security and Protection of Digital Services and Cyber Security 2020*. 2020, pp. 1–8. ISBN: 9781728164281.

[80] Tal Moran and Moni Naor. "Receipt-free Universally-Verifiable Voting with Everlasting Privacy". In: *Lecture Notes in Computer Science* 4117 LNCS (2006), pp. 373–392. ISSN: 16113349.

[81] Malik Hamza Murtaza, Zahoor Ahmed Alizai, and Zubair Iqbal. "Blockchain Based Anonymous Voting System Using zkSNARKs". In: *Proceeding of the 2019 International Conference on Applied and Engineering Mathematics*. 2019, pp. 209–2014. ISBN: 9781728123530.

[82] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *www.bitcoin.org* (2008), pp. 1–9.

[83] Peter G. Neumann. "Security Criteria for Electronic Voting". In: *Proceedings of the 16th National Computer Security Conference*. Baltimore, USA, 1993, pp. 1–7.

[84] Valtteri Niemi and Ari Renvall. "Efficient voting with no selling of votes". In: *Theoretical omputer Science* 226 (1 1999), pp. 105–116. ISSN: 03043975.

[85] Hannu Nurmi, Arto Salomaa, and Lila Santean. "Secret Ballot Elections in Computer Networks". In: *Computer and Security* 10 (6 1991), pp. 553–560.

[86] Tasuaki Okamoto. "Receipt.Free Electronic Voting Schemes for Large Scale Elections". In: *Lecture Notes in Computer Science* 1361 (1998), pp. 25–35. ISSN: 16113349.

[87] Tatsuaki Okamoto. "An electronic voting scheme". In: *Advanced IT Tools* (1996), pp. 21–30.

[88] opensea.io. *OpenSea NFT Marketplace*. URL: https://opensea.io/ (visited on 02/18/2024).

[89] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. "Efficient Anonymous Channel and All/Nothing Election Scheme". In: *Lecture Notes in Computer Science* 765 LNCS (1994), pp. 248–259. ISSN: 16113349.

[90] Tiphaine Pinault and Pascal Courtade. "E-voting at Expatriates' MPs elections in France". In: *Electronic Voting* (Feb. 2012), pp. 289–195.

[91] pse.dev. *What is Semaphore*. URL: https://docs.semaphore.pse.dev/ (visited on 01/05/2024).

[92] Ron Rivest, Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 26 (1 1983), pp. 96–99.

[93] Zuzana Rjašková. "Electronic Voting Schemes". Master's Thesis. Bratislava, Slovakia: Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University, 2002.

[94] Aayush Sagar et al. "SoulBound E-Voting System". In: *International Journal for Research in Applied Science and Engineering Technology* 11 (3 Mar. 31, 2023), pp. 1520–1525. DOI: 10.22214/ijraset.2023.48548.

[95] Safdar Hussain Shaheen, Muhammad Yousaf, and Mudassar Jalil. "Temper Proof Data Distribution for Universal Verifiability and Accuracy in Electoral Process Using Blockchain". In: *13th International Conference on Emerging Technologies*. 2017.

[96] Adi Shamir. "How to Share a Secret". In: *Communications of the ACM* 22 (11 1979), pp. 612–613.

[97] Shalini Shukla et al. "Online Voting Application Using Ethereum Blockchain". In: *2018 International Conference on Advances in Computing, Communications and Informatics*. 2018, pp. 873–880. ISBN: 9781538653142.

[98] *The Cadence Programing Language*. 2023. URL: https://cadence-lang.org/docs/language/ (visited on 12/02/2023).

[99] tivi.io. *TiVi*. 2021. URL: https://tivi.io/ (visited on 2021-05-26).

[100] Lawrence J. Trautman. "Virtual Art and Non-Fungible Tokens". In: *Hofstra Law Review* 50 (2 2022), pp. 371–426.

[101] Shantanu Vidwans et al. "Permissioned Blockchain Voting System using Hyperledger Fabric". In: *Proceedings of the 2022 International Conference on IoT and Blockchain Technology, ICIBT 2022*. 2022, pp. 1–6. ISBN: 9781665424165. DOI: 10.1109/ICIBT52874.2022.9807702.

[102] Voatz. *Voatz - Secure, accessible voting at your fingertips*. 2021. URL: https://voatz.com/ (visited on 2021-05-28).

[103] Baocheng Wang et al. "Large-scale Election Based On Blockchain". In: *Procedia Computer Science* 129 (2018), pp. 234–237. ISSN: 18770509.

[104] Jin Wang et al. "Blockchain Based Data Storage Mechanism in Cyber Physical System". In: *Journal of Internet Technology* 21 (6 2021), pp. 1681–1689. ISSN: 20794029. DOI: 10.3966/160792642020112106010.

[105] Linjie Wang, Youliang Tian, and Suo Zhang. "Towards Cross-Domain Dynamic Accumulator Authentication Based on Blockchain in Internet of Things". In: *IEEE Transactions on Industrial Informatics* 18 (4 2022), pp. 2858–2867. ISSN: 19410050. DOI: 10.1109/TII.2021.3116049.

[106] Qin Wang et al. "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges". In: *arXiv* (Oct. 24, 2021). URL: http://arxiv.org/abs/2105.07447.

[107] Eric Glen Weyl, Puja Ohlhaver, and Vitalik Buterin. "Decentralized Society: Fiding Web3's Soul". In: *SSRN Electronic Journal* (2022). ISSN: 1556-5068. DOI: 10.2139/ssrn.4105763.

[108] Tifan Wu. "An E-Voting System Based on Blockchain and Ring Signature". Master's Thesis. University of Birmingham, 2017. 54 pp.

[109] Wenbin Zhang et al. "A Privacy-Preserving Voting Protocol on Blockchain". In: *2018 IEEE 11th International Conference on Cloud Computing*. 2018, pp. 401–408. ISBN: 978-1-5386-7235-8.

[110] Zhichao Zhao and T-H. Hubert Chan. “How to Vote Privately using Bitcoin”. In: *Lecture Notes in Computer Science*. Vol. 9543. 2016, pp. 82–96.

[111] Yuanjian Zhou et al. “An improved FOO voting scheme using blockchain”. In: *International Journal of Information Security* 19 (3 2020), pp. 303–310. ISSN: 16155270. DOI: 10.1007/s10207-019-00457-8.

[112] Dimitrios Zissis. “Methodologies and Technologies for Designing Secure Electronic Voting Information Systems”. PhD Thesis. Lesbos, Greece: Department of Product and Systems Design Engineering, Univerity of the Aegean, 2011.