Decentralized Voting Platform Based on Ethereum Blockchain

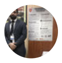
Conference Paper · November 2018

DOI: 10.1109/IMCET.2018.8603050

4 authors, including:

David Khoury
American University of Science and Technology
18 PUBLICATIONS **332** CITATIONS
SEE PROFILE

Elie Kfoury
University of South Carolina
59 PUBLICATIONS **694** CITATIONS
SEE PROFILE


Hamza Harb
American University of Science and Technology
1 PUBLICATION **98** CITATIONS
SEE PROFILE

Decentralized Voting Platform Based on Ethereum Blockchain

David Khoury, Elie F. Kfoury, Ali Kassem, Hamza Harb
Department of Computer Science
American University of Science and Technology
Beirut, Lebanon
{dkhoury, ekfoury}@aust.edu.lb, {amk10062, hmh10056}@students.aust.edu.lb

***Abstract*—In centralized environments, the results of voting events have always been questionable and perceived differently by voters. Most existing E-Voting systems are based on centralized servers where the voters must trust the organizing authority for the integrity of the results. In this paper we propose a novel approach for a decentralized trustless voting platform that relies on Blockchain technology to solve the trust issues. The main features of this system include ensuring data integrity and transparency, and enforcing one vote per mobile phone number for every poll with ensured privacy. To accomplish this, the Ethereum Virtual Machine (EVM) is used as the Blockchain runtime environment, on which transparent, consistent and deterministic smart contracts will be deployed by organizers for each voting event to run the voting rules. Users are authenticated through their mobile phone numbers without the need of a third party server. Results showed that the system is feasible and may offer a step towards ideal environments for such experience.**

***Index Terms*—Voting, Blockchain , Ethereum, Authentication, EVM, Smart Contract, trustless, decentralized, MSISDN**

I. Introduction

Throughout the years, Voting has always been regarded as the primary method used by individuals to share their opinions on controversial issues and debates . It is a democratic practice, enabling people to formally express their choice against a ballot question, candidate election, political party and others [1].

The currently familiar E-Voting systems are based on the well-known client-server architecture: The server and the voting data are governed by a trusted third party responsible for the ownership and integrity of the votes. Unfortunately, voting is strongly dependent on trust imposed by the organizing authorities running the election. Over the past decades, several allegations criticizing the election processes were reported [2] [3]. Most of the developed E-voting solutions adopt this architecture in which data and application's business logic reside on the authorities servers. While this topology seems practical for some applications, it has major drawbacks on others:

1) Lack of data integrity and security protection measures.
2) Single point of failure.
3) Centralized control and lack of secure transaction validation protocols.
4) Obscure run time environment.
5) Unknown business rules running in the server.

Blockchain on the other hand is an emergent technology that provides network decentralization with no single point of failure and ensures data immutability through cryptographic functions and consensus algorithms and protocols. The Ethereum Blockchain [4] is an open-source distributed computing platform featuring a Turing-complete scripting language in which software engineers can deploy decentralized applications (DApps) and benefit from the distribution property inherited from the Blockchain technology. Therefore, DApps will have the following Blockchain features:

1) Strong data integrity.
2) Decentralized control and validation through consensus mechanisms.
3) Transparent run time environment.
4) Public business rules running in the run-time environment.
5) High-availability.

Blockchain is gaining attention in several domains, even in the telecommunication industry [5].

In this paper we propose a decentralized online voting platform based on the Blockchain technology aiming at resolving the trust concerns raised by conventional E-voting systems. This system introduces a novel mechanism for authenticating and validating the eligible voters.

The main contributions of this solution include: (1) Enforcing voting data immutability and data integrity, (2) Ensuring robustness and reliability of the voting system, (3) Decentralizing the registration and validation mechanisms of voters, (4) Transparency, clarity and determinism of the voting environment,(5) Public visualization of the smart contracts votes, (6) Restricting each voter to have a single vote per valid Mobile Station International Subscriber Directory Number (MSISDN), and (7) Privacy-aware regarding the confidentiality of the recorded votes.

The rest of the paper is divided as follows: Section II gives an overview on Blockchain technology and Ethereum, Section III discusses related work and highlights their limitations, Section IV introduces the proposed system and its advantages compared to the existing E-voting systems. Implementation and results are demonstrated in Section V. Section VI provides some concluding remarks and the intended future work.

II. Background on Blockchain Technology and Ethereum

Blockchain can be referred to as a public decentralized database with replicas distributed over several nodes simultaneously [6]. In Blockchain there is no authority in charge of managing and maintaining the ledger of transactions. The validity of the ledger's version is established through a consensus mechanism among the validating nodes. The use of Blockchain technology allows a secure validation of transaction's data integrity. Bitcoin, for instance, is the first application developed over Blockchain by Satoshi Nakamoto [7].

On another hand, Ethereum Blockchain is an open-source, distributed and decentralized computing infrastructure that executes programs called smart contracts [4]. It is developed to enable decentralization for applications and not only for a digital currency. It is achieved using a virtual machine (Ethereum Virtual Machine, EVM) to execute a Turing-complete scripting language. Unlike Bitcoin in which only Boolean evaluation of spending conditions are considered, EVM is somehow similar to a general-purpose computer that simulates what a Turing machine can execute. Changing the state of a contract in the Blockchain requires 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): An account identified by a wallet address and controlled by a private key. The holder of this private key can transfer ether and sign transactions from this account. EAOs are considered user type accounts and are linked to unique cryptographic keys pair, generated upon account creation. The public key is used to reference the account and also called EOA address whereas the private key on the other hand is used to sign transaction before executing any type of transaction on the network to prove authenticity. EOAs have balances which hold Ether cryptocurrency [8].

2) Smart Contract: A smart contract is an account that is controlled by its own code [9]. It is considered as an autonomous agent executed by the EVM and is the core foundation and the main building blocks of any DApp [10]. Once this code is deployed on the Blockchain, the EVM will take care of running it as long as the conditions apply. It is important to note that smart contracts once deployed to the Blockchain network, they can be publicly visited and viewed via their address with all their associated 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 provide an important added value: The code ruling the business logic is now public (easily verifiable) and not obscure as in conventional servers.

B. The Light Ethereum Subprotocol

As mentioned previously, the Blockchain's ledger must be stored on each validating node. However, this requires a considerable amount of memory and storage as the history of all transactions from the genesis block (the first mined block) till the current block must be downloaded. The Ethereum chaindata size as of March 2018 is approximately 440 GB. This imposes a severe problem on mobile phones and Internet of Things (IoT) devices. For this purpose, an alternative to the Ethereum full node, is developed [11] to overcome the aforementioned limitations. The Light node chaindata is approximately 0.005 GB (March 2018), a relatively small size compared to the full validating nodes. This chaindata is downloaded *once* on a mobile (or other devices as well), and maintained (synchronized) for all DApps. The main building block of the light client is Merkle Tree [12]. This data structure invented by Ralph Merkle allows secure and efficient verification of the queried data from the Blockchain. A light client initiates a query to light client servers, and the server in turn replies back with the requested data along with the Merkle branch. The branch allows the client to verify the integrity and the authenticity of the data by going through the list of hashes from the returned object up to the tree's root.

C. Private Ethereum Blockchain

Not only public Blockchain is available, but also a permissioned version exist. This version is also referred to as *Private* Blockchain [13]. The question whether to adopt each type depends heavily on the application's requirements. In a public Blockchain, any EOA can send transactions to other addresses and explore the network using online explorers such as Etherscan. In a permissioned Blockchain, a central authority is needed to control and maintain its own ledger. In a country election process for instance, a permissioned Blockchain is preferred as the government is in control of the election process.

III. Literature review & Related Work

We present in this section various solutions that attempt to integrate E-voting and Blockchain to enable decentralization of voting services. We then highlight the added value of our proposed system compared to the others.

a) Towards Secure E-Voting Using Ethereum Blockchain: Ali Kaan Ko et al. discuss in their paper entitled ”Towards Secure E-Voting Using Ethereum Blockchain” [14] a decentralized voting solution based on Ethereum Blockchain. It states that an E-Voting system must be secure by being fully transparent (privacy-aware) and not allowing duplicated votes. It suggests deploying the E-Voting application as a smart contract and allowing users with valid EOAs to vote on that contract (once per address to a single question). Nevertheless, this solution lacks true automated address verification protocol since the EOAs get their right to vote from a *Centralized Authority* to become eligible voters. The main advantages it offers are business rules transparency and single vote restriction per EOA.

b) The Future of E-Voting: In their paper entitled ”The Future of E-Voting”, Tarasov et. al discussed E-Voting and its potential use with Blockchain [15]. In addition to transparency, privacy, and integrity which became inherent properties of Blockchain DApps, this solution proposes a registration phase to verify the users’ identities. Registration is the first step of the protocol, and is required as part of the identity verification for audit purposes. It helps keeping track of which voters have cast a ballot. Although the verification process is done using a Challenge-Response handshake protocol, it involves again a server (*Centralized Authority*) to handle the verification process and add the users’ data (email addresses) to the database. It is worth mentioning that email addresses are relatively easy to spoof nowadays.

c) Decentralized, Transparent, Trustless Voting on the Ethereum Blockchain: Fernando Lobato Meeser, in his paper entitled ”Decentralized, Transparent, Trustless Voting on the Ethereum Blockchain” [16] discusses two types of ongoing issues with E-Voting solutions. First, the capability of anyone to tally the results from the smart contract before having all the votes casted, and second, the anonymity of the votes since public keys can be associated with the recorded votes. In this paper, the author presents the implementation of a voting system as a smart contract running on Ethereum that uses threshold keys and linkable ring signatures. Nevertheless, this solution again includes a registration phase, and voters rely on a *Centralized Authority* to register their public key for casting a vote.

d) ”Removing Trusted Tallying Authorities: Self-Enforcing E-Voting over Ethereum”: Published paper by Patrick McCorry et at. [17], claim that while preserving the voters privacy, their protocols allow anyone, including observers, to verify the integrity of the election without having to trust authorities. They achieve by namely Open Vote Network (OV-net), Direct Recording Electronic with integrity (DRE-i), and DRE-i with enhanced privacy (DRE-ip)[13]. In spite of that, their system requires an authority to setup a list of eligible voters and transfer them to the Ethereum Blockchain before the elections starts. Although having a predefined list of voters is a good choice for certain use cases, but the challenge remains in fully decentralizing the voting process.

IV. Proposed System

In this section we introduce our proposed voting system that aims at solving the existing barriers in Blockchain-based E-Voting systems. Fig. 1 illustrates at high level the proposed system architecture and the interaction between the system’s components.

A. System Components

The proposed platform consists of the following components:

1) Web application: The web application aids event administrators in creating and managing new voting events. Each Voting event is represented as a separate Smart Contract in the Blockchain network. The administrator fills-in the list of questions and their corresponding answers and then initiate an HTTP request to the Event Management Server containing the entered data. The goal of this Web application is to be available as an Application Programming Interface (API) [18] allowing any user to create new voting events.

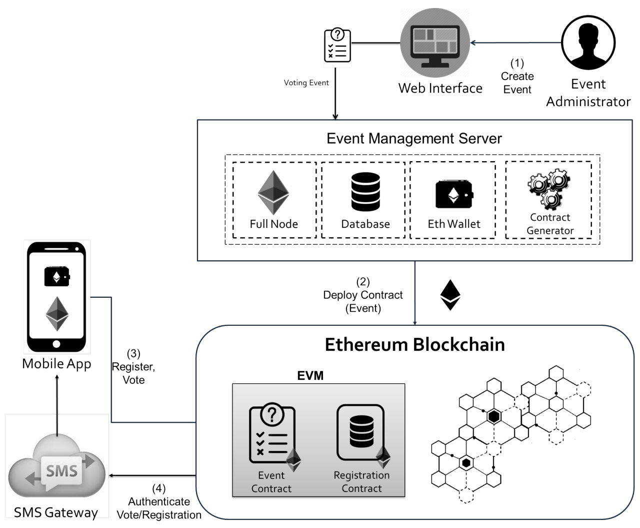


Fig. 1. System Architecture

2) Event Management Server: The main goal of the Event Management Server is to deploy the Smart Contract to the network with the data (questions and answers) received from the web application. Therefore, it contains an Ethereum Wallet (address) which is required to deploy the contract, a full node to interface the Ethereum network, and a database to store the list of contract addresses which will be fetched later by the mobile application.

3) Smart contracts: Two types of smart contracts exist in our system: 1) Registration contract, 2) Voting contract. The registration contract is deployed once for all voting events. It serves at securely registering and authenticating the voters. The voting contract is written once at development time, and deployed several times by the Event Management Server with different questions and answers specified by the event administrator as explained previously. Appendices A and B list the code of both contracts.

4) SMS Gateway: An SMS Gateway [19] is ultimately mandatory in our system as it plays an important role in authenticating the users via sending SMS messages to the destined MSISDNs.

5) Mobile application: The mobile application is used by voters to register themselves in the system and then vote. It also provides the users the ability to fetch events, view questions and options, and visualize in real-time the results. Moreover, the application provides a detailed report showing the voting event statistics related to the frequency of votes per time slot, location, and others ... As the voting process takes place in the Ethereum network, it is mandatory to have an interface connecting the mobile application to the

Blockchain network. Therefore, an Ethereum light client (discussed in Section II Part B) is integrated within the mobile app. All transactions transmitted from the app are sent to the Blockchain network through this client.

B. Registration & Configuration

To become an eligible voter, a user must first register to the system. Fig. 2 describes the voter's registration mechanism. Upon launching the application for the first time, the application automatically retrieves the user's MSISDN (phone number) from the Subscriber Identity Module (SIM card). Registration and Voting require the EOA to possess sufficient Ether as transactions to the Blockchain cost GAS which is priced in Ether. The application generates an empty Ethereum wallet and requests the user to fill it through the wallet management function method as described in [20].

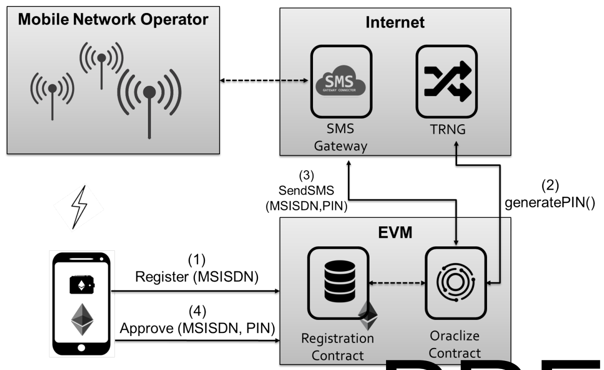


Fig. 2. Voter Registration Mechanism

Once the user has sufficient Ether balance, the *Register(MSISDN)* function is called with the user's MSISDN as a parameter. The smart contract then validates that the MSISDN does not exist in the list of approved phone numbers. Then, it sends an HTTP request through the Oraclize contract to a True Random Number Generator (TRNG) server to generate a random Personal Identification Number (PIN) code. Oraclize is a service that offers a secure connection between the smart contracts and external web APIs. When the PIN is generated, the contract again interfaces Oraclize to contact the Short Message Service (SMS) Gateway which in turn sends an SMS to the MSISDN containing the PIN as payload.

Once the SMS is received by the MSISDN, the user then enters the PIN code to the application which will invoke the *Approve(MSISDN, PIN)* function. The contract then validates the received transaction by checking if its address (*msg.sender*) matches the address of the first register call, and checks if the PIN is correct.

C. Creating a Voting Event

An event organizer uses the web application discussed previously to create a new voting event. This organizer will be able to monitor ongoing event statistics through graphs, charts and textual representations updated at a realtime.

To create a new event, the event organizer is requested to input the question(s) and the corresponding answer(s) through the web.

Technically, creating a voting event means creating a voting contract on the Blockchain. Therefore this transaction must be charged for the event organizer as transactions cost in Ethereum. To simplify this process for organizers, a payment API is integrated into the web application allowing the organizer to pay using fiat currency for the transaction. After securing the transaction cost, the web app deploys the contract to Ethereum the network. The address of the newly created smart contract will be returned to the organizer and also stored in the database to track it later in the mobile application.

D. Voting

The voting mechanism is depicted in Fig. 3. While Voting, the application calls the *VoteFor(string option)* method of the designated smart contract deployed on the EVM. Next, the voting contract contacts the registration contract to check if the user is already registered. Then, it checks if the user already voted or the event is finished. If the conditions are satisfied, the contract increments the count of the selected option, marks the user as voted, and returns a success message to the application.

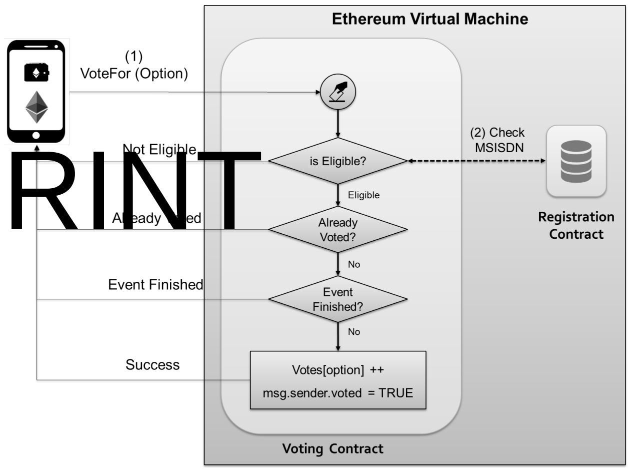


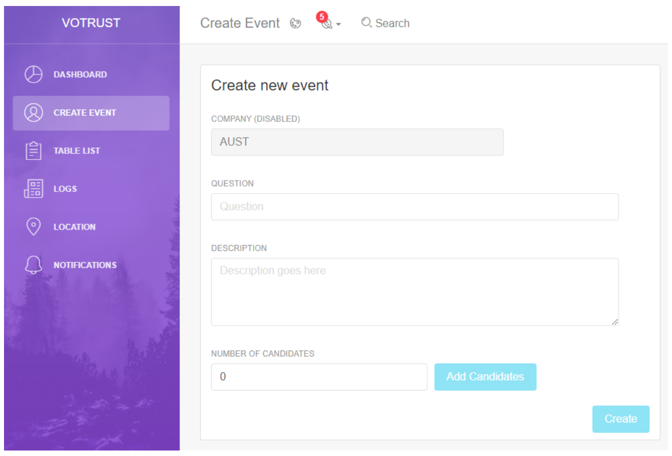
Fig. 3. Voting Mechanism

The contract automatically rejects duplicate votes, allowing to restrict one vote per MSISDN. This is considered the major advantage of our system compared to the others.

V. Implementation and Results

To validate the proposed system, we implemented the solution using various technologies. Solidity, a contract-oriented programming language for writing both registration and voting smart contracts [21], NodeJS [22]: Server side scripting for the Event Management Server, Web3js to interface the light client [21], and HTML5 web-app compiled using Apache Cordova [23] for the mobile side. The Ropsten Testnet [24] is used to simulate the Blockchain network. Twilio [25] is used as the SMS gateway API.

Fig. 4 shows the web page used by event organizers to create a new voting event.



Create new event

COMPANY (DISABLED)
AUST

QUESTION
Question

DESCRIPTION
Description goes here

NUMBER OF CANDIDATES
0

Fig. 4. Creating a New Voting Event

Fig. 5 demonstrates the dashboard in which event organizers can visualize the voting results.

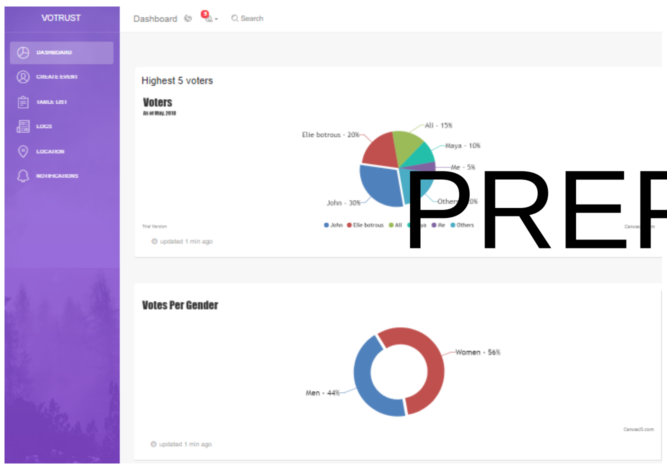


Fig. 5. Realtime Visualization of the Results

The registration of a user to the system took from 2 to 4 minutes. This amount of time is spent only once upon configuration. Voting on the other hand took from 40 seconds to 2 minutes. This time is spent on each vote.

VI. Conclusion and Future Work

In this paper we have proposed a decentralized voting platform based on Ethereum Blockchain. The main contribution of this platform is the restriction of multiple votes per mobile (MSISDN). This system could be developed further to make it more eligible for national government elections, based on fingerprint or a special device located in the voting centers. The user interface and results visualization could be customized and adapted to the customer requirements. This platform could replace the existing centralized systems based on SMS polling and facilitate voting organized by governments, competitions, expositions, etc... This platform opens up a new business model for a voting service providers where the players include: Voting event organizers, voting service providers, and voters. The Voting service provider enables the voting event organizers to deploy an event voting smart contract. The Event Management Server deploys in the Ethereum network the voting contracts configured according the voting event customer. The Voting service provider revenues could be generated from two sources: the Voting Event Organizers as a fixed cost to compensate for the deployment of the smart contract in Ethereum, and from the voters upon registration and voting.

Appendix A
Registration Contract

```
pragma solidity 0.4.24;

import "browser/OraclizeAPI1.sol";
import "github.com/willitscale/solidity-util/lib/Strings.sol";

contract Register is usingOraclize
{
    using Strings for string;

    mapping(address => bool) public EligbleVotersAddresses;
    mapping(string => bool) RegisteredPhoneNumbers;
    mapping(bytes32 => string) OraclizeIDtoResult;
    mapping(address => bytes32) GeneratePINOraclizeID;
    mapping(address => string) AddressesExpectedToVerify;

    function isEligble(address _address) public view returns(bool) {
        if (EligbleVotersAddresses[_address] == true)
            return true;
        else return false;
    }

    function verify(string code) public {
        bytes32 OracleID = GeneratePINOraclizeID[msg.sender];
        if (OraclizeIDtoResult[OracleID].compareTo(code)) {
            EligbleVotersAddresses[msg.sender] = true;
            RegisteredPhoneNumbers[AddressesExpectedToVerify[msg.sender]] = true;
            delete AddressesExpectedToVerify[msg.sender];
            delete OraclizeIDtoResult[OracleID];
            delete GeneratePINOraclizeID[msg.sender];
        }
    }

    function SendSMS(string phonenumber) public payable {
        bytes32 OracleID;
        if (RegisteredPhoneNumbers[phonenumber] == true ||
    phonenumber.compareTo(AddressesExpectedToVerify[msg.sender])
    || GeneratePINOraclizeID[msg.sender] == 0)
            revert();
        bytes32 ID = GeneratePINOraclizeID[msg.sender];
        string memory jsonStart = '{"To" : "';
        string memory jsonMiddle = '","From" : "+19103708518" , "Body": "';
        string memory jsonEnd = '"}';
        OracleID = oraclize_query('URL', 'json(https://API:PASS@api.twilio.com..../
            Messages.json).status',
    strConcat(jsonStart, phonenumber, jsonMiddle, OraclizeIDtoResult[ID], jsonEnd))
        ;
        AddressesExpectedToVerify[msg.sender] = phonenumber;
    }

    function generatePIN() public payable {
        bytes32 OracleID;
        if (EligbleVotersAddresses[msg.sender] == true)
            revert();
        if (!AddressesExpectedToVerify[msg.sender].compareTo("0
            x0000000000000000000000000000000000000000"))
            delete AddressesExpectedToVerify[msg.sender];
        OracleID = oraclize_query("WolframAlpha", "2 digit integer random");
        GeneratePINOraclizeID[msg.sender] = OracleID;
    }

    function __callback(bytes32 _OraclizeID, string _result) public {
        if (_result.compareTo("Queued") == false)
            OraclizeIDtoResult[_OraclizeID] = _result;
    }
}
```

Appendix B
Voting Contract

```
pragma solidity ^ 0.4.0;

import "browser/IRegister.sol";

contract VotingContract {
    bytes32 PollWinner;
    bytes32 Question;
    address EventAdmin;
    uint NumOfCand;
    bool completed;
    uint totalNumVotes;
    address system = 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C;
    address RegistrationContractAddress = 0
           x0c762d861a8873c54ed938c68ea1d5f627b562aa;

    mapping(address => bool) public verify;
    mapping(bytes32 => uint) public votes;
    mapping(uint => bytes32) public options;

    modifier onlyOwner {
        require(msg.sender == EventAdmin);
        _;
    }

    function getNumVotes() public view returns(uint) {
        return totalNumVotes;
    }

    function getNumCandidates() public view returns(uint) {
        return NumOfCand;
    }

    function getQuestion() public view returns(bytes32) {
        return Question;
    }

    function isEligble(address _address) public view returns(bool) {
        Register reg = Register(RegistrationContractAddress);
        return reg.isEligble(_address);
    }

    function VotingContract(bytes32 theQuestion, bytes32[] candidateNames) {
        Question = theQuestion;
        EventAdmin = msg.sender;
        NumOfCand = candidateNames.length;

        for (uint i = 0; i < candidateNames.length; i++) {
            options[i] = candidateNames[i];
            votes[candidateNames[i]] = 0;
        }
    }

    function getCandidate(uint candidateIndex) public view returns(bytes32) {
        if (candidateIndex > NumOfCand - 1)
            return (0);
        else return (options[candidateIndex]);
    }

    function CanVote() onlyOwner view returns(bool) {
        if (verify[msg.sender] == true || completed == true || !isEligble(msg.
             sender))
            return false;
        else return true;

    }

    function getTotalVotesFor(bytes32 candidate) public view returns(uint) {
        return (votes[candidate]);
    }

    function VoteFor(bytes32 candidate) public payable {
        if ((verify[msg.sender] == true) || (completed == true)
    || (msg.value == 0) || !isEligble(msg.sender))
            revert();
        else {
            votes[candidate] += 1;
            totalNumVotes += 1;
            verify[msg.sender] = true;
        }
    }

    function getPot() onlyOwner view returns(uint) {
        return this.balance;
    }

    function FinishAndDistributeTheRevenues(bytes32 winner) onlyOwner returns(uint,
           uint) {
        EventAdmin.transfer(this.balance / 3);
        system.transfer(this.balance);
        PollWinner = winner;
        completed = true;
        return (EventAdmin.balance, system.balance);
    }
}
```

References

[1] A. J. Bott, *Handbook of United States election laws and practices: political rights*. Greenwood Publishing Group, 1990.
[2] W. R. Mebane Jr, "Fraud in the 2009 presidential election in iran?" *Chance*, vol. 23, no. 1, pp. 6–15, 2010.
[3] R. Jiménez and M. Hidalgo, "Forensic analysis of venezuelan elections during the chávez presidency," *PloS one*, vol. 9, no. 6, p. e100884, 2014.
[4] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.
[5] E. F. Kfoury and D. J. Khoury, "Secure end-to-end volte based on ethereum blockchain," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2018, pp. 1–5.
[6] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology? a systematic review," *PloS one*, vol. 11, no. 10, p. e0163477, 2016.
[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[8] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.
[9] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 79–94.
[10] M. Pilkington, "11 blockchain technology: principles and applications," *Research handbook on digital transformations*, p. 225, 2016.
[11] Ethereum, "Light ethereum subprotocol." [Online]. Available: https://github.com/ethereum/wiki/wiki/Light-client-protocol
[12] R. C. Merkle, "Method of providing digital signatures," Jan. 5 1982, uS Patent 4,309,569.
[13] M. Vukolić, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 2017, pp. 3–7.
[14] A. K. Koç and U. C. Çabuk, "Towards secure e-voting using ethereum blockchain."
[15] P. Tarasov and H. Tewari, "The future of e-voting." *IADIS International Journal on Computer Science & Information Systems*, vol. 12, no. 2, 2017.
[16] F. L. Meeser, "Decentralized, transparent, trustless voting on the ethereum blockchain," 2017.
[17] P. McCorry, E. Toreini, and M. Mehrnezhad, "Removing trusted tallying authorities," Technical report, Newcastle University, 2016. Cited on, Tech. Rep., 2016.
[18] D. Orenstein, "Quickstudy: Application programming interface (api)," 2000.
[19] V. K. Katankar and V. Thakare, "Short message service using sms gateway," *International Journal on Computer Science and Engineering*, vol. 2, no. 04, pp. 1487–1491, 2010.
[20] E. F. Kfoury and D. J. Khoury, "Secure end-to-end voip system based on ethereum blockchain," *Journal of Communications*, vol. 13, no. 8, pp. 450–455, 2018.
[21] C. Dannen, *Introducing Ethereum and Solidity*. Springer, 2017.
[22] J. Wilson, *Node. js 8 the Right Way: Practical, Server-side Javascript that Scales*. Pragmatic Bookshelf, 2018.
[23] R. K. Camden, *Apache Cordova in action*. Manning Publications Co., 2015.
[24] K. Iyer and C. Dannen, "The ethereum development environment," in *Building Games with Ethereum Smart Contracts*. Springer, 2018, pp. 19–36.
[25] "Twilio - connect the world with the leading platform for voice, sms, and video." [Online]. Available: https://www.twilio.com/