# EVO: An E-Voting System using Blockchain

Vishnu Narayanan S R, Aswin R,
Ponnu G Vijay, Thanush Ram S
UG Scholar: Department of Computer Science
College of Engineering, Perumon
Kollam, India

Devi Dath
Assistant Professor: Department of Computer Science
College of Engineering, Perumon
Kollam, India

***Abstract:*** **Being a part of the largest democracy, one of our major challenges is to choose the right government. But still many adult citizens don't go to cast their votes. They may be out of station from the voting booths or might be fed up with the long queues in voting booths and many even believe that their vote doesn't count because of the unfair election system. Vote rigging, hacking of EVM (Electronic Voting Machine), election manipulation, and polling booth capturing are the major issues in the current voting system. And also, during these pandemic times it will be very hard for people to vote through the conventional voting systems. So we are trying to build a solution to this problem through this project.**

**Blockchain is said to be an emerging, decentralized, and distributed technology that promises to enhance different aspects of many industries. Through this project, we aim to implement the application of blockchain as a service to build a distributed electronic voting system while providing a solution to eliminate all disadvantages of conventional elections.**

***Keywords—E Voting; Ethereum; Blockchain***

## I. INTRODUCTION

Individuals have always regarded voting as the main means by which they express their views on divisive topics and debates over the years. It is a civic practice that allows people to demonstrate their official opposition to a ballot question, a nominee nomination, a political party, and other issues.

The well-known client-server architecture is used in today's E-Voting systems: A trusted third party is in charge of the server and the voting results, as well as the ownership and fairness of the votes. Unfortunately, voting is highly reliant on the election's organizers' confidence. Several accusations have been made against the electorate in recent decades, they are:

1) Inadequate data integrity and security protection.
2) There is only one point of failure.
3) Lack of reliable transaction validation protocols and centralized control.
4) A mysterious runtime environment.
5) The server is running unidentified business law.

Blockchain, on the other hand, is a modern technology that ensures data immutability by using cryptographic functions and consensus algorithms and protocols to provide network decentralization with no single point of failure. The Ethereum blockchain[1] is an open-source distributed computing framework with a Turing-complete scripting language that allows software engineers to create decentralized applications (DApps) that take advantage of the blockchain technology's distribution property. Therefore, DApps will have the following blockchain features:

1) Data integrity is essential.
2) Consensus processes have decentralized oversight and confirmation.
3) A run-time environment that is transparent.
4) In the run-time climate, public business laws are active.
5) A high level of availability.

Blockchain is growing in popularity in a variety of industries, including telecommunications.

We propose a decentralized online voting platform based on blockchain technology in this paper, with the goal of addressing the confidence issues posed by traditional E-voting systems. This scheme employs a novel method for validating and authenticating registered voters.

The main contributions of this solution include:(1) Immutability and data integrity of voting data, (2) robustness and reliability of the voting system, (3) decentralization of voter registration and validation mechanisms, (4) transparency, clarity, and determinism of the voting environment, (5) public visualization of smart contracts votes, (6) restricting each voter to a single vote per valid Mobile Station International.

## II. BACKGROUND ON BLOCKCHAIN TECHNOLOGY AND ETHEREUM

Blockchain is a shared decentralized network of replicas spread across several nodes at the same time. There is no central authority in charge of controlling and preserving the transaction ledger in blockchain. A consensus process among the validating nodes determines the validity of the ledger's edition. The use of blockchain technology allows for safe data integrity confirmation of transactions. Bitcoin, for example, was Satoshi Nakamoto's first Blockchain-based application [2].

Ethereum blockchain, on the other hand, is an open-source, distributed, and decentralized computing platform that runs smart contracts. Not only for a digital currency, but also for applications, it is being designed to allow decentralization. It's done by running a Turing-complete scripting language on a virtual machine (Ethereum Virtual Machine, EVM). Unlike Bitcoin, which only considers Boolean evaluations of spending conditions, EVM is more akin to a general-purpose computer that simulates the capabilities of a Turing machine. In the blockchain, altering the state of a contract necessitates transaction fees, which are priced in Ether. Ether is considered as the fuel for operating the distributed application platform.

### *A. Account Types in Ethereum*

There are two types of accounts in Ethereum:

1) Externally Owned Accounts (EOA): A wallet address identifies an account, which is managed by a private key. From this account, the holder of this private key may pass ether and sign transactions. EAOs are user accounts that are linked to a pair of unique cryptographic keys that are generated when the account is created. The public key is used to refer to

the account and is also known as the EOA address, while the private key is used to sign transactions before they are executed on the network to prove their validity. EOAs have Ether cryptocurrency balances in their accounts [3].

2) Smart Contract: A smart contract is a cryptocurrency account that is run by its own code. It is the central framework and key building block of any DApp and is considered an autonomous agent performed by the EVM. The EVM will take care of running this code once it is deployed on the blockchain as long as the requirements are met. It's worth noting that, once deployed to the blockchain network, smart contracts can be visited and viewed publicly via their address, along with all of their related transactions (to address, from address, timestamp, etc...).

Triggering functions in the smart contract can be performed from any account as long as the following two conditions are met: 1) Address of the smart contract is known, 2) The function caller has sufficient Ether to trigger.

Smart contracts have a significant advantage over traditional servers in that the code governing the business logic is now public (and easily verifiable).

*B. The Light Ethereum Subprotocol*

As previously stated, each validating node must store the blockchain's ledger. However, since the history of all transactions from the genesis block (the first mined block) to the current block must be downloaded, this requires a large amount of memory and storage. As of March 2021, the Ethereum chain data size was about 820 GB. On cell phones and Internet of Things (IoT) computers, this poses a serious issue. In order to address the aforementioned limitations, an alternative to the Ethereum full node is being created. The Light node chaindata is around 100 MB in size (March 2021), which is a tiny amount of data as compared to the maximum validating nodes. This chaindata is downloaded once on a mobile device (or other devices) and then synchronized through all DApps. Merkle Tree is the light client's main building block. This data structure, created by Ralph Merkle, allows for safe and efficient verification of blockchain data queries. A light client sends a question to light client servers, which then responds with the requested data as well as the Merkle branch. By going through the list of hashes from the returned object up to the tree's base, the client will check the data's integrity and validity.

*C. Private Ethereum Blockchain*

There is a permissioned version of blockchain as well as a public version. Private blockchain is another name for this edition. The decision to use each form is strongly affected by the application's specifications. Any EOA can submit transactions to other addresses and explore the network using online explorers like Etherscan in a public blockchain. A central authority is required in a permissioned blockchain to manage and maintain its own ledger. A permissioned blockchain is preferred in a country election process, for example, since the government controls the election process.

## III. LITERATURE REVIEW AND RELATED WORKS

There exist many different ideas for a secure e-voting system. This review will focus on 5 major ones which emerged repeatedly throughout the literature reviewed.

Biometric-secure cloud based e-voting system for election processes [4], a paper by J.A Samsul and M. B. Limkar. Security of the system is assured by the use of biometric fingerprint and iris authentication. When a voter wants to cast their vote, his fingerprints if matched will be allowed to cast a vote otherwise it would be prompted that he/she is not a registered user. If a voter comes to scan a second time then after scanning it would prompt that you have already casted vote. But there exist certain challenges. The system fails to recognize the user if their physical traits change even the slightest. If the data were stolen, they can't try to ‘change’ their identification traits like they can change their passwords during a security breach. The biometric system is also unreliable because it's an automatic system that depends on electricity to run. If there is a power shortage, no user can enter or exit.

In, an Implementation of Secure Online Voting System [5], Anisaara, N., Rakhi, B., Ashmita, K., Durgesh, G., and Tushar, N proposed a two-fold system, one is voting through a website and other is through mobile phones. The voter can use either of the two ways as per his convenience. First fold system is Internet voting which includes OTP generation for user identity. To increase security iris scanning and verification of the human eye is implemented. The second fold of the system is voting through normal mobile phones for which IVR (interactive voice response) system is implemented. The disadvantages of this system include, in the case of internet Voting, system crash or power failure and security risk (hackers/computer virus), and in the case of IVR, difficulty to understand voice prompts, long menus and wait time that stretches on for quite some time.

Homomorphic Cryptographic Solution on E-voting Systems [6], paper by Ahmed A. Abu Aziz, Hasan N.Qunoo, Aiman A. Abu Samra, uses homomorphic encryption and Non-Interactive Zero Knowledge Proof for securing e-voting systems. The proposed E-voting system software consists three main programs, authentication Server program, voting Server program, voter Program The Authentication Server Program is responsible for key generation processes and voter authentication program ( VA), the voting server program calculates a vote mask for every vote, the voter program encrypts the vote using the public key provided on BB. But the system has a centralized structure and remote voting is not possible. Also, encryption does not prevent DDoS attacks or hacking of databases.

Ring Signatures For An Anonymous E-Voting System [7], a paper by Oleksandr Kurbatov, Oleksiy Shapoval, V. N. Karazin Kharkiv Kravchenko describes the mechanisms for using ring signatures to ensure anonymity in a decentralized e-voting system. System consists of 3 elements: Validators, User identity system, End users. Validators are the main nodes of the system. In order to sign a transaction and, at the same time ensure the anonymity of the vote, the user selects a list of public keys of other users including his own. He calculates the value of the ring signature and sends the transaction to one of the validators. If all the specified keys have permission to vote,

then the transaction is correct and can be confirmed. In this system, it is hard to detect dual voting. Other disadvantages include inability of the voter to check the accuracy of his own vote. The user may be allowed to change the value of his voice. In this case, not one transaction will be counted, but the last transaction that was added to the blockchain.

The paper, blockchain-Based E-Voting System [8] by Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson, Mohammad Hamdaqa, Gísli Hjálmtýsson evaluate the use of blockchain as a service to implement an electronic voting (e-voting) system. The proposed blockchain-based e-voting system uses "permissioned blockchain (private blockchain)". What are Smart contracts?[9]. Smart contracts are programmable contracts that automatically execute when predefined conditions are met. Key benefits of smart contracts are cost saving, enhanced efficiency and risk reduction. Smart contracts redefine trust, as contracts are visible to all the users of the blockchain and can, therefore, be easily verified. In order to satisfy the privacy and security requirements for e-voting, and to ensure that the election system should not enable coerced voting, voters will have to vote in a supervised environment, which leads to have a 2 types of nodes in the network; District node: Represent each voting district, where each district node has a software agent that autonomously interacts with the "bootnode" and manages the life cycle of the smart contract on that node, Bootnode: Each institution, with permissioned access to the network, hosts a bootnode, where the bootnode helps the district nodes to discover each other and communicate. Defining a smart contract includes two parts: Election roles, and election processes. Election roles allow participation of election administrators and voters, district nodes and boot nodes. Election process consists of election creation, voter registration, tallying results, verifying votes and vote transaction.

## IV. PROPOSED SYSTEM

In this part, we present our proposed voting system, which aims to overcome the obstacles that currently exist in blockchain-based e-voting systems.

### *A. System Components*

The proposed platform consists of the following components:

1) Web application: Election commission may use the web application to create and administer new election events. In the blockchain network, each election is represented by a different smart contract. The election admins create the list of candidates participating in the election, then send an HTTP request to the Application Server with the results. This Web application's purpose is to be accessible as an Application Programming Interface (API) that allows admins to create new election events.

2) Application Server: The Application Server's main goal is to deploy the smart contract to the network using the data (questions and answers) from the web application. As a result, it includes an Ethereum Wallet (address) for deploying the contract, a complete node for connecting to the Ethereum network, and a database for storing the list of contract addresses that will be retrieved later by the mobile application.

3) Smart contracts: In our scheme, there are two parts for the smart contract: 1) Voting part, 2) Registration part for all voting cases, the registration contract is used only once. It is used to register and authenticate voters in a safe manner. As previously mentioned, the voting contract is written once during development and deployed multiple times by the Application Server with different election constituencies and corresponding candidate choices specified by the election commission.

4) SMS Gateway: An SMS Gateway is essentially necessary in our framework because it is used to authenticate users by sending SMS messages to the corresponding MSISDNs.

5) Mobile application: Voters use the smartphone application to enroll themselves in the system and then vote. It also allows users to view elections, see candidate lists and cast their votes. Furthermore, the application generates a comprehensive report detailing election statistics such as the number of votes cast per time slot, venue, and other factors. Since the voting takes place on the Ethereum network, an interface connecting the mobile application to the blockchain network is needed. As a consequence, the mobile app contains an Ethereum light client. The app sends all transactions it receives to the server.

### *B. Registration & Configuration*

A consumer must first register with the system in order to be able to vote. The program automatically retrieves the user's phone number from the Subscriber Identity Module (SIM card) when it is launched for the first time. Since transactions to the blockchain cost GAS, which is priced in Ether, the EOA must have enough Ether to register and vote. This app uses two Ethereum wallets, one for managers and one for voters, in which the election administering body fills with Ethereum. The private keys of the wallet are not shared with the users of the wallet. The server manages it, so that the wallet transactions can only be made from within the application. This ensures that no one misuses the crypto balance.

The Register function is called with the user's Voter ID as a parameter. The smart contract then checks to see if the Voter ID is on the list of accepted Voter IDs (Govt Issued). Then, using the Oraclize contract, it sends an HTTP request to an OTP(One Time Password) server, which generates an OTP code. Oraclize is a service that connects smart contracts to external web APIs in a safe manner. When the OTP is created, the contract connects Oraclize to the Short Message Service (SMS) Gateway, which sends an SMS to the phone number with the OTP as the payload.

After the MSISDN receives the SMS, the user enters the OTP code into the app, which triggers the Approve(MSISDN, OTP) feature. The contract then checks to see if the received transaction's address (msg.sender) matches the address of the first register call, as well as the OTP.

### *C. Creating an Election*

To create a new election, the election commission uses the previously mentioned web application. This organizer would be able to publish election information using graphs, maps, and textual representations.

To create a new election, the election commission is requested to register the candidate(s) through the admin panel of the web application.

Creating an Election on the Blockchain involves creating a voting contract. As a result, as transactions cost in Ethereum, the admin wallet must be used for payment for this

transaction. To make this process easier for organizers, payment has been automated into the web application. The web app then deploys the contract to the Ethereum network after securing the transaction cost. The newly generated smart contract's address will be returned to the organizer and also stored in the database so that it can be tracked later in the mobile app.

*D. Voting*

The program calls the Vote(string candidateID) method of the Vote(string candidateID) class when voting.

On the EVM, there is a dedicated smart contract. The voting contract then makes contact with the registration contract to see if the person has already enrolled. The program then checks to see whether the user has already cast a vote or whether the election has ended. If the conditions are met, the contract increases the count of the chosen choice, labels the user as voted, and sends the application a success letter.

The contract automatically rejects duplicate votes, restricting to one vote per Voter ID. This is considered the major advantage of our system compared to the others.

## V. IMPLEMENTATION

To implement the proposed scheme, we use a variety of technologies. Solidity [10], a contract-oriented programming language for writing smart contracts for both registration and voting, Django: server-side scripting for the Application Server, and eth-brownie to interface the ethereum client. To model the blockchain network, the Ropsten Testnet is used. Twilio's SMS gateway API is used.

## VI. CONCLUSION

We propose a decentralized voting network based on the Ethereum blockchain in this paper. The platform's key contribution is the mitigation of election frauds. Based on fingerprints or a special device located in polling centers, this method may be strengthened to make it more eligible for national government elections. The user interface and results visualization could be tailored to the needs of the consumer. This platform could replace existing centralized election polling systems and make voting easier for governments, competitions, and expositions, among other items. This platform incorporates a new business model for voting service providers, with election organizers, blockchain providers, and voters as participants. The blockchain provider allows voting smart contracts to be implemented by election organizers. The voting contracts configured according to the election norms are deployed in the Ethereum network by the Application Server. The revenue for the voting service provider will come from two places: the Election Organizers as a fixed cost to pay for the implementation of the Ethereum smart contract, and the voters when they register and vote.

## APPENDIX I

## OUR SMART CONTRACT

```
pragma solidity ^0.8.1;
contract Voting {
    address adminAddress;
    bool electionOn = false;
    modifier onlyOwner {
        require(msg.sender == adminAddress, "You must
be admin to do this.");
        _;
    }
    event voterAdded(string name, string
constituency);
    event candidateAdded(string name, string
constituency, string symbol);
    struct Voter {
        string name;
        uint idNo;
        string constituency;
        bool canVote;
        bool exists;
    }
    struct Candidate {
        string name;
        string constituency;
        string symbol;
        uint noOfVotes;
        bool exists;
    }
    Voter[] voters;
    mapping(uint => uint) ballot;
    Candidate[] candidates;
    mapping(uint => uint) votersList;
    constructor(){
        adminAddress = msg.sender;
        candidates.push(Candidate("nonce", "nonce",
"nonce", 0,  false));
        voters.push(Voter("nonce", 0, "nonce", false,
false));
    }
    function startElection() public onlyOwner {
        electionOn = true;
    }
    function endElection() public onlyOwner {
        electionOn = false;
    }
    function createVoter(string memory _name, uint
_id, string memory _constituency) public {
        require(!voters[votersList[_id]].exists,
"Voter id is already registered!.");
        voters.push(Voter(_name, _id, _constituency,
true, true));
        votersList[_id] = voters.length - 1;

        emit voterAdded(_name, _constituency);
    }
    function getVotersList() onlyOwner public view
returns(Voter[] memory){
        return voters;
    }
    function getCandidatesList() onlyOwner public view
returns(Candidate[] memory){
        return candidates;
    }
    function createCandidate(string memory _name,
string memory _constituency, string memory _symbol)
onlyOwner public {
        require(!electionOn, "Sorry. The election has
already began. You cannot add new candidates.");
        uint candId =  _generateHash(_name,
_constituency, _symbol);
        require(!candidates[ballot[candId]].exists,
"Candidate already in ballot list!.");
        candidates.push(Candidate(_name,
_constituency, _symbol, 0, true));
        ballot[candId] = candidates.length - 1;
        emit candidateAdded(_name, _constituency,
_symbol);
    }

    function _generateHash(string memory _name, string
memory _constituency, string memory _symbol) public
```

```
pure returns (uint){
        uint hash =
uint(keccak256(abi.encodePacked(_name, _constituency,
_symbol)));
        return hash;
    }
    function vote(uint _voterId, string memory
_voteTo, string memory _constituency, string memory
_symbol) public{
        require(voters[votersList[_voterId]].exists,
"You are not registered to vote.");
        require(electionOn, "Sorry. The election has
ended.");
require(voters[votersList[_voterId]].canVote, "Sorry.
You already voted.");
        uint candId = _generateHash(_voteTo,
_constituency, _symbol);
require(candidates[ballot[candId]].exists, "Selected
candidate not found.");
require(keccak256(abi.encodePacked(candidates[ballot[c
andId]].constituency)) ==
keccak256(abi.encodePacked(voters[votersList[_voterId]
].constituency)), "Sorry. You are voting for a
candidate who is not in your constituency.");
candidates[ballot[candId]].noOfVotes++;
voters[votersList[_voterId]].canVote = false;
    }
    function getElectionResult() public view returns
(Candidate[] memory){
        require(!electionOn, "Election has not
ended.");
        return candidates;
    }
    function getElectionState() public onlyOwner view
returns (bool){
        return electionOn;
    }
}
```

## REFERENCES

[1] V. Buterinet al., “A next-generation smart contract and decentralized application platform,”white paper, 2014.
[2] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
[3] G. Wood, “Ethereum: A secure decentralised generalised transactionledger,”Ethereum project yellow paper, 2014.
[4] J. A. Samsul and M. B. Limkar, “A biometric-secure cloud based e-voting system for election processes,” International Journal of Electrical and Electronics Engineering Research (IJEEER), 2014.
[5] Anisaara, N., Rakhi, B., Ashmita, K., Durgesh, G., and Tushar, N. (2015). An implementation of secure online voting system. International Journal of Engineering Research and General Science
[6] Aziz, Ahmed & Qunoo, Hasan & Abusamra, Aiman. (2018). Using Homomorphic Cryptographic Solutions on E-voting Systems. International Journal of Computer Network and Information Security. 10. 44-59. 10.5815/ijcnis.2018.01.06.
[7] O. Kurbatov, P. Kravchenko, N. Poluyanenko, O. Shapoval and T. Kuznetsova, "Using Ring Signatures For An Anonymous E-Voting System," 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), 2019, pp. 187-190, doi: 10.1109/ATIT49449.2019.9030447.
[8] Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson, Mohammad Hamdaqa, Gísli Hjálmtýsson 2018,Blockchain-Based E-Voting System,2018 IEEE 11th International Conference on Cloud Computing (CLOUD).
[9] What Are Smart Contracts? A Beginner’s Guide to Smart Contracts. (2018). Retrieved 26 8, 2018, from https:// blockgeeks.com/guides/smart-contracts/., 2018.
[10] C. Dannen,Introducing Ethereum and Solidity.Springer, 2017.