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

October 21, 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

The modernization of voting systems is an active area in academic research, namely in electronic based systems that can improve upon the traditional paper and pen system that many modern democracies still rely on. This research has been mainly focused in developing systems that verify a finite set of security criteria that are used to infer in the ability of such system to deliver the choice of a voter to a tally authority in a safe, transparent and private fashion.

Technological evolution in voting systems before the mid 1970's was restricted to improving on existing voting methods, always with the fundamental limitation of requiring the physical presence of the voter to initiate the process. Both paper ballots and modern touch screen voting terminals limit voting activities by imposing a time and place that does not take into account unforeseen circumstances during the limited availability window nor other limitations that can affect a voter and hinder his or her ability to enact this civic duty.

In 1976, a landmark article by Diffie and Hellman [29] provided the basis for the development of commercial cryptography. Until this point, cryptographic research was confined to governmental (i.e., military) applications. The ideas introduced by this publication triggered a flurry in cryptographic research which produced many of the cryptographic schemes and tools that still support online communication to this day. The development of asymmetrical encryptions schemes based in unidirectional problems, i.e., mathematical problems that are trivial to compute in one direction but unfeasible to solve in the opposite direction, produced popular schemes, such as the *Rivest-Shamir-Adleman (RSA)* [79] cryptosystem, which uses the factorization of the product of two large prime numbers as such intractable mathematical problem. The *ElGamal* cryptosystem is another example that provides similar functionalities but relying in the difficulty of computing discrete logarithms as its one-way function.

Commercial cryptography was a boon for e-voting research. The years following Diffie and Hellman's article saw a significant increase in publications proposing voting systems that use cryptographic schemes and tools as these get developed by other researchers. Though technically sound, none of these proposals was able to make it as a real-world application, mainly due to security concerns. One limiting aspect of these proposals was their reliance in server-client architectures, which are prone to creating a single point of failure in the system, often the point of focus in an attack by an adversary party.

The increase in cryptographic based e-voting proposals also increase the attention of the research community to this topic, namely to the conceptualization of security in these systems, which revealed itself with researchers addressing this topic in e-voting proposals or even dedicating entire publications to it, such as [70] or [37], which were among the first defining trust in an e-voting system through the implementation of specific security criteria, with the rationale that trust in the system is proportional to the number of criteria implemented. These criteria add the properties that define them to the system, such as *accuracy* if a system is able to prove that no invalid votes can be added to the final tally, *privacy* if it is able to maintain the secrecy of a vote from the moment it is cast to when it is finally tallied, etc.

The nomenclature as well as the definition of such criteria is still quite subjective among relevant publications, as a consequence of a lack of standardization in this regard, with the same criteria being often referenced in publication by different names. [4] performed an extensive systematic literature review among e-voting proposals spanning several decades with the goal of establishing an unified set for these criteria. These include *accuracy*, *privacy*, *eligibility*, *verifiability* and *robustness* as a **minimal set**, as in the criteria that directly relate to increased overall security. An **additional set** containing the *convenience*, textitflexibility and *mobility* criteria was also included in this publication as criteria that simplify the usage of the system by a voter, but do not directly contribute to a more secure operation.

E-voting research followed the cryptographic stream for around 30 years, until another landmark article was published in 2009. [69] introduced Bitcoin and blockchain to the world, with the latter being a critical development in the development of a new approach to e-voting. Blockchain is also heavily dependent in cryptography, and as such this new development was not as extreme, providing some continuation from the techniques developed thus far. But the new decentralized approach provided a new avenue of research and soon after the first blockchain-based e-voting systems were published in academia. The evolution of such systems is in itself a window into the evolution of blockchain: earlier proposals were limited to find creative ways around cryptocurrency transactions, which for some years was the sole application of blockchain technology, while later proposals use Smart Contracts, as well as functionalities from the Distributed Virtual Machines that support them, extensively.

This proposal follows the blockchain stream by exploring how Non-Fungible Tokens (NFTs), relatively new feature that gained popularity in recent years, can be used to implement a voting system. NFTs are used to establish ownership of digital objects in a blockchain. Up to recently, most NFT applications are reduced to digital collectibles and digital art. Our approach is to use them to represent digital votes and research on how the mechanism in which ownership is established in the blockchain can be used to devise a more secure e-voting system. In addition to this proposal, we also extend this study to include an implementation in Ethereum, the most popular blockchain with smart contract support, and another implementation in Flow, a more recent blockchain implementation that was optimized for NFT operations, namely transactions and storage model used. These two implementations provide valuable insight to the usefulness of Non-Fungible Tokens in an e-voting context.

The rest of this article is structured as follows: the following Section 2 details the current state of related works. Section 3 provides an introduction to the fundamentals workings of a Non-Fungible Token. Sections 4 and 5 detail the implementation of an basic e-voting system that uses NFT metadata to encapsulate vote selections, and Section 6 provides a summary of the results and an objective comparison between the two approaches considered. Section 7 concludes the article.

2 Related Works

The commercialisation of cryptography triggered a stream of new approaches to e-voting systems using new cryptographic schemes and tools derived from these to implement these ideas. Diffie and Hellman's 1976 publication [29] was followed by new symmetrical and asymmetrical cryptographic schemes proposals, such as [25], [31], and [79], which in turn were instrumental to define cryptographic tools that have been extensively used in e-voting system development, such as blind signatures [22], Mix-Nets [23], Homomorphism in threshold cryptosystems [82] and cryptographic knowledge proofs [40].

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 the cryptographic tools indicated thus far. 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. [70] was among the first to attempt such characterisation, with subsequent publications, such as [37], [10], [52], [58], [60], [51], [8], and [24], continuing this trend. These articles followed the rationale that the more security of a voting system is proportional to the number of security criteria it implements, which translates in an assurance that a voter 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 crucial characteristic that can hinder a wide adoption of the proposed system. Contemporary elections can go from exercises where the system is only expected to process up to a thousand votes at one point, to national-wide events that might require the processing of millions of votes. The relationship between the scalability of a system and the amount of available resources is 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 [26], [72], [50], and [71], this trade-off was shifted towards security and transparency at the expense of scalability. The authors do recognize this limitation and, as such, are explicit in restricting the usage of their system to small-scale elections.

On the other side of this spectrum, proposals such as [14], [15], [76], [53], [74], [73], [62] or [67] present scalable systems that are simpler than the ones considered in the last paragraph, but their adoption in large-scale elections implies a sacrifice in security and transparency. Proposals suitable for large-scale elections implement the lowest number of security criteria.

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 [41], Estonia in 2005 [45], Switzerland in 2005 [16], Norway in 2011 [33], France in 2012 [77], and Australia in 2015 [42].

The first decentralised e-voting proposals were limited to cryptocurrency-centric blockchain, such as Bitcoin, due to the lack of alternatives in the early years of blockchain development. This proposals were somewhat simple in the sense that they conceived convoluted and impractical methods to exchange information using transactional metadata from cryptocurrency transfers. Proposals such as [92], [27], [18], [61], [81], [90], [30] 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. 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. 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 low block rate. Bitcoin adds a new block every 10 minutes, which limits 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 such as [56], [7], [21], [19], [91], [54], [68], [34], [55], [43], [49], [63], [93], [5], [44], [86] or [64] adopted private blockchains using custom-made solutions in customisable frameworks such as Hyperledger Fabric, Quorum, and Multichain. 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 to 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. Proposals such as [65], [57], [28], [38], [48], and [66] were among the first to implement a voting system via Ethereum smart contracts.

The same time period also produced some blockchain-based proposals for e-voting systems in a real-world scenario. Unlike the centralised approach, these solutions have significant overlap with the academic proposals considered. Among these real-world examples, we cite *Follow My Vote* [32], *TiVi* [84], *Agora* [2], and *Voatz* [87]. The difference between the nature of approaches regarding their real-world applications is an indicator of the potential of blockchain in this scenario. Real-world blockchain-based e-voting solutions follow the academic approach 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 protocols that can be used to establish a secure and transparent e-voting system. [59] 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 [20], *Semaphore* [78], *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 vote element.

So far, none of the proposals considered used, or even mentioned, NFTs in their processes. Nevertheless, a search for recent proposals with this characterizing element was conducted. 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 success. The closest article to a NFT-based e-voting system we were able to consider was [80], where the authors use SoulBound NFTs, a special case of non-transferable NFTs that can potentially be used for identification purposes [89], to circumvent the need for a trusted third party to implement voter *eligibility*. The proposed system only interacts with these NFTs during voter validation. 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 how limited their use of NFTs is.

Two other proposals, namely [17] 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 [88] 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. [88]

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

Non-Fungible Tokens (NFTs) follow cryptocurrencies as another base feature made possible by blockchains. From the network point of view, cryptocurrencies associate user addresses to a value: the cryptocurrency balance of that account, in that network. NFTs invert this logic to create a new way to represent digital ownership. Each NFT is created as digital object but with an internal parameter (typically named 'id' or similar) that ensures that every object is unique by setting them with a different id. From the point of view of the blockchain, a NFT is a series of synchronized records that create an unequivocal relationship between the unique id of the NFT and the account address that owns it.

The NFT concept is transversal to all blockchains, but since the concept was introduced through the Ethereum blockchain, 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 [36] and ERC-1155 [35]. It is possible for someone to define an NFT outside of these standards since they are not legally enforceable. 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 assurance, due to the function and parameter 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. For example, if a user is interacting with a NFT smart contract that implements the ERC-721 standard, the user can call the function 'balanceOf' without needed to check the smart contract code to see if it was implemented.The user can provide an address as argument to it and find out how many NFTs from that contract the given address currently owns. The 'balanceOf' function is made available by the interface and it is available by any contract that implements the ERC-721 interface. Some interface functions can be overwritten to include additional functionalities or requirements, but this process preserves the function name and, therefore, the guarantee that if a user invokes it without checking the contract code a priori, this invocation does not raise any errors and retains the basic functionalities.

The application potential of this technology regarding digital collectibles has in itself motivated the creation of NFT-centric blockchains, such as Flow [46], 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 [75]) as long as the NFT smart contract implements the standards indicated. Popular NFT marketplaces usually only accept tokens whose implementing smart contracts ensure the implementation of some standards, ERC-721 one of the fundamental ones.

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 [85].

3.1 NFTs vs. Cryptocurrencies

In a blockchain context, NFTs represent objects, while cryptocurrencies are variables. With cryptocurrencies, account addresses are the "keys" while their cryptocurrency balance is used as "value", if a key-value scheme is used to represent the cryptocurrency balances in the network. Account addresses have their balances being modified through transactions, but the correspondence in this case is always 1:1, i.e., one account address to one cryptocurrency balance. When working with NFTs, this relationship is also 1:1, but the arguments are inverted, with the NFT's ids as the unique "keys" and the owner's address as the corresponding "value" in the same key-value scheme. This arrangement reinforces the unique nature of NFTs by enforcing the uniqueness of their id set, i.e., *digital scarcity*, while their respective owners can change, thus allowing NFTs to be transferred among accounts. This aspect is also what gives NFTs their "Non-Fungible" characteristic. Since each has their own record, each NFT is it's own unique digital record and cannot be exchanged by another with the same characteristics because, by definition, there is none. Cryptocurrencies work inversely, and their tokens do not have any kind of unique identifiers, therefore any unit of a cryptocurrency is functionally equivalent to all other units of the same token. Users can exchange equal quantities of a cryptocurrency without losing anything in the process (other than transaction fees), thus exploiting their fungible aspect.

Though most NFTs produced thus far represent **digital** objects, there are no strict requirements in that regard. NFTs can 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 contained 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 more efficient and flexible means to achieve the same result. 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 only through a transaction digitally signed by the NFT owner.

3.2 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. The substantial differences between these architectures justifies an exploration on how NFTs might fare in a e-voting context.

3.3 Token Burning

Token ownership in a blockchain is abstracted by the ability of a user to access the account containing that token or the capability of that user to generate a valid digital signature that can sign a transaction to transfer that token to some other location. This implies that the user owns or controls 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 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. But anyone is free to transfer a token that he or she owns to this zero-address. 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 stored in an unaccessible account [6].

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, while exploring the differences of using a contract-based blockchain versus a account-based approach.

3.4 Flow Blockchain

This proposal is based in two implementations in two quite different blockchains. We omit the introduction to the Ethereum blockchain on the assumption that, since it is the most used blockchain in academic research, it is easy for the reader to familiarize himself/herself with it if needed. The Flow blockchain is still relatively unknown, especially within the academic context. This section provides a brief introduction to this blockchain, for context.

The Flow blockchain was formally launched in October 2020 through the usual *Initial Coin Offering (ICO)* that helped launch most public blockchains in recent years. Flow followed a trend that was addressing the high energy consumption associated to earlier, *Proof-of-Work* based blockchains, and established itself with the more efficient and energy friendly *Proof-of-Stake (PoS)* consensus mechanism [46].

In a Proof-of-Stake based blockchain, new blocks are "mined" by active nodes based on their *stake* on the network, namely, the volume of the blockchain's native cryptocurrency the node currently holds in its account. Higher stakes mean higher probability of being awarded a block publication by the protocol that regulates the network, with the token incentives that such entitles, and vice versa. This process is more energy efficient and faster than the "classical" PoW consensus, where nodes pointlessly spend computer cycles solving cryptographic puzzles only to win a race towards its completion. Given the significant energy waste due to the popularity of PoW blockchains, such as Bitcoin, a blockchain that stays clear of such nefarious protocol is not just a smart choice but a conscientious one as well [47].

Flow was created by the same team that created the *CryptoKitties* project in Ethereum, one of the first NFTs based blockchain games in this network to adopt the *ERC-721* standard for Non-Fungible Tokens. Even though Ethereum was the first one to offer NFT support, it did so on top of a chain that did not envision such applications at the time of its conception. This was evident by the lack of flexibility, inherent security flaws in the code implementing the tokens and its mechanics, as well as a general complexity in writing code and operating with these constructs in the Ethereum network. Ethereum was so ill prepared to deal with the popularity of this initiative that its network went down briefly due to its inability to cope with the added network traffic [13]. Flow was developed with NFTs support as its main feature, prioritizing NFT creation and mechanics over other applications, a clear contrast over blockchains focused in cryptocurrency transactions [39].

Flow establishes a new computational paradigm in this context, namely a *Resource Oriented Paradigm (ROP)*. It does so through *Cadence*, a smart contract programming language developed for Flow. *Cadence* establishes *ROP* through a special type of object, named *Resource*. From a technological point of view, a Resource is akin to a structure or an object from an Object-Oriented context, offering a significant degree of freedom on the type of data that it can encode. But Cadence regulates operations through the notion that every Resource is unique in Flow. This means that Resources need to be accounted at all times, and they cannot be copied, only moved. They can be stored in decentralized storage accounts and loaded from them but, at all times, the Resource is either in storage or out of storage, never in the same state at the same time. These Resources can be created only through smart contracts functions and are exclusive to the Flow blockchain, i.e., not interchangeable with other NFTs from other blockchains.

3.4.1 Account-based Storage in Flow

Flow was conceived to be a NFT-oriented blockchain, unlike the usual, cryptocurrency focus of other alternatives. As such, data storage was emphasized regarding other blockchain characteristics. Other blockchains, Ethereum being a prime example, centralize NFT storage in the respective contract, i.e., the digital object - owning account key-value pair is saved related to the contract. This means that, if the contract is deleted, ownership of its issued NFTs disappears with it, as well as the contract-based NFTs. This mechanic was substantially changed in Flow. Storage in this blockchain is account-based and every NFT created has a specific data type inherited from the contract that minted it, which allows these NFT's to exist outside of the contracts that implement them. Creating an account in Flow is a process similar to creating a cryptocurrency wallet in any other blockchain in the sense that it is also based in the creation of an asymmetric encryption key pair, where the account address is derived from the public key and account access requires control of the corresponding private key. Storage also follows a key-value scheme, as it is common in this context, but every key stored is prefixed by the account address, which is also unique for every user in the Flow ecosystem. A Flow NFT stored in an account is uniquely identified by a concatenation between the account address of the owner of the token, followed by the name of the contract that implements the token and the name of the token itself, using a period to separate the elements. For example, consider a Flow NFT named "ExampleToken" that was defined in a smart contract named "ExampleContract" and it is owned by account 0x1234. This token is uniquely identified in the Flow blockchain by the unique string "0x1234.ExampleContract.ExampleToken". Storage space depends on the amount of Flow token currently deposited in the account, according to a rate of 100MB of storage per token [83].

Users control the visibility of the data stored in their accounts by issuing capabilities to other accounts, which can be controlled by other users or other smart contracts. Flow uses different storage domains to implement these visibility levels. By default, once a resource is stored in an account's storage, it goes to the **/storage** domain where only the owner can access it and modify it. This user can then issue a capability that essentially creates a public reference in the **/public** domain. The capability is able to filter which parameters are visible to other users and any interactions happen always through the reference, i. e., never the actual digital object [46].

3.5 The case for Flow

Flow strengthens its case as a prime candidate for a NFT-based project by offering a relatively high block rate and low gas fees, which translates into fast transactions and low overheads. It also uses a novel way to store data, using an account-based storage instead of the "traditional" contract-based storage in Ethereum and similar blockchains. The mechanism of storing resources under the user's account, it is uniquely apt to a context where sensible data needs to be transferred through the blockchain. Securing this data under an individual account, and set it private so that only the account owner can access it, is a feature that distinguishes this blockchain from similar ones. It is easy to understand the security problems behind a more "conventional" blockchain storage system, where all data is referenced from a single point of access, typically the regulating contract address, a centralizing element in a purely decentralized application.

4 Solidity Implementation in Ethereum Blockchain

4.1 Introduction

Ethereum is the most popular blockchain in academic research. Smart contracts were introduced with this blockchain, whose computation is delegated to the *Ethereum Virtual Machine (EVM)*, essentially a distributed computing platform composed by the aggregated resources (CPU cycles, storage, memory, etc.) of the nodes that are active in the network at any point to run the computations required by smart contract function calls. This feature expanded the scope of usage of blockchain immensely and the academic community was quick to capitalise on it.

The popularity of this chain warrants that we implement this system using a Solidity implementation, the programming language used to write Ethereum smart contracts, to both provide a standardised implementation and to serve as a baseline, since Solidity contracts have a more robust knowledge base than any other types, for future comparisons. This standardisation is achieved by implementing ERC standards in the contracts that establish the framework for this system.

Solidity implements NFTs in a non-structured fashion, in the sense that these are not digital objects from a programming standpoint. Solidity NFTs are not like typical digital objects from any Object-Oriented programming language, which are defined by internal parameters and functions. In Solidity, any NFT parameters are established by mapping the unique NFT identifier to another variable (integer, url string, address, etc.) and the collection of all these mappings with the same id key compose the actual NFT object. Functions are defined in the same smart contract that defines the NFT, so these are not necessarily NFT functions, as some languages allow objects to have. These functions may be limited to operate on NFT-defining mappings, thus restricted to operate on NFT data, but they belong to the containing contract and have no logical relation with the NFTs themselves.

Due to the lack of specificity in Solidity NFTs, as in they do not carry a data type themselves, from this point forward we designate the NFTs used to carry the votes as **VoteNFT**.

4.2 Smart Contract Implementation

The main voting contract was defined to emulate a voting booth. Like one, this contract can give a ballot to a voter as a NFT transfer to the voter address, allow the voter to vote by changing a parameter in the VoteNFT metadata to his or her choice and even burn the voteNFT.

The vote function also allows for multiple vote casting, i.e., after submitting a vote, if the election is still active, the voter can submit another voteNFT at a later stage and this one replaces the previous one. From this implementation point of view, the voter only edits the same NFT field again, since it is a more practical and cheaper (gas wise) alternative to burn the old VoteNFT, mint a new one and run the vote function again. Additional edits are protected against unauthorised uses like the initial vote.

If a voter can replace a vote, he/she should also be able to revoke it. This feature is established by a 'burn' function that adds more requirements for its execution, but otherwise it is inherited from the *ERC721Burnable* standard implemented, i.e., it transfers the VoteNFT to address 0.

4.2.1 ERC Standards Used

The voting booth smart contract implements the base *ERC721* standard that implements the majority of internal mappings that define our VoteNFT, as well as useful functions such as *balanceOf* and a base NFT minting function. This standard can be expanded with other *ERC721* based "sub-standards", i.e., standards based on the *ERC721* that add new functionalities to a NFT implementation. The *ERC721URIStorage* and *ERC721Burnable* were added to the base *ERC721* standard. The *Burnable* extension adds the *burn* function and the *URIStorage* extension includes a id to string mapping that we use to encode the vote, but it is often used to write a URL that points to the "actual" digital object abstracted by the NFT (image, video, audio clip, etc.). This standard also exposes a **_setTokenURI** internal function that can be used to modify the NFT metadata towards our goals.

4.2.2 Using *balanceOf* to Ensure One Person, One Vote

A fundamental requisite for any voting system, electronic or otherwise, is to allow one and only one vote per eligible voter. This is actually a fundamental requirement from representative democracies, and not necessarily from the systems used to capture those votes. It is important to note that this requirement does not conflict with the *multiple vote casting* feature nor with the ability of a voter to revoke an already submitted vote. Allowing voters to cast multiple votes does not violates the "one person, one vote" rule, as long as only one of those submission ends up in the final tally (usually the last one submitted). The same logic stands with allowing votes to be revoked, in the sense that submitting zero votes is also a valid scenario for a voter.

The *ERC721* standard provides a *balanceOf* function that receives an argument as input and returns the number of NFT tokens that are defined in the contract instance that are owned by such address. This function returns the value of one of the *ERC721* internal mappings, namely the **_balances** one, associated to the address key. By ensuring that this field can only be 0 or 1, the system ensures that no voter can hold more than 1 VoteNFT at any point in the process. Since this function is made available by implementing the interface and it does exactly what it is required to implement this requirement, no additional functions were created and this one is used whenever this verification is necessary.

The *ERC721* standard ensures that, as long as one uses the functions inherited from the standard to operate on NFTs, namely, minting and transfers, the internal **__balances** mapping is always updated correctly. With that in consideration, we employ functions from the implemented standards as close to their original and tried format as possible. Also, all alterations included when overriding one of these standardises functions do not interfere in any capacity with this mapping.

4.2.3 The VoteNFT

Mappings Solidity defines NFTs as an aggregate of mappings that establish relationships between a unique identifier in the NFT and other parameters. NFTs are defined in Solidity in a similar fashion as centralised databases work, namely, how a database is usually spread through multiple tables that maintain relationships between their records, typically by having sharing one or more columns in common. The full set of parameters that define a NFT are found by retrieving every mapping record that references the unique NFT identifier. These mappings include all defined in the original smart contract and any inherited from implementing standards.

As such, the proposed implementation for this token can be explained by the mappings that compose it. Fig. 1 identifies these mappings as well as their origin, i.e., if these were inherited from one of the standards implemented or if they were created in the NFT smart contract.

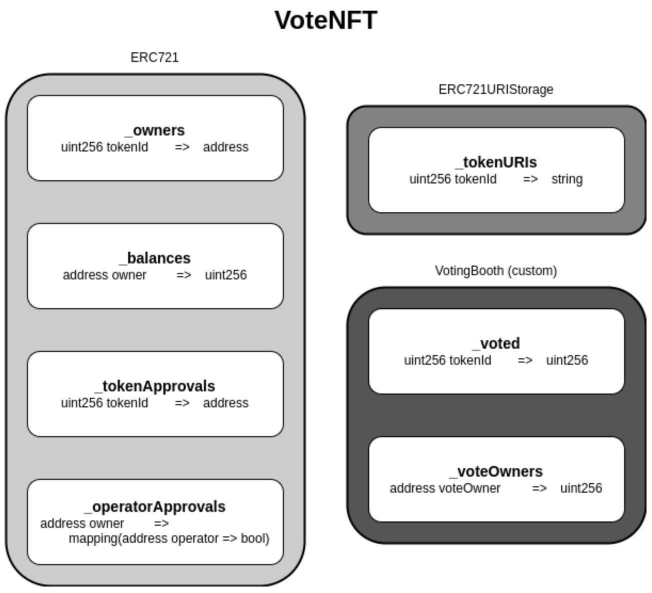
VoteNFT
ERC721
_owners
uint256 tokenId => address
_balances
address owner => uint256
_tokenApprovals
uint256 tokenId => address
_operatorApprovals
address owner => mapping(address operator => bool)
ERC721URIStorage
_tokenURIs
uint256 tokenId => string
VotingBooth (custom)
_voted
uint256 tokenId => uint256
_voteOwners
address voteOwner => uint256

Figure 1: Mappings used to implement the Solidity version of VoteNFT.

The mappings that establish the base definition of this token are all inherited from the *ERC721* standard, namely the **__owners** and **__balances**, which are used to establish the ownership of the issued tokens and to ensure that only one token gets owned per voter. The **__tokenApprovals** and **__operatorApprovals** are used to add two layers of access control to the tokens produced by the contract. These can be used to delegate access to a minted token to accounts other than the owner. This is antithetical to the purpose of the vote tokens, therefore these are not currently used. In fact, having any data in either of these mappings might be interpreted as a security risk since they might allow access to accounts other than the one from an eligible voter, which can be used to modify the voteNFT data after the owner has done so, for example. These mappings are all inherited from the *ERC721* standard.

This next part might need revision in the future, if I decide to use this mappings to change the whole token access logic

The *ERC721URIStorage* standard provides the **__tokenURIs** mapping that is used to store the ballot contents. Most NFTs so far use this field to set an URL pointing to the digital resource that characterises the NFT (an image, video or audio clip) but in this case we used it to encode the voter option. This way, all the information regarding the vote remains in the blockchain, therefore it enjoys all the data immutability and transparency from having this data distributed through a series of nodes in the network. Setting a URL to a proxy server where the actual ballot is negates this entirely and it is a major security breach since it introduces an external element to the blockchain network. Realistically, the final tally for a particular smart contract can be computed by reading and counting all the values in this mapping since all the data needed to determine the winner is contained in this mapping.

The mappings inherited from the standards are sufficient for most of what is needed, but to ease some operations, two extra, custom mappings were added to the VoteNFT smart contract, namely, the **__voted** and **__voteOwners** mappings. **__voted** maps the token's unique identifier to another integer that represents the number of times that the metadata of the VoteNFT with the id in question was edited after minting, which in this context means a new vote being submitted. This mapping is only being used to detect if a voter is voting for the first time (_voted[tokenId] = 0) or if the voter is replacing a previously submitted vote (_voted[tokenId] >= 1) and, so far, it is being used to determine which event should be emitted, namely the *VoteSubmitted(voteId)* event or the *VoteModified(voteId)* event. These events signal that a vote with a specific id was submitted or modified but without revealing any more information about it than the id of the token itself. The **__voteOwners** mapping is a simple inversion of the *ERC721* **__owners** one. It maps the owner address to the id of the VoteNFT awarded to him/her and it is used to provide redundancy to the **__balances** mapping in the sense that locks one address to one voteId (Solidity mappings, just like dictionaries in other programming languages, do not allow duplicate keys) more strictly than using the **__balances** mapping alone. It also provides a easier and cheaper (gas wise) way to determine if a user address already has a token issued to it. Without this mapping, the only process to determine this is to exhaustively check every value from the **__owners** mapping, which can be a very time and gas expensive operation.

Relevant Contract Functions Alongside the mappings indicated in Section 4.2.3, the VotingBooth smart contract contains a series of functions used to operate in the NFTs created by the same contract:

- *Constructor* The constructor function for the VotingBooth smart contract, not the VoteNFT. The constructor in a smart contract is a function that runs once when the contract is deployed. It can take input arguments but once the contract is deployed, this function never runs again. The constructor does not influence the VoteNFT tokens, but it is used to define the election in question by establishing internal contract parameters such as the name and location, as well as the ballot string, i.e., the list of candidates in play, a referendum question, etc.

- *mintVoteNFT* The minting function to generate a VoteNFT. DMinting VoteNFTs is a critical operation in this system. As such, this function can only be executed by the contract deployer. If a transaction signed by another account other than the one that deployed the contract, a requirement at the top is going to prevent it from continuing if the field *msg.sender* does not matches the allowed address. The function sets the sole input argument, an address, as the NFT owner. Minting a VoteNFT consists in taking an internal counter value as the id for the token that is about to get created, and then proceeds by using that id to create a record into the mappings indicated in Fig. 1. The function finishes by incrementing the internal counter for the next mint call and emitting the event that signals a successful minting.

- *vote* Similar to the minting function, the vote function is also limited to the owner of the NFT and validates this requirement using a similar requirement at the top. Only transactions signed by an address that matches the address in the **_owners** mapping for the id in question is allowed to proceed. If this validation is cleared, the function replaces the contents from the **_tokenURIs** mapping for the string provided as input. If this modification is the first one after minting, the action is assumed as vote. Otherwise it is assumed as a replacement vote. The function finishes by emitting a *VoteSubmitted* or *VoteModifies* event accordingly

- *burn* This function removes a VoteNFT from circulation by transferring it to an unrecoverable address. Voters have the option to revoke a previously submitted vote without requiring to provide a substitution one. In these cases, the *burn* function is executed instead. Burning a VoteNFT is restricted to the token owner as well and a *VoteBurned* event gets emitted when the function terminates successfully.

4.3 Usage example

A usage example is very useful for a tutorial, but is it worth doing it in a architecture proposal article?

Encryption layer

5 Cadence Implementation in Flow Blockchain

Cadence Implementation in Flow

6 Results and Implementation Comparison

Results and System Comparison

7 Conclusion

Conclusion

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] bbc.com. *CryptoKitties craze slows down transactions on Ethereum*. URL: https://www.bbc.com/news/technology-42237162 (visited on 06/22/2022).

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

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

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

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

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

[26] 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.

[27] 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).

[28] 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.

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

[30] 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.

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

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

[33] 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.

[34] 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.

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

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

[37] 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.

[38] 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.

[39] 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).

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

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

[47] Alexander Hentschel et al. *Flow: Separating Consensus and Compute - Block Formation and Execution*. URL: http://arxiv.org/abs/2002.07403 (visited on 06/22/2022).

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

[49] 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.

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

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

[60] 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.

[61] 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.

[62] 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.

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

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

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

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

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

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

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

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

[76] 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.

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

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

[79] 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.

[80] 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.

[81] 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.

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

[83] Flow Development team. *Storing Data on Flow*. URL: https://developers.flow.com/learn/concepts/storage (visited on 11/10/2022).

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

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

[86] 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.

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

[88] 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.

[89] 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.

[90] Tifan Wu. “An E-Voting System Based on Blockchain and Ring Signature”. Master’s Thesis. University of Birmingham, 2017. 54 pp.

[91] 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.

[92] 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.

[93] 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.

Todo list