VoteBlok: Blockchain-Based Governmental Election System

Mert Yapucuoglu, Murad Mikayilzade, Ahmet Turan Bulut

Computer Science
University of Oregon
1585 East 13th Avenue Eugene, OR 97403
merty@uoregon.edu
muradm@uoregon.edu
turanb@uoregon.edu

***Abstract*– The act of voting is a fundamental pillar of democratic societies, yet traditional voting systems have faced challenges related to security, transparency, and accessibility. In response, VoteBlok emerges as a transformative solution, harnessing the power of blockchain technology to modernize the voting process. By leveraging the inherent properties of blockchain - decentralization, immutability, and transparency - VoteBlok establishes a secure and transparent voting system.**

VoteBlok is a comprehensive voting system composed of a multiple-page website, secret API, government database, and a blockchain smart contract. This paper outlines the design and implementation of each component, highlighting the necessary adjustments to the blockchain smart contract, secure integration of the API, and deployment of the front-end for voter use. The workflow of the system involves the election agency configuring the contract, integrating the API endpoints, and connecting a government database. Voters register using a dedicated webpage, receive a unique identifier, and subsequently participate in the election by selecting their preferred candidate. The system processes API requests and records votes as transactions on the blockchain. After the election, the "Election Results" page displays the outcomes and the addresses associated with each candidate, allowing voters to verify their votes. This paper provides a comprehensive overview of the VoteBlok system's design, workflow, and the research objectives it aimed to address, promising a secure and transparent voting experience for democratic processes.

***Keywords*– VoteBlok, blockchain-based elections, governmental election, secure voting, immutability, election auditing, accessible elections, decentralization, election auditing, transparent elections**

1. Introduction

The act of voting lies at the heart of democratic societies, allowing citizens to exercise their right to have a say in the decisions that shape their communities and nations. However, traditional voting systems have long faced challenges related to security, transparency, and accessibility. These shortcomings have led to concerns about voter fraud, tampering, and the overall trustworthiness of election results. In the digital era, leveraging innovative technologies becomes imperative to address these issues and modernize the voting process.

VoteBlok emerges as a transformative solution, harnessing the power of blockchain technology to transform the way we vote. By employing the inherent properties of blockchain - decentralization, immutability, and transparency- VoteBlok strives to establish a secure, and transparent voting system. With VoteBlok, participants can engage in elections and referendums with ease, confident that their votes are protected by cutting-edge cryptographic algorithms and a distributed ledger that cannot be tampered with.

The core principle behind VoteBlok is to provide a voting system that is resistant to manipulation, fraud and coercion. By utilizing encryption techniques, each vote cast through

VoteBlok is securely recorded as a transaction on the blockchain, ensuring its integrity and immutability. Moreover, by leveraging advanced authentication mechanisms and privacy-preserving techniques, VoteBlok safeguards the identity and confidentiality of voters, fostering trust in the overall process. By embracing blockchain technology, VoteBlok eliminates the need for physical ballots, reducing paper waste and the associated environmental costs. The decentralized nature of the blockchain further minimizes the reliance on centralized physical infrastructure, such as polling stations and transportation logistics, which can consume considerable energy and resources. As a result, VoteBlok presents a more sustainable alternative, aligning with global efforts to promote environmentally friendly practices and reduce carbon emissions.

The adoption of VoteBlok voting system holds immense potential in simplifying the process of election auditing and combating election fraud. Any attempt to manipulate or tamper with the recorded votes would be immediately apparent, as the decentralized nature of the blockchain ensures that a consensus among multiple network participants is required for any changes. With this technology, the likelihood of detecting and preventing election frauds is significantly enhanced, fostering trust in the electoral system and ensuring that the voice of every voter is accurately reflected in the final results.

2. Related Work

Blockchain is a platform where everyone can easily write their own smart contracts and deploy them for other people to use. VoteBlok is not the first voting dApp written on blockchain nor will be the last, but it is the first fully complete and scalable dApp created on Ethereum's blockchain, requires no wallet provider such as MetaMask so that every citizen can practice their rights to vote without having to be a tech-savvy person, with the intention of being it fit to governmental elections.

3. Design

Voteblok is a sum of its parts, containing a multiple-page website, secret API and government database, and a blockchain smart contract. The government must adjust the blockchain smart contract, securely integrate the API into their servers and databases, and publicly deploy the front end for voter use. The design and implementation of each part are further elaborated in their respective chapters further below.

The workflow of the system is as follows: The election agency will adjust the contract by adding the candidates and appointing a public blockchain address to them. Then they will integrate the API endpoints into the front-end, and connect a government database that fits the requirements in 3.4. Voters will register for the online vote using the "Register for Voting" webpage and receive a "Unique Identifier". They will use this identifier to log in to the "Participate in Election" page. Then they will choose who they want to vote for, upon which several API requests will follow, and at last their vote will be sent as a transaction to the blockchain. The user will be shown their blockchain address and will be told to keep it if they wish to confirm their vote afterward.

When the election is over, the results and the addresses that voted for each candidate will be shown on the "Election Results" page. The voters will be able to find their blockchain address to confirm that their vote has been counted. This is a general overview of the design and workflow of the Voteblok system. Before we delve into each of these parts, it is important to talk about their reason, and what our research aspired to fix.

3.1 Design Rationale

Contrary to our hopes and ambitions, a pure blockchain system is not enough to deliver a satisfying election system, let alone a governmental one. One of the reasons is that blockchain is anonymous by nature, untraceable, and hard to monitor. These are not unappreciatable virtues on their own, however, when they converge without additional support, make it impossible to make an election system that is trustworthy. There are several concerns that arise from this issue: The determination of eligible voters, preventing duplicate votes, gas fees of the vote transactions, retaining voter anonymity, and providing vote confirmation. We will inspect how we have attempted to solve these issues in the upcoming chapters, while also implementing a functional online voting system.

3.1.1 Eligible Voter Determination

This is an online system. Contrary to a regular election, we do not connect the voter to their address as we do in mailed ballots, or to their identification and faces as in regular go-and-vote voting. We have only an internet connection with the potential voter, or a malicious user. It is important to the whole process that only those who must be eligible to vote have access to the inner system. For this purpose, we have locked all of the voting processes behind a “Unique Identifier” that is assigned to a social security number. However, this begets yet another problem where we need to confirm if a potential voter is indeed who they claim they are. To solve this, we have a webpage “Register for Voting”, where we accept sensitive information of the potential voter, and determine if they are eligible. The process if further explained in 3.4. As a result, the voter will have a unique identifier, which they will use to log in to the voting system when the voting begins.

3.1.2 Preventing Duplicate Votes

Even if the “Unique Identifier” solves the issue of determining eligible voters, as long as the contract is public and the candidate addresses are known, anyone can send a voting transaction and tamper with the voting process. To address this problem, we enforced some limitations with the smart contract. Only voters registered in the smart contract can send a voting transaction, and a voter can only be registered by the contract owner. This ensures that the website is the only place where votes can be cast, as the contract owner is the one who decides who can vote. The details of this is explained in 3.2. Also, even if the user is voting through the system, we need to ensure they can’t log in to the voting page twice, or prevent staying on the voting page through some means and voting more than once. To fix that, before the voting process even begins after the button is clicked, the front-end sends an API call with the personal information to tag the voter as having voted. Along with this, the page will check the login credentials of the user twice to ensure valid login.

3.1.3 Gas Fees of Transactions

Naturally, a blockchain system needs to be hashed. Hashing and inclusion of the transactions in the blocks means gas, meaning money. It would be preposterous to expect such a system to be used when a free alternative exists. To solve this issue, we made it so that the government creates blockchain addresses as many as the registered online voters, and fills them with adequate funds. When the voter casts their vote on the website, a request with no personal information is sent to the API, which provides a blockchain address to be used for transactions. This is further explained in 3.4.

3.1.4 Retaining Voter Anonymity

The blockchain itself is fine with regard to anonymity. However, because the address is provided by the government, we need to ensure that the address is not related to the voter in the process. To fix this, we designed it so that the request that asks for a blockchain address contains no personal information. This way, the address can’t be connected to the voter, and as such, their vote remains anonymous.

3.1.5 Vote Confirmation

This is not necessarily a need for the system to be functional, but rather an ambition of our team to deliver something more than the minimum. We want people to be able to see that their vote counted and that it was not tampered with. To do this, our website gives the voter their blockchain addresses after the voting transaction is done. When the voting period is over, the government or the election agency will open access to the “Election Results” page. On the page, the candidate pictures will be shown, and under the pictures will be the blockchain addresses that voted for that candidate. A voter can search for his address and confirm that their vote counted.

3.2 Blockchain
The blockchain implementation forms the core infrastructure of the VoteBlok. It leverages the decentralized and immutable nature of blockchain technology to ensure the security and transparency of the voting process. Ethereum's blockchain is utilized for its smart contract functionality, which allows for the execution of self-enforcing contracts.

3.2.1 Smart Contract Design
When designing the contract for VoteBlok, we focused on creating an efficient and secure solution for governmental elections. Oir goal was to optimize the contract’s performance while ensuring voter anonymity and preventing unauthorized access. To achieve this, we built a streamlined approach that carefully considered the specific needs of candidates and voters. By differentiating between these two roles, we were able to design compact data structures that minimized the storage costs and computation power, such as each candidate storing their own vote count in their own instances of a Candidate structs, eliminating the need to additional tables. To be able to prevent anyone with a blockchain address from casting a vote, we implemented a robust verification system to prioritize anonymity while ensuring a reliable and secure voting process. This system allows us to selectively authorize blockchain addresses to be able to cast a vote which is explained in more detail at 4.2.3 Participate in Election.

3.3 Front-End
The frontend of our system, “Voteblok” is a simple React JS application with 7 pages: “Main Page”, “Register for Voting”, “Registration Success”, “Election Login”, “Participate in Election”, “Voting Success”, and “Election Results”. The order in which we list the pages here and down below represents the common flow the voters will follow in order to vote and use the system. We will inspect each page on its own and talk about its functionality and its interactions with other components.

3.3.1 Main Page
The main page of the website does not serve any functionality in terms of voting. It summarizes the benefits of using an online blockchain voting system and provides a simple chart of the process steps. The user will be able to access the “Register For Voting”, “Participate in Election”, and “Election Results” pages through the navigation bar present at all times at the top.

3.3.2 Register For Voting
This page contains a form for accepting personal information. After the potential voter fills out the form and clicks submit, a “unique identifier” will be created and their information along with the unique identifier will be sent to the government API, which will determine if they are eligible. If they are, the page will receive a positive response, and the user will be redirected to the “Registration Success” page. If not, the page will not give a response.

3.3.3 Registration Success
The “unique identifier” generated for the voter during the “Register for Voting” page will be forwarded to this page upon navigation. The identifier will be displayed in bold, and the voter will be told to take note of it if they want to be able to vote.

3.3.4 Election Login
When the voting day comes, voters will be able to use their ssn and “unique identifier” to log in to the voting system. This will send a request to the government API and will see if the unique identifier and ssn are in the system. If so, they will log in and will be redirected to the “Participate in Election” page.

3.3.5 Participate in Election

On this page, there will be pictures of candidates, and a button below them that says “Vote for …”. Upon clicking any of these, there will be a series of API calls and at last the voting transaction will be sent, finalizing the vote. If all go accordingly, the voter will be sent to the “Voting Success” page.

3.3.6 Voting Success

After the voter successfully votes, they are redirected to this page. The navigation also includes the blockchain address they used in the vote transaction, and it is shown on the screen in bold big letters. They are also told to note their address if they wish to confirm their vote once the election is over.

3.3.7 Election Results

After the election process is complete, this page will be open to access, and it will show the total votes of each candidate, and the pictures of the candidates side by side. Under the pictures will be the blockchain addresses that voted for them. By searching through these addresses for their address provided on the “Voting Success” page, they can confirm that their vote was counted and was valid.

3.4 API and Database

The API and database design for VoteBlok system follows a secure and efficient approach. The APIs are developed using TypeScript and deployed to Firebase, utilizing an Express server.

The need for API endpoints arises from the distributed nature of the system, where different components, such as the front-end user interface, the government database, and the blockchain smart contract, need to interact seamlessly. By defining specific API endpoints, the system establishes a clear and secure pathway for information transformation between these components.

The primary objective behind the design of our API and database is to prioritize voting security by preventing unauthorized individuals or external entities from tampering with the elections. By implementing robust validation processes and tracking mechanisms via database, we effectively prevent multiple voting attempts, safeguarding the integrity of the electoral system.

4. Implementation

Naturally, we are not able to imitate the security measures of a government, its API, and its databases. We also do not possess the funding to fully create and live-test a blockchain system. And as such, our implementation will be slightly different than how we designed it for real-life use. Our implementation revolves around deploying an Ethereum Smart Contract locally using Hardhat, deploying a Google Firebase database along with an Express API server, and a React JS front-end that is run locally.

The reason that we don’t deploy our front end is that we don’t have live blockchain accounts that we can use for testing or even using the system. However, Hardhat provides 20 public accounts that are for test purposes, which come with enough ETH in them to make the voting transaction. By keeping the contract and front end local, we can simulate and test the voting process and ensure that it works. But in an ideal real scenario where a government would use our system, they would have no problem creating their own addresses and putting them in their API database to be distributed to voters. Our design allows this easy adaptability so that the project serves as a proof of concept from a scalable perspective.

4.1 Blockchain

The smart contract in VoteBlok is responsible for handling the registration of voters and the candidates, recording voters’ votes, and storing the necessary information related to the election. The contract is written in Solidity, the programming language used for Ethereum smart contracts.

The contract includes the following key functionalities:

1. Candidate Creation: This function allows the contract owner, typically the election

agency in this case the government, to create a new candidate by providing the candidate's name, age, image URL, and blockchain address. The contract owner is the only entity authorized to add candidates to the system.
2. Voter Creation: This function enables the contract owner to register a new voter by providing the voter's blockchain address. Only registered voters can cast their votes through the system, and the contract owner is responsible for registering voters.
3. Casting the Vote: When a voter wants to cast their vote, they send a transaction to the smart contract specifying the candidate's blockchain address they wish to vote for. Upon execution, the vote count for the chosen candidate is incremented, and the vote is recorded on the blockchain.
4. Data Retrieval: The smart contract provides several helper functions to retrieve information stored on the blockchain. These functions allow users to access data such as vote counts, vote confirmation, candidate details, and other relevant election-related information.

The smart contract serves as the core of the voting system, ensuring the integrity of the election process and preventing unauthorized vote casting or tampering with the recorded votes.

4.2 Front-End
Front end of this project is important not only for user interface and API interaction but also for actually using the smart contract, the piece that holds the whole election together. The Smart Contract is held inside the front-end files, in both raw and compiled form. Only our “Participate in Election” and the “Election Results” pages interact with the contract. They use the “ethers” node package to create an object instance of the contract with the signature rights of the provided blockchain address. This will be explained further in 4.2.3 and 4.2.4.

In section 3.3, the design and functionality of each front-end page were explained. In this section, we will further elaborate on how each page precisely accomplishes its task, detailing its interaction with other components, namely the smart contract, and API. Some of the pages hold no implementation value as they are static pages, and as such will not be mentioned in this section.

4.2.1 Register for Voting
The form on this page asks the user for a lot of personal information to make sure they are indeed who they claim they are and not an identity thief. The requested pieces of information are: First name, last name, address, phone number, birth date, and most importantly, the social security number. After the user presses submit, an algorithm that generates a random 12-character string is run, and a “unique identifier” is generated. Then the personal information and the unique identifier are sent to the “voter-register” endpoint in the API as a post request. If the API approves, the user is added as a voter. In a more secure and real-life environment. It would be beneficial to implement government-chosen encryption that encases this whole transaction and ensure no one can intercept and steal the personal information or unique identifier of the voter.

4.2.2 Election Login
On this page, the form for the user asks for two pieces of information. The social security number and the unique identifier of the voter. Once the voter fills in this information and presses submit, the page sends an API request to the “voter-login” endpoint to confirm if the voter is registered. If they are, a positive response is sent from the API, and the page is forwarded to the “Participate in Election” page, along with the ssn and unique identifier information.

4.2.3 Participate in Election

This page is the main flagship when it comes to the intricate workings of the voting system in the front end. Upon navigation to the page, an API request is sent to the “voter-login” page, to ensure that the navigation is valid. If the response is not positive, or there is no response, the user is kicked off back to the login page. If the response is positive, nothing happens. Once the user presses a button to vote, a sequence of events happens in order.

First, we do the confirmation again to ensure they are a voter, then we send a request to the “get-address” endpoint to get a blockchain address to do the voting transaction with. There is no personal information in this request, which is how we ensure that the address is not secretly associated with the voter's personal information. Once the response is back and is positive, an instance of the contract is created, using the signature of the contract owner. With this contract object, we call the “createVoter” function of the contract, passing the received address from the API as a parameter. This adds the address eligible for voting in the contract. Then, another instance of the contract is created, but this time with the address received from the API. Then using this contract, the function “vote” is called, passing the desired candidate’s address as a parameter. This created the transaction for the vote to be processed. And finally, an API request is sent to the “vote” endpoint with the ssn and unique identifier of the person, which tags the login information of the voter as having voted. This prevents the voter from logging in again and voting once more. Afterward, the user is redirected to the “Voting Success” screen.

4.2.4 Election Results

When navigated to the page, a contract instance with the owner’s signature is created, and the smart contract function “getAllVotersAndVotesInNames” is called. This returns all addresses that voted along with the name of the candidate they voted for. This list is processed and separated into each candidate. Then the addresses are displayed as a vertical list below their respective candidates.

4.3 API and Database

The first API, "/get-address," randomly returns an available blockchain address that has been previously created by the government without any matching of user identity. This approach ensures voter confidentiality, similar to a ballot in a real-life scenario. These addresses are stored in a NoSQL database with fields for "blockchain_address" and "is_available." Each time an address is issued, its corresponding "is_available" value is set to 0, preventing it from being assigned to another person in subsequent requests. By generating and issuing available blockchain addresses on the election day, the system ensures that only eligible voters can participate, preventing outsiders from voting using any blockchain address. The process is designed to restrict access to the blockchain addresses and restrict their availability to prevent unauthorized usage.

The second API, "/voter-register," allows eligible voters to register well in advance of the election date. All eligible voters are stored in a NoSQL Firebase database, with hashed social security numbers (SSNs), issued identifiers, and an "is_voted" value. Once a voter has cast their vote, the "is_voted" value is set to 1, preventing them from voting again. Additionally, this API ensures that the person attempting to register is eligible to vote, maintaining the integrity of the electoral process.

The third API, "/voter-login," facilitates logging into the system on the election day to cast a vote. It validates the inputted SSN and identifier against the "eligible-voters" collection in the database, confirming eligibility and ensuring that only authorized individuals can access the voting process.

The fourth API, "/vote," initiates the blockchain transaction that casts the vote. It triggers the necessary changes in the "eligible-voters" table to indicate that the person has voted, preventing multiple voting attempts.

5. Results and Analysis

As a result of our design and implementation, we have a blockchain election system that is functional and minimally secure in a non-malicious environment. Users can register to vote, use their unique identifiers to log in, cast their vote, see the results of the election, and confirm their participation. This on a large picture fulfills our goals when it comes to what we aimed to provide the voters. However, if the government or malicious parties really wanted to, they would be able to find weaknesses that can be attacked.

Our security measures in the front end and the API provide some scope of anonymity, confirmability, resistance to tampering, and resistance to duplicate voting. Following the implementation exactly as it is should not provide perfect protection. Several encryptions and further front-end precautions would be advisable before this can even be considered for real-life use. However, through trying to get this to be a functioning system, we have seen the problems blockchain proposes as an election tool. These are further explained in section 6.

All things said, we are happy with our results. We spent a lot of time thinking of ways to make it secure while keeping other attributes that make a voting system great. We hope that our concerns lead to other people figuring out ways to deal with them.

6. Issues

6.1 Scalability in terms of cost

In 2022, Ethereum went through a significant transition from being a proof of work to a proof of stake blockchain resulting in cheaper gas fees and better energy efficiency. However, it is important to consider that we have not seen enough data to consider the full impact of this transition since it has not been observed during a cyprocurrenct boom nor a large-scale event like a governmental election. In the context of organising an election where millions of people vote in the span of few days might cause a significant increase in gas fees due ti high volume of transactions. This increase in gas fees can present a significant expense to an election agency in this case to the government. Since there is a limited data available on the cost implications of such scenarios, it becomes challenging to make a cost comparison between a traditional election and a blockchain based voting system.

6.2 Voter Anonymity

In the realm of online activities, there is always a possibility of someone monitoring oir actions and tracking our online behavior. When it comes to blockchain technology, its complete transparency poses challenges for ensuring absolute voter anonymity in a blockchain-based voting system like VoteBlok. Every transaction on a blockchain is publicly visible, including the details about the sender and the receiver. This means that simply relying on the blockchain itself may not impose a guaranteed voter anonymity. Additionally, since not everyone with a blockchain address should be eligible to vote in a governmental election, there is a need to create a mechanism for the government to identify which addresses belong to citizens of the country. In terms of the design of the VoteBlok, randomly assigning pre-registered blockchain addresses to eligible voters is a good solution but this still may not guarantee the voter anonymity. If a pattern can be discovered in the assignment process of if the time when an addres is assigned is correlated with the information of when that address logs onto the platform , it could compromise the anonymity of the voter. It is essential to recognize that online voting inherently carries risks to voter anonymity. Guarding the voter privacy and the anonymity in a digital environment requires a comprehensive measures beyond the blockchain technology alone.

6.3 Data Protection

The method we have came up with to protect voter anonymity by assigning random blockchain addresses given their private information for their confirmation of the eligibility to vote in an election may raise some issues if not taken care of really tediously. One major one being that cyber attacks by malicious attackers. During an election period, millions of people will be registering to vote on the website, which requires them to input personal information, including their SSN. In this process, there is a risk of interception of the API calls between the system and the client, resulting in a potential data leak. This could lead to various issues, such as identity theft. When a system like VoteBlok is being built, there should be huge focus on the encryption and the safety of the API calls and the database.

7. Conclusions

VoteBlok emerges as a transformative solution to the challenges faced by traditional voting systems by harnessing the power of blockchain technology. By leveraging the properties of blockchain, such as decentralization, immutability, and transparency, VoteBlok establishes a first-of-its-kind governmental election system. Throughout this paper, we have presented the design and implementation of each component of the VoteBlok system. We have outlined the necessary information for the blockchain smart contract, the integration of the API, and the deployment of the front end for voter use. In the design of the VoteBlok, we have identified and addressed several key concerns related to eligible voter determination, preventing duplicate votes, concerns about the gas fees and voter anonymity, and providing a crucial feature of voter confirmation that the traditional elections do not offer. On top of that, VoteBlok not only offers advantages over traditional election systems, but also has a great chance to be the better approach to elections in terms of sustainability, and environment friendly. Although this paper provides a comprehensive overview of the design and implementation of VoteBlok, it is important to acknowledge that the system's effectiveness and scalability in terms of gas fees would need to be further evaluated through real-world testing. In conclusion, VoteBlok presents a promising solution for secure, transparent, and accessible governmental elections.

8. Lessons Learned

When we embarked on VoteBlok, none of us possessed the knowledge of how exactly blockchain operates, and the steps we needed to undertake to turn VoteBlok from an idea to a reality. As a team, this was a really great opportunity to dive deep into the blockchain and develop a decentralized application while letting ourselves explore a different branch of technology we have never dealt with. Along the way we came across many issues, one example being that every tutorial we have seen online was using a wallet provider such as MetaMask, and we did not want to rely on a 3rd party application which complicates the accessibility of the voting system. Difficulties like these forced us to think outside of the box and this created a difficult journey for the development of VoteBlok but a fruitful one we really appreciate.

9. References