Smart Poll: Blockchain Powered Voting on the Ethereum Virtual Machine

Timothy Coyne
timothycoyne@vt.edu
Virginia Tech

Elliot Jimeno
lemjimeno@vt.edu
Virginia Tech

ABSTRACT

Voter security is increasingly a point of hot contention in the United States, but also it requires the use of the anachronistic postal service or opting to physically be present at a polling station. Voting in today's society requires a level of trust that can pose a number of concerns for those who are casting their ballots.

Decentralized voting presents a compelling alternative to traditional voting systems due to its potential to enhance transparency, security, and accessibility in the electoral process. Unlike traditional voting, which often relies on centralized authorities and paper ballots, decentralized voting leverages blockchain technology to create a tamper-resistant, immutable ledger of votes. This not only reduces the risk of fraud and manipulation but also ensures that every vote is accurately recorded and counted. Furthermore, decentralized voting can be conducted electronically, allowing for greater convenience and accessibility, particularly for those unable to participate in physical polling locations. The elimination of intermediaries also reduces the likelihood of bias or external influence, fostering a more direct and inclusive democracy

In an effort to provide a more secure and reliable means of voting, smart contracts can provide a digital and decentralized manner of voting. By leveraging the use of blockchain within the election process, governments can have more peace of mind knowing that this technology provides upgraded capabilities when it comes to privacy and security.

Smart Poll emerges as an exemplary blockchain-based e-voting platform, characterized by its high efficiency, robust security measures, and innovative design. The platform's comprehensive nature, integrating front-end, middle-end, and back-end components, ensures a seamless, ready-to-deploy solution for electoral administration. The incorporation of One-Time Password (OTP) technology significantly enhances user authentication, reflecting our commitment to secure and user-friendly electoral processes. Additionally, the integration of advanced technologies like the IBFT 2.0 consensus mechanism and Besu software underpins the platform's resilience against network attacks and service disruptions, thereby ensuring uninterrupted and reliable voting experiences.

Virginia Tech, December 2023, Blacksburg, VA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the most recent election for the presidency in Belarus, an opposing candidate had officially lost in the results provided by the Belarusian authorities. However, the citizenry cried fowl to this widely recognized illegitimate election. In an effort to prove that the population overwhelmingly voted for the opposition candidate, over one million people posted pictures of their ballots online. In private and unchecked elections, conniving schemes can be easily accomplished when there is no public accountability. However, if the status quo for hosting elections were to transition into being held on open blockchain-powered platforms, maintaining election integrity would be ensured.

Historically, voting has been orchestrated in rather archaic and centralized manners. For example, final polling results in governmental elections are rather nontransparent and require public servants to be open and honest in ensuring that results are properly accounted for and published accordingly. When looking at various countries around the world including Belarus, Uzbekistan, and many more, we see manipulated voting results and opaque counting procedures.

By providing a platform that manages the entire polling process in a decentralized manner, one could ensure a fair and transparent election process due to blockchains being inherently trustless platforms. Not only this, but electronic voting can be accounted for as fast as blocks can be confirmed on-chain. When looking at recent elections in the United States, some states required days of waiting before a clear winner could be announced due to enormous volumes of mailed-in ballots. Internet voting through the EVM would remove this unnecessary waiting period.

Furthermore, a similar idea applies to more closed door voting procedures such as those that occur in shareholder meetings. By providing tokens through a smart contract, such as by utilizing ERC-20 or ERC-721 [1], a share-oriented voting system can be instantiated as the tokens represent voting power through owned shares. These tokens themselves can then be auctioned or gifted to whomever like on the open stock exchange market of today.

Current implementations suggest that voter information is kept as a hash in the blockchain and that results are encrypted until the election is over[2]. Voters should then be able to verify their casted vote after the election ends. Through the use of clever cryptography, we will be using Tessera which is a privacy-preserving transaction manager that integrates into Hyperledger Besu. This allows the chain to securely take in votes whilst preserving the underlying decentralized advantages of using an Ethereum Virtual Machine (EVM) chain. Current implementations lack the use of providing a one-time-password (OTP) option during registration. This can be seen as a hindrance to voters as unauthorized individuals could gain access to the system after the voter has signed in once. Our implementation will utilize OTP technology as to increase security of voters registering themselves and obtaining an authorized public and private key combination.

We seek to implement a variety of ideas including tokenization of voting power that can be used for single-use individual voting in governmental elections or shareholder style voting using the power of Ethereum's vibrant smart contract ecosystem. In order to avoid paying unnecessary fees, our implementation will be using its entirely own chain built off of Ethereum using Hyperledger Besu. By using an isolated chain, we gain a multitude of advantages including not having to worry about paying out gas fees to external users, deploying Proof of Authority (PoA), and removing noise from other unrelated third party contracts. Furthermore, voters will be able to cast their vote from anywhere at any time, thus making it a more cost-effective and time-saving solution.

In the United States, the use of traditional polling places and "trusted" middlemen can also serve as the source for several security concerns. Our system will ultimately render the use of a centralized third parties useless and therefore significantly reduce the possibility of election-tampering. Additionally, it would improve election result wait time, minimize the need for law enforcement monitoring, and eliminate the need for brick and mortar voting locations.

2 BACKGROUND

Blockchain is quickly becoming a hot-topic for many businesses when it comes to the security and trust of personal and/or sensitive information. After initially being introduced with the creation of the Bitcoin cryptocurrency, its rise has contributed to a multitude of advancements within the software sector. As mentioned in the introduction, one such area that can greatly benefit from this technology is elections or even any type of polling software. A decentralized approach to the election process would provide additional layers of security while making it more convenient for voters. The academic realm has contributed several proof-of-concept projects have been developed in order to validate the feasibility of such a system.

Decentralized voting systems have garnered significant attention in recent years, driven by a growing demand for secure, transparent, and efficient electoral processes. Traditional voting systems have faced challenges, including concerns about the integrity of election results, cumbersome paper-based procedures, and limited accessibility for remote or marginalized populations. In this context, blockchain technology has emerged as a powerful tool to address these issues. By leveraging the cryptographic principles and decentralized nature of blockchain, it becomes possible to revolutionize the way we conduct elections, ensuring the trustworthiness of the system and increasing voter participation.

Blockchain represents a step forward in modernizing electoral processes, making them more adaptable to the evolving digital landscape. The decentralized approach also decentralizes power, reducing reliance on central authorities and increasing the autonomy of individual voters. As we continue to witness the expansion of blockchain applications beyond the financial sector, its integration into voting systems stands as a testament to the technology's versatility and potential to bring about meaningful change in governance and public participation.

Threat model. The U.S. Vote Foundation argues that blockchain does not offer any security from cyber attacks. They further argue that like other online election architectures, a blockchain election is vulnerable to a long list of new threats that would leave it exposed to hacking and manipulation by anyone on the internet. Furthermore, they insist that such a platform would allow for anyone to vote without proper authorization in an undetectable manner. Our model seeks to build a platform that only accepts votes that have been verified by the relevant voting authority. For example, no vote will be counted that is nefariously sent onto the network unless it has been notorized.

Related works. There exists another implementation that prioritizes voter verification that utilizes smart contracts to built out its platform. This approach uses Solidity, which is a programming language designed to create smart contracts that run on Ethereum. However, it lacks the use of a one-time-password upon registration as well as experiences increased costs due to storing the encrypted vote in the blockchain at ballot-submission time. Our implementation will secure the registration by utilizing OTP issued from a voting authority.

Another for-profit implementation that has been used in official United States governmental elections is Voatz. During the 2018 midterm elections, West Virginia became the first state in the U.S. to allow select voters to cast their ballot on a mobile phone via the proprietary app. In a paper written by representatives of the Massachusetts Institute of Technology they discussed the security risks involved with the system including many vulnerabilities of which were confirmed by the vendor [3]. The paper goes into detail about the worrisome prospect that any network adversary can actively disrupt voting procedures through a number of ways including through a single successful attack on the server which could give the attacker full control over the election process. They also found that Voatz uses code from a variety of third party libraries and developers including Samsung's Knox libraries, headquartered in South Korea. From a national security standpoint, using foreign libraries can be a cause for concern when implementing a system that will determine who will hold office.

By implementing the use of an OTP option during voter registration, our system will have an additional layer of security. Furthermore, we can minimize overall cost by storing the encrypted vote in our gasless chain developed through Hyperledger Besu which supports such a feature. We will also be utilizing a modern secure transaction protocol baked into Besu which is discussed later on.

2.1 Design Overview

Our approach will consist of implementing an Angular frontend to go along with a Java backend while maintaining version control through the use of Git. The chain itself will be created through Hyperledger Besu and its smart contracts developed in Solidity.

Client side. Angular was chosen over other common frameworks as it provides cleaner code development and maintains more consistency for standard components such as routing and HTTP request handling. Client side technologies (as used by the Angular framework) will include HTML, CSS, and JavaScript together to provide the voting environment. Angular service objects will be used to streamline client-side processes such as profile authentication.

Server side. Server side technologies will be composed of a self-hosted server and another cloud based virtual machine. From here, Hyperledger Besu will be used alongside a helper program written in Java. This helper program will broadcast the transactions, or votes, to the Proof of Authority (PoA) nodes.

Tessera is an open-source private transaction manager developed under the Apache 2.0 license and written in Java. The primary application of Tessera is to serve as the engine for powering privacy-enabled Ethereum clients such as Besu[4]. Furthermore, our consensus algorithm of choice will be Istanbul Byzantine Fault Tolerant (IBFT 2.0) Proof of Authority. Together, these two technologies will allow for voters to privately make their choice whilst not compromising their privacy and the network's integrity.

Necessary Disclosure: Given the time constraints accompanied with developing such a complex system, it must be stated that our prototype will only build off of current implementations. This means that any security issues brought forth due to third party libraries and server/network insecurities are a result of time constraints that are not under our authority to manipulate. Starting from scratch would require a project composed of fully proprietary code and hardware to serve our project, a luxury we simply cannot meet. However, our enhanced version of existing implementations will still be able to fully demonstrate software implementations and improvements, and aim to follow appropriate guidelines such as using less third party libraries. As a result, we will not consider these aforementioned security risks when analyzing our end product and strictly evaluate it based on back-end performance.

More specifically, we are confident that our implementation will be generally safe but will be without any professional audit. Therefore, caution must be advised before bringing this technology to any production environment.

2.2 Project Timeline

For the Fall 2023 semester, our project's timeline was structured to ensure a systematic and efficient progression of our work. The timeline was divided into several key phases, each with specific goals and deadlines.

October 27 Deploy our simple Hyperledger Besu network on both self hosted and cloud-based servers. This will serve as the base for our blockchain network where we will build all of the necessary components such as the smart contracts which support the entire platform. The front-end is to also be hosted and ready for development at this time with the bare bones of the site also complete at this time.

November 3 Upgrade the network so that it contains the necessary privacy features such as by using the IBFT 2.0 consensus model and Tessera. Furthermore, ensure that the smart contract powering the chain has a rough draft completed at this time. The front-end site for users to interact with should also be navigable, but not functional.

November 17 The front-end is to be at least partially functional at this time. The smart contract should also have the ability to interact with the front-end. The main objective of this milestone is the intermediary program that is necessary for communication between the user interacting with the front-end and the network infrastructure.

December 8 The front-end is fully deployed and ready for individuals to register and vote on. The intermediary software is robust and able to exchange between the network and voters reliably. The network is secure and fully available and validating blocks. The project is complete at this time.

3 DESIGN

Our design implementation abstracts away all core mechanics into three separate categories which are front-end, middle-end, and back-end. For instance, the front-end will consist of the website interface that users will interact with, the middle-end acts as the reverse proxy connecting the chain itself with the front-end, and lastly the back-end which will compose of the smart contracts, chain, and nodes. As with most software, this division is necessary to ensure that individuals interacting with the system only access what is absolutely and critically necessary. For instance, a voter without technical skills may only want to access the front-end to register and cast their vote through the site without having to worry about what is behind it. However, it must be ensured that users are not locked into a single platform such as a computer browser. Access to the chain directly is inherently allowed since in order to cast one's vote through the website in a decentralized manner, they must hook directly into the chain via a node. Furthermore, users will be able to setup their own nodes to directly tie into the chain and forward anyone's votes to the smart contract all without relying on and third party once their token is authorized for voting.

3.1 Front-end

Users must first interact with a front facing internet site to register their wallet for voting under their name. This site will include all of the documentation necessary for someone to vote ranging in abilities from someone who simply wants to interact directly with the site to anyone who wishes to setup their own node and submit it through there. The stages for a user interacting with the front-end are quite straight forward and each must be completed before continuing.

Wallet Linkage

In order for a user to verify that they are the owner's of a specific wallet, they will need to submit the address of the one they choose to use alongside a signed message verifying that they are in control of the private keys. Otherwise, someone might make a typo when specifying their wallet or input one that they do not have control over for somebody else to use. Ensuring that the message they are required to sign is randomized each time ensures that the verification scheme's integrity is preserved.

Registration

After linking your wallet, the next step is to register how you would if it were done in-person. The user must provide information that the state or municipal government requires for registering or verifying one's right to vote. For example, a state such as New York may only require one's name and address, but another state such as Texas may require proof of identification such as a social security number or a picture of one's face alongside an identification card such as a driver's license. Once submitted, this will be sent to an authentication server which will make the determination of whether or not the individual should be allowed to proceed.

Voting

Once the verification process has completed, the user is ready to cast their ballot as soon as the election opens up. Speaking purely from the front-end, the individual will be presented with a list of candidates that they will select from. Upon selection, they will be prompted to affirm the transaction with their Meta-mask wallet and the transaction will be sent over to the back-end blockchain itself.

3.2 Middle-end

Regardless of how one chooses to vote with the chain, authentication is necessary to ensure that a wallet only receives a voting token if they are both who they say they are and have not already claimed a voting token. Municipalities are free to host their own authentication servers, which is required for functionality, that have their own stringent requirements. As mentioned before, a voter from Texas might be required to submit a picture of themselves to the server for a facial recognition verification to see if they are who they say they are. However, New York may only require textual information. Furthermore, every state may have their own idea of how best to secure the server and exactly how it will be incorporated into existing authentication frameworks. Each state's authentication server will be slightly different in their configurations and some may forward this information to other services depending on the individual circumstance. The front-end will ensure that the information from a user is sent to the correct server, and a common exchange protocol will be used to ensure mutual understanding between the protocol layers of what the status is of verification.

3.3 Back-end

The election manager smart contract is the central core of the voting process. It is responsible for keeping track of who has the right to vote and what the tallies are. Furthermore, the integrity of the contract's programming is of the utmost importance since a flaw in it could have devastating impacts on the entire process. However, with thoughtful design this should not be an issue whatsoever. The contract itself is deployed on top of the private PoA chain itself. This chain, through the same technology that powers Ethereum, will be the bedrock for powering the smart contract and its many abilities. However, the security of the mainline Ethereum network or Polygon is not to be underscored and should be considered in production environments.

The infrastructure that we are providing is quite unlike what other traditional voting platforms allow for. While the nodes that build the blocks themselves are permissioned and not open to the public, creating and maintaining a node that cannot vote on blocks will be available to anyone around the world. Voting has consistently sparked grassroots movements across the country for getting people involved in the process. For example, setting up sites to get people registered to vote has become wildly popular in recent years. Our system will allow enthusiastic individuals to host their very own nodes which connect directly into the network thus providing a gateway to transact privately and allow others to cast their votes without overloading a small number of nodes. By increasing the number of nodes that are supporting the chain, the throughput to handle a large number of voters increases as well.

One concern for launching nodes on the network will be the computational requirements for doing so. However, managing such a network is not entirely out of the question for even consumer hardware due to the fact that it simply isn't all that strenuous. For example, Besu's system requirements are simply thirty-two gigabytes of RAM and an SSD with sufficient storage. However, these requirements signify data storage requirements for the main Ethereum chain and not a smaller private chain. Due to the fact our chain would only be running around the time of the election, it's not realistic for storage concerns to be a problem. Later on in our elevation section, we will discuss more specifically the exact scalability parameters of the network.

Smart Contract

Written in Solidity, the voting contract is the core of the entire voting process on our chain. It is responsible for both giving and receiving voting tokens. Due to the complexity of verifying someone's identity on a state-by-state basis, the current process of authorizing and sharing a voting token is managed by the middle-ware who has operator access to the contract. In other words, the middle-ware has the ability to send a token to a given address, but only one token can still be sent to any given address.

The contract operates on the ERC-721 standard which is a published standard by the Ethereum foundation. This provides a robust bedrock for building our platform, since ERC-721 is wildly popular and well tested in the wild.

4 IMPLEMENTATION

Due to the variety of tools developed, we used a variety of technologies to build our product. In constructing our platform, we employed a diverse range of technologies, each meticulously chosen to align with the specific functionalities and requirements of the various components of our product. This approach mirrors the structure we outlined in the design section, where we categorized the development into three distinct yet interconnected segments: the front-end, the middle-end, and the back-end. Every choice was made to ensure that our product could perform at a scale necessary for looking professional and able to handle the task of accepting millions of votes.

4.1 Front-end

For full deployment, we would need to utilize a load balancing solution for millions or even just thousands of users to ensure that everything remains online. Digitalocean has an excellent platform for building this up at a reasonable price, and it would have been our go-to for demonstrating enterprise readiness. Furthermore, we would use nginx for actually running the website and making it accessible to the world. It is a very mature technology stack at this point and will do nicely.

In order to interact with the Ethereum blockchain network over HTTP, Web3.js will be integrated into the Angular-facing front end. Once web3 is installed and imported into the project with the help of npm (Node Package Manager), users will be able to connect to the crypto-wallets installed on their machines and access the global web3 API through the associated *window.ethereum* object. This API allows websites to request user login, load data from blockchains the user has a connection to, suggest the user sign messages and transactions, and track the status of local wallets.

Wallet Linkage

Upon pressing the 'connect wallet' button in the header, a third party modal will appear prompting the user to connect their wallet to the application. This will then be used to store, receive, and deliver voting tokens at election time. The following code snippet shows how the 'connect wallet' button-click event is binded to the *connectWallet()* function and how the function triggers the third party modal to appear.

```
<div class="connect" (click)="connectWallet()">
    Connect wallet
</div>

async connectWallet() {
    if (window.ethereum) {
      window.web3 = new Web3(window.ethereum);
      await window.web3.eth.requestAccounts();
    } else {
      // Non-Ethereum browser detected.
    }
}
```

If the user does not have a crypto-wallet installed on their active browser, they will be prompted to install one. If the window detects one or more crypto-wallets, they will be prompted to connect (or create) one and ultimately sign into our application with their unique public identifier, or address.

Outside of the web3 library, several different components will be created in order to modularize and simplify the creation of the front end. Things such as the header, footer, poll-containers, candidate-profiles, and more will be organized into these. Additionally, service files will be used in order to streamline and address the principle of separation of concerns. This will help significantly when connecting the frontend with the middle-end, before eventually bridging it with the backend. The goal is to make the user-experience as straightforward and self-explanatory as possible, while also maintaining and upholding proper design practices. Several user tests can be performed in order to receive feedback and ultimately measure the user experience into something that we can maximize with the outlined final product.

Registration

Our authentication server is to be written in Java with the ability to accept HTTP requests from the front-end site containing all of the information that user has provided for authentication. This data is then crosschecked against a list of known individuals and a result is returned back. For our example, our authentication server will store names in a simple String array but could easily be connected to any other type of database. We would recommend something like MariaDB or a NoSQL database depending on the interest of the developer. Because every municipality uses its own technology stack, our open authentication platform will allow any to build on our platform. Below is the protocol used for communication between the front-end and authentication server also known as Client to Server communication.

4.1.1 Authenticate User. This is the request to the authentication server with the information that may be needed such as first, middle, and last names, age, social security number, and address.

```

  "request": "authenticate",
  "data": {
      "firstName": "John",
      "middleName": "Fruit",
      "lastName": "Appleseed",
      "age": 22,
      "ssn": "123456789",
      "address": "123 Wayland Street, New York, NY"
  }

```

Listing 1: Authenticate JSON POST Request

Using the POST/authenticate request, our Angular application is able to communicate with the middle ware and send the voter's registration information to be received, reviewed, and returned with an appropriate response. Depending on the response, the front end will display a success or error message to the user notifying them of the status of their registration. Below is the typescript code snippet that sends the registration form-data to the middle ware using Angular's HttpClient module.

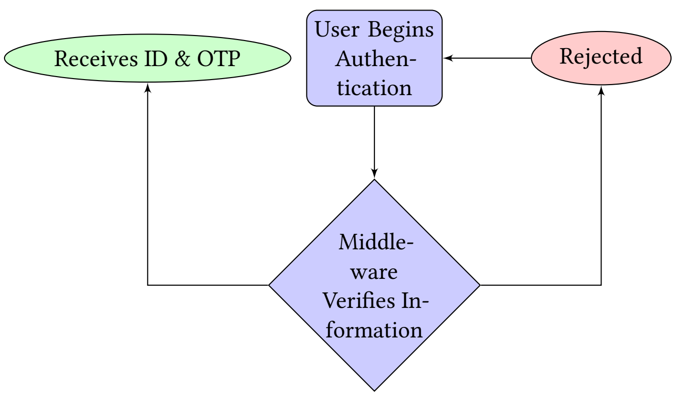
```
submitRegistration(registrationForm: FormGroup): Observable<Object> {
    const data = {
      request: "authenticate",
      data: {
        firstName: registrationForm.value.firstName,
        middleName: registrationForm.value.middleName
        lastName: registrationForm.value.lastName,
        age: registrationForm.value.age,
        ssn: registrationForm.value.ssn,
        address: registrationForm.value.address
      }
    }
   return this.http.post('http://votechain.cactuspowered.net:20001/',
    JSON.stringify(data));
}
```

4.1.2 Authentication Response. This is the response the middle-end will provide once the user has been verified.

```
"response": true,
"data": {
    "id": 22,
    "otp": "ba4bKL"
}
```

Listing 2: Authenticate JSON Response

At this point the user will be given their unique ID and OTP which are both necessary for completing the next step in obtaining a ballot. Below is a diagram of the entire process showing the outcomes and each step.



Voting

The voting procedure within the front end environment can be split into two separate procedures. The first one is to receive your token, or in voting terms, your ballot. This is done by sending a specific POST request to the middle-end. If the OTP has been used within five minutes of it being generated from registration, the middle-end will interact with the blockchain directly through one of our Nodes to mint a token just for that individual.

4.1.3 Request Token. When a user is ready to obtain their voting token, they must provide the id and OTP provided in the authentication step. Note that some very long fields such as the transaction hash and Ethereum address have been truncated for formatting purposes.

```
"request": "get_token",
"data": {
    "id": 22,
    "otp": "ba4bKL",
    "address": "0x5775eE7E629b94BfC1D..."
}
```

Listing 3: Request to obtain a voting token

4.1.4 Token Received. If a user was successfully verified by the middle-end, they will be given a confirmation that their token has been successfully minted on-chain. The JSON represents the request that must be sent to the middle-end, and the following JavaScript is the code uses by the front-end to execute this request.

```
"response": true,
"data": {
    "id": id,
    "txhash": "0x8a19a01adb9ab07d37f...",
}
```

Listing 4: Voting token has been minted in the listed transaction

```
getToken(id: string, otp: string) {
    const data = {
      request: "get_token",
      data: {
        id: id,
        otp: otp,
        address: environment.accounts[0]
      }
    }
    return
   this.http.post('http://votechain.cactuspowered.net:20001/',
    JSON.stringify(data));
}
```

It is at this point that individual no longer has any need to communicate with the middle-end. Its sole purpose of verifying one's identity and minting the token has been completed at this time for them. Once the user has received their token, they will then gain the right to vote in whatever election it belongs to.If using the front-end, they can either decide to cast their vote at that moment or wait until another time which is more convenient for them. If they wait, it will be unnecessary to authenticate all over again since the token has already been awarded to their account. They will be redirected to a page of candidates which will be able to activate MetaMask through Web3.js and build a contract interaction transaction which will send over the vote. It is worth noting that this transaction would normally cost a more than minimal transaction fee if running on the main Ethereum network, but our gasless chain allows for free voting once a token has been obtained.

We note that the *sendTransaction* function is not required to be ran by the front-end, and anybody with Node access, which will be anybody with a properly configured MetaMask, can execute this transaction another way. The front-end does not lock the user into its platform and they are free to explore alternatives built by the community.

```
async sendTransaction(candidate: string, tokenId: string) {
    if(this.isConnected()) {
      try {
        const userAddress = environment.accounts[0];
       const contractAddress = environment.contract_address;
```

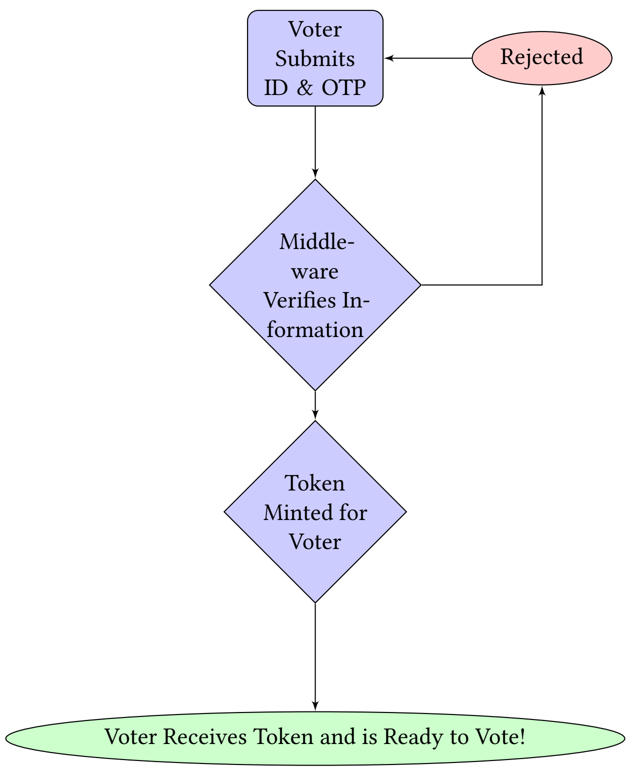
```
        // Create a contract instance
        const votingContract = new
        window.web3.eth.Contract(contractABI, contractAddress);

        // Encode the function call
        const encodedData = votingContract.methods.vote(
            candidate, tokenId
        ).encodeABI();

        const transactionObject = {
          from: userAddress,
          to: contractAddress,
          gasPrice: '0',
          data: encodedData
        }

        window.web3.eth.sendTransaction(transactionObject)
        .then((res: any) => console.log(res))
          } catch (error) {
            console.error('Error casting vote:', error)
          }
    }
  }
```

Below is another flowchart which demonstrates what a registered user must do once they receive their ID and OTP.



4.2 Middle-end

The middle-end primarily consists of the authentication server. It is to be written in Java with a multi-threaded back-end and its services would need to be geo-distributed to ensure that voters are not reliant on a limited or even single number of servers. Java provides an excellent ecosystem for building safe programs that interact seamlessly with low-level networking stacks. The experience of one of our developers in the language also makes it a compelling language for building such a tool in a limited amount of time. This program would consist of a front-facing server socket which waits for an incoming connection and sends it off to a worker in a thread pool consisting of 512 threads. Thread pools allow for quick execution in multi-threaded environments because spawning and destroying threads is a costly endeavour on a system. Therefore, our middle-end will be quite competent at serving voters.

4.3 Back-end

Our choice of Hyperledger Besu allows us to build a blockchain in a quick and efficient manner without resorting to building something totally from the ground-up like other chains had to such as Bitcoin. The chain utilizes PoA to build blocks, but its a very similar process to the main Ethereum chain when it comes to spreading that information around to other nodes. In order to ensure that the network is robust, municipalities will need to run their own validation nodes which must get authorized by existing PoA nodes. Our network will consist of a number of seed nodes which will be given validation permission automatically at the genesis of the network. Not only will they be backup nodes for any peer looking for a connection into the network, but they will also have operator permissions on the smart contract to mint voting tokens in coordination with the authentication servers.

Smart Contract

The voting smart contract is the core of our entire voting process. While many things are of course offloaded to other parts of the project, the core methodology that keeps the numbers accounted for and prevents individuals from voting for multiple candidates must be partially bestowed upon an authoritative middle-end connected to government servers. Meanwhile, the contract itself is wholly capable at preventing a given address from breaking any standard rules of an election such as only being able to cast one's vote a single time. This mechanism of preventing an address from voting more than once is simple in that it only uses Solidity's mapping feature, but its a powerful security feature. If tokens can only be provided through the middle-end, cannot be transferred to other users, can only be used to vote once before being permanently burned, and only one can be received, then it makes for a well secured contract.

For some more background on ERC-721, it is a standard used for implementing popular Non-Fungable-Tokens (NFTs)[5]. Extremely popular and regularly traded on platforms like OpenSea, there is a clear technological precedent for building up infrastructure on this framework. It allows for a number of functionalities like minting, burning, transferring, and more. With this, it's possible for someone to build whatever comes to their imagination such as implementing a tax whenever someone transfers an NFT that can then be donated, or in our case building security guardrails that ensure the NFT platform can be used for safe and fair elections. The smart contract, serving as the backbone of our voting process, exemplifies the convergence of security, reliability, and innovation in our project. Its design, rooted in Solidity's mapping feature, elegantly addresses the critical requirement of ensuring that each voter casts only one vote.

Security Considerations

Like any software, understanding the security implications of our multi-focal platform is critical for building out a robust voting platform. Therefore, analyzing the front, middle, and back-ends will be paramount in understanding exactly what risks must be mitigated or eliminated. To start, the site itself will require little computation and data transfer on its end since it will be almost an entirely client-side application outside of feeding the actual bytes to the internet connected user. As long as the site is geodistributed to mitigate DDoS attacks and to expand access to many different regions, it will be very accessible. Additionally, having a server physically closer to an individual improves the user experience through lower latency and ideally quicker responses since anyone located outside of their region should be redirected to another server.

The authentication servers serve as one of the most important parts of the entire chain. Because they control who is allowed to receive tokens and who is not, maintaining the security of these servers is crucial. This project does not look for a decentralized way of verifying someone's identity, but something like this could be useful for better integration with Ethereum and removing this point of failure.

Smart contract technology has been used for years on the main Ethereum chain. Because of this, they are a tried and true method of deploying software. The major security concerns for it are no different than any other piece of software written outside of a blockchain. Ours ensured several key rules that must be upheld to ensure that voters are not able to unfairly tamper with the election. These are all held within the context of ERC-721 through the various methods it provides. Here are some of the methods below and associated rules with each:

transferFrom Called when a voting token is moved from one wallet to another. Users may not move their token to any other user whatsoever. However, they are able to burn the token which allows them to send it to a burner address that is inaccessible by anyone without unfathomable amounts of computing power that cannot be achieved with current technology. Upon doing this, they will not be able to request a new token.

balanceOf Obtains the number of tokens that the account owns which must always be one or zero.

ownerOf Given a tokenID, this presents the owning address of it. Freely accessible.

vote Transfers the token into the burner address and increases the tally amount for the candidate that was chosen. One account can only vote once since it will be marked as having voted.

mint Mints a voting token and sends it to the specified address as long as it does not currently own a token and never has.

Ethereum is an actively developed chain, and therefore clients that run it like Besu are rigorously maintained and patched for security vulnerabilities that are discovered. As a result, the blockchain software itself is very secure and is admitted to being enterprise-ready by Besu.

While a malicious non-voting node would be unable to directly damage the chain outside of various DDoS attacks which Besu attempts to mitigate, voting nodes under PoA must be well guarded such that they are well secured. If a large number of voting nodes were to be held under the control of an adversary, then the entire chain would be at risk of collapsing and making the election invalid. This is no different than the risks associated with more centralized voting schemes and in-person paper ones: polling stations and servers must be well-secured. The best way to mitigate this is to ensure that municipalities run many voting nodes such that a failure of a small number would have no effect on maintaining the integrity of the network. As more nodes are connected to the network, its reliability increases.

As mentioned before, individuals are free to host their own non-voting nodes for interacting with the chain. Since these nodes have no impact on producing blocks, erroneous behavior from them will not impact the integrity of the network. The main concern with these would be if someone tried to connect to a malicious node that was advertised to them which could be accomplished in and unending amounts of ways such as through fishing, false advertising, and DNS poisoning. However, this can be mitigated through using our front-end website which ensures that the correct node is being accessed. If someone wants to build their own node and use that for voting, then they are likely a power user who is capable of maintaining the security of their ballot on their own and is outside the scope of our concerns.

5 EVALUATION

Our evaluation of the platform is a combination of several factors which are the innovative features provided that are non-existent in other platforms, the security protocols and considerations in-place, and the flexibility of the design. The combination of these three items shines a light on Smart Poll which reveals a highly efficient, secure, flexible, and innovative platform.

We first begin with the innovations we developed. To start, our system is a fully packaged platform. The combination of our front-end, middle-end, and back-end which are all ready to be used allows anybody looking to host an election to hit the ground running. In these, we wanted to ensure a modern and practical experience for election administrators and users alike. One way we did this was implementing OTP technology into the core of the authentication scheme. When a user successfully registers, they must use that password within five minutes in order to authenticate. With the advent of two factor authentication, this kind of temporary password is passively popular. Of course, this could be further built upon with another form of authentication such as 2FA through hardware security keys or a phone number that would further ensure that the person is who they say they are from registration.

Security was of course paramount in our development process. We ensured that the latest innovations such as IBFT 2.0 consensus was used to build out our custom chain. More specifically, if more than one third of validator nodes go offline, then the entire network will halt to prevent any attack from manipulating the chain.

Attacks directly on the network itself is not the only area of concern when it comes to safeguarding Smart Poll. Denial-of-service (DoS) attacks are an incredibly common security concern in Ethereum.

By increasing the surface area of the network to allow for anybody to setup a Node and connect, we will increase the throughput for transactions and thus reduce the risk of packet flooding. Not only this, but the Besu software is well matured at this point and has a number of security features in-place to prevent attacks. We also consider the prospect of more legitimate attacks through simply sending Ethereum around. Due to the network being gasless, it would be free for anybody with any amount of Ethereum on the Smart Pool network to send it around as much as they can to create congestion. The mainline Ethereum network has a disincentive for somebody to do this since transaction fees for sending Ethereum are measured in dollars and such an endeavour would not be cheap. Therefore, restricting the availability of Ethereum on our chain would remove the possibility of somebody spamming transactions. In our PoA chain, Ethereum can only be generated from the genesis file and distributed by the accounts given the original amount. By burning the entirety of all the Ethereum generated or not generating any, it would be impossible for anybody to obtain any of the currency. As a result, our gasless network can eat its cake and have it too by providing users with fast and free transaction while not having to deal with spam attacks on the network.

Excitingly, our platform represents a massive shift forward in transparency when it comes to electronic voting platforms. Blockchain technology is at its core meant to be a replacement to the traditional model of proprietary and closed-source block boxes that have dominated the commercial software industry. By developing on Ethereum, we have paralleled this core value of the technology by making all of our software open-source including the smart contract, middle-end, and front-end stacks. This allows for anybody to audit and built upon our platform in any way that they desire without paying any royalties. In fact, deploying our solution is completely free in every way since Ethereum is a permissionless network and our open-source software by definition requires no payment to use.

Aside from the software itself being written on open platforms, integration with them is not guarded and is free for anyone to use. For example, our specification for authenticating with the middleware is fully open and documented comprehensive detail in this paper. The beauty of this is that, like the manifesto of blockchain, our custom software is also permissionless and anybody can send packets to it without needing special authorization. This provides a unique opportunity for the community participating in a given election to build their own solutions in helping people vote. For instance, the state itself would not have to build everything themselves and independent developers would be able to build apps and other services to help others vote. Currently, there are a plethora of other sites available which seek to help others register to vote, and there exists with our platform a similar opportunity for grassroots movements to catapult into developing new and other innovative additions to what we present.

Aside from the front and middle-ends being fully open, the chain itself which builds the blocks and executes the smart contract is open to the world for interaction. While validator Nodes would have to be restricted to trusted servers that would likely be run by government institutions or other well secured organizations, non-validating Nodes would be free for anybody to host. This provides a glance at a exciting future where enthusiastic individuals or groups can host Nodes for anybody to connect with and use to facilitate their transactions.

Scalability is a critical facet of developing any application and is of particular concern when it comes to decentralized ones. By creating an open network that anybody can participate in, we ensure that connectivity can be spread around in a grassroots style. Furthermore, by utilizing the same protcol as Ethereum we allow for an ecosystem where scalability is measured by the combination of blocktimes, block gas limits, and number of nodes. Our network operates on two second block times which is immensely fast compared to Ethereum's 12 seconds. The produces blocks are also immenseley different than those of Ethereum with ours providing a whopping '0x1fffffffffffff' gas per block. This immense scalability provides our platform with the ability to tackle even national elections. Below is a chart representing the various considerations between our chain and mainline Ethereum. It is worth considering that our implementation is workable on any Ethereum-based chain, but our IBFT network is gasless and thus does not require any gas to submit transactions.

Chain	Block Time	Gas Limit	Scalability	Storage
Ethereum	12s	0x1c9c380	High	High
Smart Pool	2s	0x1fffffffffffff	Very High	Very High

We can see that while our network implementation would demand a vastly higher storage requirement per generated block, its superior gas limit and block creation time allows for much faster confirmation time of transactions and more of them per block. This is incredibly important when looking at current voting electronic voting solutions. It's important that a vote transaction is confirmed in a reasonable amount of time as current centralized solutions do not require immense amounts of time to count your vote.

6 RELATED WORK

This section aims to provide a comprehensive overview of the current state of blockchain-based voting systems, drawing on the insights and findings from key researchers and practitioners in the field.

Jae-Geun Song, Sung-Jun Moon, and Ju-Wook Jang, presented an in-depth study aimed at addressing the inherent challenges in implementing scalable and anonymous voting systems using Ethereum blockchain technology [6]. They focused on three major bottlenecks: the division problem in encrypted voting values for anonymity, the large time complexity in tallying, and the issues arising from no votes by registered voters.

They proposed innovative algorithmic solutions to overcome these challenges and introduced a new encryption scheme for voting values, altering the traditional approach to encrypt and broadcast votes, which significantly enhances the efficiency of the tallying process. This new method not only simplified the tallying process but also reduced the computational resources required, thereby addressing the issue of scalability.

Their paper includes an extensive performance evaluation, comparing the proposed system with existing solutions. The evaluation demonstrates a substantial reduction in gas consumption – a critical factor in Ethereum blockchain transactions – and improvements in time complexity. The study concludes that the proposed approach offers a more scalable and efficient solution for implementing anonymous voting systems on the Ethereum blockchain, suggesting a significant step forward in the application of blockchain technology in the electoral processes.

The possibility of a decentralized voting system using the Ethereum blockchain and smart contracts was also explored by Mulla Almas in "E-Voting System using Smart Contracts and Blockchain" [7]. The author studied the challenges regarding privacy, correctness, and integrity. A prototype was implemented as a proof of concept, including studies on social and environmental aspects of electronic voting. They identified areas for improvement in blockchain-based voting systems, suggesting the use of a private blockchain for implementation, and concludes that further work is needed to employ this technology in crucial fields like voting. Similar to Smart Poll, our implementation does uses its own private chain.

Similar to our implementation, there exists such a system provides voter anonymity by keeping the voter information as a hash in the blockchain [8]. It also provides fairness by keeping the casted vote encrypted till the ending time of the election. After ending time, the voter can verify their casted vote, ensuring verifiability. To test the protocol, the authors deployed in on Ethereum 2.0, a blockchain platform that uses Solidity as a programming language to create smart contracts. The adoption of smart contracts provides a safe means for performing voter verification, ensuring the correctness of voting results, making the counting system public, and protecting against fraudulent activities. The system's performance was analyzed based on security and gas costs. It showed improvement in terms of security characteristics and the related cost for the necessary infrastructure.

7 CONCLUSION

In conclusion, our evaluation of Smart Poll, a blockchain-based e-voting platform, reveals it as a highly efficient, secure, and innovative solution in the realm of decentralized voting systems. This platform stands out due to its comprehensive package, combining front-end, middle-end, and back-end components, all ready for immediate deployment. This integration ensures a modern and practical experience for both election administrators and users. A key innovation in our system is the incorporation of One-Time Password (OTP) technology into the authentication scheme, enhancing security and user verification processes.

Security has been a paramount consideration in the development of Smart Poll. We have integrated cutting-edge innovations like the IBFT 2.0 consensus mechanism, ensuring robustness against network attacks and maintaining system integrity even if a significant number of validator nodes go offline. Moreover, our approach to expanding network surface area and employing mature Besu software fortifies the platform against common threats like Denial-of-Service (DoS) attacks. The gasless nature of our network also addresses potential spamming issues, adding another layer of security.

Furthermore, Smart Poll marks a significant advancement in transparency for electronic voting platforms. By developing on an open-source framework and making our software, including smart contracts, accessible for audit and further development, we align with the core values of blockchain technology. Our system's open and permissionless nature allows for widespread integration and community-driven innovations, offering opportunities for grassroots movements to contribute to the voting process. The scalability of our platform, with its fast block times and immense block gas limits, ensures that it can handle large-scale electoral processes, positioning Smart Poll as a pioneering solution in decentralized voting.

7.1 Looking Forward

While our implementation was full of innovative technology and provides a robust platform for hosting elections, there are some areas we would like to have invested more time into. The privacy manager Tessera is a privacy preserving framework that groups of Nodes can utilize to provide a private transaction layer for participating Nodes. For the purposes of our demo, this has not been integrated with the limited time to build up our platform. Privacy in elections is key in ensuring that every vote can be counted whilst preserving the privacy of the citizen, so this could be an exciting future research area.

Another avenue of potential future exploration would be the incorporation of zero knowledge proofs. This is an up and coming area of Computer Science that can provide better anonymity for voters. More specifically, a zero knowledge proof system where it can be proven that somebody has executed the vote method of the smart contract with a valid token without identifying them would be a huge step forward. Currently, token ids can be traced back to the authenticated user which means something would have to change to provide true anonymity. Protocols like Monero utilize a vast array of privacy tools to preserve transactions completely, so perhaps future research could develop a new type of network that moves away from Ethereum that combines the privacy of these coins with the functionality of smart-contract chains.

The last area of potential improvement would be in the middleware. It is a generally centralized process that involves communicating with a server operating under the guidance of an authoritative entity such as a state government's voting office. It alone has permission to mint tokens and verify individuals. This does present a weak point for attackers to penetrate and destabilize the election. So long as servers are secure this is not an issue, but a future where authentication could somehow be built into a permissionless system would be an exciting prospect. Perhaps a system of zero knowledge proofs could be built such that one could verify their identity without revealing exactly who they are and where they are authenticating from.

REFERENCES

[1] Buterin Vogelsteller. Erc-20: Token standard. https://ethereum.org, 2015.
[2] Islam Ahamed Alvi, Uddin. Dvtchain: A blockchain-based decentralized mechanism to ensure the security of digital voting system voting system. Journal of King Saud University - Computer and Information Sciences, 2022.
[3] Weitzner Specter, Koppel. The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in u.s. federal elections. *Massachusetts Institute of Technology*, 2020.
[4] Tessera Maintainers. Consensys tessera. https://docs.tessera.consensys.net/, 2023.
[5] Opensea rankings. https://opensea.io/rankings, 2023.
[6] Sung-Jun Moon Jae-Geun Song and Ju-Wook Jang. A scalable implementation of anonymous voting over ethereum blockchain. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8229461/, 2021.
[7] Mulla Almas. E-voting system using smart contracts and blockchain. https://papers.ssrn.com/sol3/papers.cfm?abstract$_id$ = 4228522, 2022.
[8] Syada Tasmia Alvi et al. Dvtchain: A blockchain-based decentralized mechanism to ensure the security of digital voting system voting system. 2022.