Blockchain-Based E-Voting System

Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson
School of Computer Science
Reykjavik University, Iceland
{fridrik14, gunnlaugur15}@ru.is

***Abstract*—Building an electronic voting system that satisfies the legal requirements of legislators has been a challenge for a long time. Distributed ledger technologies is an exciting technological advancement in the information technology world. Blockchain technologies offer an infinite range of applications benefiting from sharing economies. This paper aims to evaluate the application of blockchain as service to implement distributed electronic voting systems. The paper elicitates the requirements of building electronic voting systems and identifies the *legal* and technological limitations of using blockchain as a service for realizing such systems. The paper starts by evaluating some of the popular blockchain frameworks that offer blockchain as a service. We then propose a novel electronic voting system based on blockchain that addresses all limitations we discovered. More generally this paper evaluates the potential of distributed ledger technologies through the description of a case study, namely the process of an election and implementing a blockchain-based application which improves the security and decreases the cost of hosting a nationwide election.**

I. Introduction

In every democracy, the security of an election is a matter of national security. The computer security field has for a decade studied the possibilities of electronic voting systems [1], with the goal of minimizing the cost of having a national election, while fulfilling and increasing the security conditions of an election. From the dawn of democratically electing candidates, the voting system has been based on pen and paper. Replacing the traditional pen and paper scheme with a new election system is critical to limit fraud and having the voting process traceable and verifiable [2].

Electronic voting machines have been viewed as flawed, by the security community, primarily based on physical security concerns. Anyone with physical access to such machine can sabotage the machine, thereby affecting all votes cast on the aforementioned machine.

Enter blockchain technology. A blockchain is a distributed, immutable, incontrovertible, public ledger. This new technology works through four main features:

(i) The ledger exists in many different locations: No single point of failure in the maintenance of the distributed ledger.
(ii) There is distributed control over who can append new transactions to the ledger.
(iii) Any proposed "new block" to the ledger must reference the previous version of the ledger, creating an immutable chain from where the blockchain gets its name, and thus preventing tampering with the integrity of previous entries.
(iv) A majority of the network nodes must reach a consensus before a proposed new block of entries becomes a permanent part of the ledger.

These technological features operate through advanced cryptography, providing a security level equal and/or greater than any previously known database. The blockchain technology is therefore considered by many [3], including us, to be the ideal tool, to be used to create the new modern democratic voting process.

This paper evaluates the use of blockchain as a service to implement an electronic voting (e-voting) system. The paper makes the following original contributions: (i) research existing blockchain frameworks suited for constructing blockchain-based e-voting system, (ii) propose a blockchain-based e-voting system that uses "permissioned blockchain" to enable *liquid democracy*.

The reminder of this paper is organized as follows: In section II, we discuss design considerations for election systems. In section III, we present our blockchain based e-voting system and evaluate some of the popular blockchain frameworks for realizing the system. In section IV, we discuss some of the security and legal considerations and limitations regarding designing an electronic voting system for national elections. Related work is presented in Section V. Finally, conclusions and directions for future work are presented in Section VI.

II. Preliminaries of E-Voting and Blockchain

This section explains the liquid democracy and its design consideration. We then provide an overview of blockchain and smart contract technology and its capabilities as a service for implementing an e-voting system for liquid democracy along with an overview of Zero-Knowledge proofs and their use cases in such systems.

A. Liquid Democracy Design Considerations

The main idea in a liquid democracy [6] is that the voter has the power, at any given moment, to review the way his vote was cast in terms of a specific legislative proposal or a bill. This allows people with domain-specific knowledge to better influence the outcome of decisions, which should lead to an overall better governance. The concept of liquid democracy could be a possible answer to the public requests, but there are technical and social barriers in the way. The solution to the technical concerns associated with the liquid democracy concept could be vital for the evolution of democracy as we know it.

Below, we list our envisioned essential requirements that need to be fulfilled by an e-voting system in order for it to effectively be used in a national election:

(i) An election system should not enable coerced voting.
(ii) An election system should not enable traceability of a vote to a voters identifying credentials.
(iii) An election system should ensure and proof to a voter, that the voters vote, was counted, and counted correctly.
(iv) An election system should not enable control to a third party to tamper with any vote.
(v) An election system should not enable a single entity control over tallying votes and determining an elections result.
(vi) An election system should only allow eligible individuals to vote in an election

B. Blockchain as a Service

The blockchain technology was introduced in 2008 when Satoshi Nakamoto created the first cryptocurrency called Bitcoin. The Bitcoin blockchain technology uses a decentralized public ledger combined with PoW(Proof-of-Work) based stocastic concensus protocol, with financial incentives to record a totally ordered sequence of blocks, the blockchain. The chain is replicated, cryptographically signed and publicly verifyable at every transaction so that no-one can tamper with the data that has been written onto the blockchain. The blockchain structure is an append-only data structure, such that new blocks of data can be written to it, but cannot be altered or deleted The blocks are chained in such a way that each block has a hash that is a function of the previous block, providing the assurance of immutability.

Whereas the Bitcoin blockchain publishes all elements of the entire chain, in general other types of blockchain can be *public*, *private* or *consortium* based. Public blockchains grant access to read and ability to create a transaction to any user on that network. This type is mostly used for cryptocurrencies (e.g., Bitcoin, Ethereum, Dogecoin and Auroracoin). Consortium blockchain is a "partially decentralized" blockchain[17], where the consensus process is controlled by a pre-selected set of nodes. Imagine a consortium of 15 financial institutions, each of which operates a node of which 10 must sign every block in order for the block to be valid. The right to read the blockchain can be public or restricted to the participants. Private blockchain limits not only the write access but the read access as well, to specific participants who can verify their transaction internally. That makes the transaction on a private network cheaper, since they only need to be verified by few nodes that are trusted and with guaranteed high processing power. Nodes can be trusted to be very well-connected and faults can quickly be fixed by manual intervention, allowing the use of consensus algorithms which offer finality after much shorter block times.[17]

In our proposal, we will use a permissioned blockchain, a variation of the consortium-based chains, which uses the proof-of-authority(POA) consensus algorithm. In proof-of-authority-based networks, transactions and blocks are validated by approved accounts, known as validators. This process is automated and does not require the validators to be constantly monitoring their computers. A permissioned blockchain which uses the POA consensus algorithm enables us to set restrictions on a set of selected known entities to validate and certify transactions on the blockchain and censor transactions arbitrarily, with their identity and reputation at stake. This otherwise needs to be done by miners on a public blockchain which uses the proof-of-work consensus algorithm. Rather than employing mining fees, like the public blockchains in operation require, using a permissioned blockchain, validators get payed for the service they provide by acting as validators in the system. Moreover, using a private network limits the possibility for an eavesdropper to monitor traffic or read the incoming data. This is needed to fulfill voting rights so that voters can cast votes without leaking their identity or voting data.

1) Smart Contracts: Smart contracts are trackable and irreversible applications that execute in a decentralized environment (e.g., blockchain). Once the smart contract has been deployed nobody can edit the code or change its execution behavior. Smart contract execution guarantees to bind parties together to an agreement as written. This creates a new powerful type of trust relationship that does not rely on a single party. Smart contracts enables better management for realizing and administering digital agreements because they are self-verifying and self-executing[19].

The phrase and concept of "smart contracts" is attributed to Nick Szabo, who has a degree in law and computer science. His expressed goal with smart contracts is to bring the highly evolved practices of law to the design of electronic commerce protocols between strangers on the Internet. Ethereum provides an open-source blockchain platform that can be used to deploy smart contracts. Ethereum introduced a new programming language called Solidity (similar to Javascript) to write these contracts. We implement our e-voting system, as a system based on a smart contract in a permissioned blockchain. The functionality in detail will be discussed in Section III.

2) Non-Interactive Zero-Knowledge proof: Another concept that is not directly related to blockchain but can be seen as an essential component for satisfying some of the requirements of building an e-voting system on a blockchain is zero-knowledge proof. A zero-knowledge proof is a cryptographical method by which one party, the prover, can prove to another party, the verifier that the prover knows a value x, without revealing any information other than the fact that the verifier knows the value x. A simple example which was first demonstrated live by Konstantinos Chalkias and Mike Hearn[20]. Using the example of "Two balls and the colour-blind friend", the ZKP works as follows: The prover has two balls, one red and one green, and otherwise identical. The verifier (the friend) is colour-blind. To prove that they are in fact differently coloured, you give the balls to your friend, who hides them behind his back. Your friend then decides whether to switch the balls between hands or not, and then reveals one of the balls. The prover declares if the balls were switched.

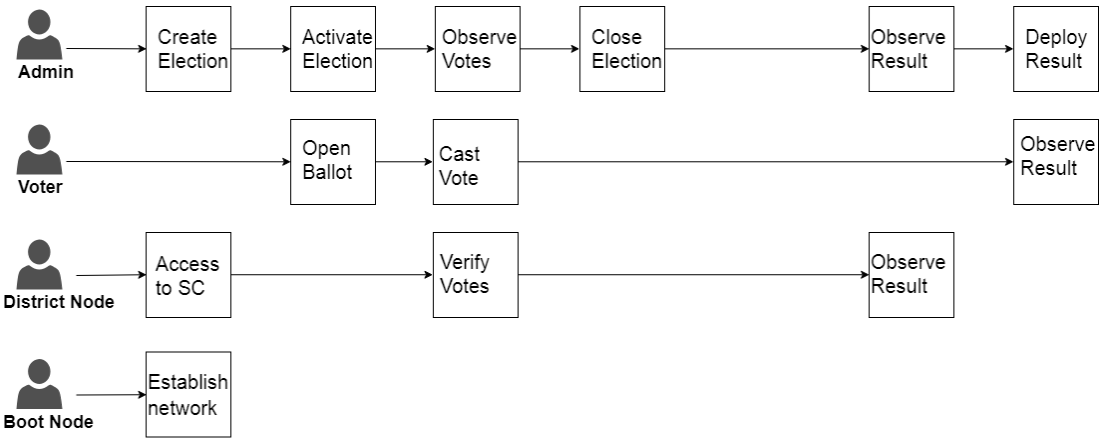

Fig. 1: Election roles and process

By repeating this process, the prover can prove that he can correctly identify the balls, as the verifier confirms that the likelihood of repeated success is halved each time.

A non-interactive zero-knowledge proof, or NIZKP for short, is a variant of zero-knowledge proofs in which there is no interaction between the prover and verifier. Blum, Feldman and Micali[21] showed that a common reference string shared between the prover and verifier is enough to achieve computational zero-knowledge without requiring interaction. The Fiat-Shamir heuristic[22] however showed that NIZKPs could also be obtained in the random oracle model, which in practice can be used as a cryptographic hash function instead[23] which enables any user to prove their identity and the authenticity of their message without a shared public key. This scheme is ideally suited for microprocessor-based devices such as smart cards, personal computers and remote control systems. The Fiat-Shamir heuristic therefore provides a simple yet efficient and secure method to authenticate and verify eligible individuals for a voting system while guaranteeing voters privacy.

III. Blockchain as a Service for E-Voting

In this paper, we consider existing electronic voting systems, blockchain-based and non-blockchain-based, and evaluate their respective feasibility for implementing a national e-voting system (see section VI). Based on this, we devised a blockchain-based electronic voting system, optimizing for the requirements and considerations identified. In the following subsection, we start by identifying the roles and component for implementing an e-voting smart contract then, we evaluate different blockchain frameworks that can be used to realize and deploy the election smart contracts. In the last subsection, we will discuss the design and architecture of the proposed system.

A. Election as a Smart Contract

Defining a smart contract includes identifying the roles that are involved in the agreement (the election agreement in our case) and the different components and transactions in the agreement process. We start by explaining the election roles followed by the election process.

1) Election Roles: As can be seen in Figure 1, elections in our proposal enable participation of individuals or institutions in the following roles. Where multiple institutions and individuals can be enrolled to the same role.

(i) **Election administrators:** Manage the lifecycle of an election. Multiple trusted institutions and companies are enrolled with this role. The election administrators specify the election type and create aforementioned election, configurate ballots, register voters, decide the lifetime of the election and assign permissioned nodes.

(ii) **Voters:** For elections to which they are eligible for, voters can authenticate themselves, load election ballots, cast their vote and verify their vote after an election is over. Voters can be rewarded for voting with tokens when they cast their vote in an election in the near future, which could be integrated with a smart city project.

(iii) **District nodes:** When the election administrators create an election, each ballot smart contracts, representing each voting district, are deployed onto the blockchain. When the ballot smart contracts are created, each of the corresponding district nodes are given permission to interact with their corresponding ballot smart contract. When an individual voter casts his vote from his corresponding smart contract, the vote data is verified by all of the corresponding district nodes and every vote they agree on are appended onto the blockchain when block time has been reached.

(iv) **Bootnodes:** Each institution, with permissioned access to the network, host a bootnode. A bootnode helps the district nodes to discover each other and communicate. The bootnodes do not keep any state of the blockchain and is ran on a static IP so that district nodes find its peers faster.[27]

2) Election Process: In our work, each election process is represented by a set of smart contracts, which are instantiated on the blockchain by the election administrators. A smart

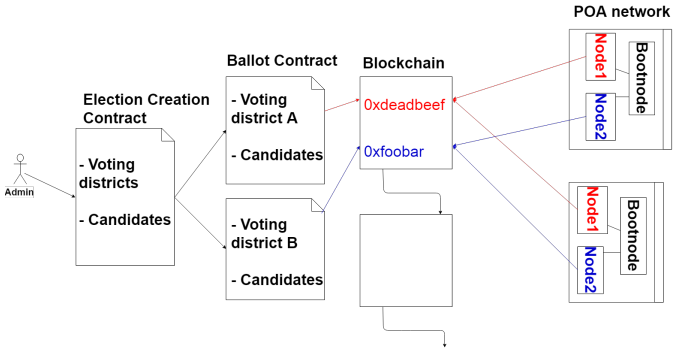

Fig. 2: Election as a smart contract

contract is defined for each of the voting districts of the election so multiple smart contracts are involved in an election. For each voter with its corresponding voting district location, defined in the voters registration phase, the smart contract with the corresponding location will be prompted to the voter after the user authenticates himself when voting.

The following are the main activities in the election process:

(i) **Election creation** Election administrators create election ballots using a decentralized app(dApp). This decentralized app interacts with an election creation smart contract, in which the administrator defines a list of candidates and voting districts. This smart contract creates a set of ballot smart contracts and deploys them onto the blockchain, with a list of the candidates, for each voting district, where each voting district is a parameter in each ballot smart contract. When the election is created, each corresponding district node is given permission to interact with his corresponding ballot smart contract (See Figure 2).

(ii) **Voter registration** The registration of voter phase is conducted by the election administrators. When an election is created the election administrators must define a deterministic list of eligible voters. This requires a component for a government identity verification service to securely authenticate and authorize eligible individuals. Using such verification services, each of the eligible voter should have an electronic ID and PIN number and information on what voting district the voter is located in. For each eligible voter, a corresponding wallet would be generated for the voter. The wallet generated for each individual voter should be unique for each election the voter is eligible for and a NIZKP could be integrated to generate such wallet so that the system itself does not know which wallet matches an individual voter.

(iii) **Vote transaction** When an individual votes at a voting district, the voter interacts with a ballot smart contract with the same voting district as is defined for any individual voter. This smart contract interacts with the blockchain via the corresponding district node, which appends the vote to the blockchain if consensus is reached between the majority of the corresponding district nodes. Each vote is stored as a transaction on the blockchain whereas each individual voter receives the transaction ID for their vote for verifying purposes (see "Verifying vote" section). Each transaction on the blockchain holds information about whom was voted for, and the location of aforementioned vote. Each vote is appended onto the blockchain by its corresponding ballot smart contract, if and only if all corresponding district nodes agree on the verification of the vote data. When a voter casts his vote, the weight of their wallet is decreased by 1, therefore not enabling them to vote more than once per election. As can be seen in Table I, a single transaction on the public Ethereum blockchain includes the transaction ID, the block which the transaction is located, the age of the transaction, the wallet which sent the transaction and who received it, the total value which was sent and the transaction fee. A transaction in our proposed system doesn't require all of this information, a single transaction only has information of the transaction ID, the block which the transaction is located at, to which smart contract the transaction was sent, in this example N1SC indicates that the vote was sent from the N1 district. Finally the value of the transaction is the data which was selected to cast, D therefore indicates that the vote casted in this transaction was the the party D. A transaction in our system (see Table II) therefore reveals no information about the individual voter who casted this particular vote. The age of a single transaction is excluded to protect individual voters from a timing attack.

TABLE I: Example of an public transaction(Ethereum)

TxHash	**Block**	**Age**	**From**	To	Value	[TxFee]
0xdead...	1337	33 sec ago	0xbeef...	Token	10 Ether	0.087
0xface...	1337	33 sec ago	0x4242...	0x1234...	1 Ether	0.056

TABLE II: Example of an transaction in our system

TxHash	**Block**	**To**	**Value**
0xdeadbeef...	1337	N1SC	D
0xG1345edf...	1330	N2SC	P

(iv) **Tallying results** The tallying of the election is done on the fly in the smart contracts. Each ballot smart contract does their own tally for their corresponding location in its own storage. When an election is over, the final result for each smart contract is published.

(v) **Verifying vote** As was mentioned earlier, each individual voter receives the transaction ID of his vote. Each individual voter can go to his government official and present their transaction ID after authenticating himself using his electronic ID and its corresponding PIN. The government official, utilizing district node access to the blockchain, uses the blockchain explorer to locate the transaction with the corresponding transaction ID on the blockchain. The voter can therefore see his vote on the

blockchain, verifying that it was counted and counted correctly.

B. Evaluating Blockchain as a Service for E-Voting

Table III shows a comparison between the three blockchain frameworks that we consider for implementing and deploying our election smart contracts. Those are Exonum, Quorum and Geth.

1) Exonum: Looking at the Exonum blockchain, it is robust end to end with its full implementation done with the programming language Rust. Exonum is built for private blockchains. It has a customized Byzantine algorithm that is used to achieve consensus in the network. With that consensus algorithm, Exonum can support up to 5000 transactions per second. Unfortunately, the limitation of the framework is that Rust is the only programming language in the current version, which limits the developers to the constructs available in that language. To solve this limitation, Exonum is planning to introduce Java-bindings and platform-independent interface description to make Exonum more developer-friendly in the near future.

2) Quorum: Is an Ethereum-based distributed ledger protocol with transaction/contract privacy and new consensus mechanisms. It's a Geth fork and is updated in line with Geth releases. Quorum changed up the consensus mechanism and aimed more towards consortium chain based consensus algorithms. Using this consensus allows it to support from dozens to hundreds transactions per second.

3) Geth: Go-Ethereum or Geth is one of three orginal implementations of the Ethereum protocol and it runs smart contract applications exactly as programmed without possibility of downtime, censorship, fraud or third party interference [4][24]. This framework supports develmopment beyond the Geth protocol, and is the most developer-friendly framework of the frameworks we evaluated. The transaction per second(transaction rate) is dependent on whether the blockchain is implmented as a public or private network. Because of these capabilities, Geth was the framework we chose to base our work on, any similar blockchain framework with the same capabilities as Geth should be considered for such systems.

C. Design and Implementation

To introduce a method of secure authentication, our proposed system is designed to use electronic ID authentication via Auðkenni[25], which is an Icelandic service provider for identity verification. Auðkenni utilizes the Nexus software and RFID scanners. When a user registers for an electronic ID, a user chooses a PIN number for its corresponding ID consisting of 6 numbers. A user will therefore identify himself in the voting booth by scanning his ID and providing his corresponding PIN number to authenticate himself to the system.

1) Any computer in any voting district can be used by any eligible voter to vote, since the wallet for the corresponding voter has information on which voting district the voter is supposed to vote from. For a user to successfully authenticate, a valid ID and PIN number needs to be presented at a voting district using a card reader and the nexus software.
2) If the authentication is successful, the corresponding smart contract is prompted for the ongoing election. The ballot for the aforementioned election is a smart contract which has a list of the candidates a voter can choose from.
3) When a voter has selected a candidate and casts his vote, the voter proceeds to sign his vote by re-entering the corresponding PIN number for his electronic ID.
4) After the voter has signed his vote, the vote data proceeds to be verified by the corresponding district node, which the voter is interacting with the smart contract through. If the aforementioned district node accepts the vote data, the vote data must be agreed upon by the majority corresponding district node.
5) If the majority of district nodes agree upon the vote data, consensus for the particular vote has been reached. The user then receives the transaction ID for the corresponding transaction of his vote in the form of a QR-code and the

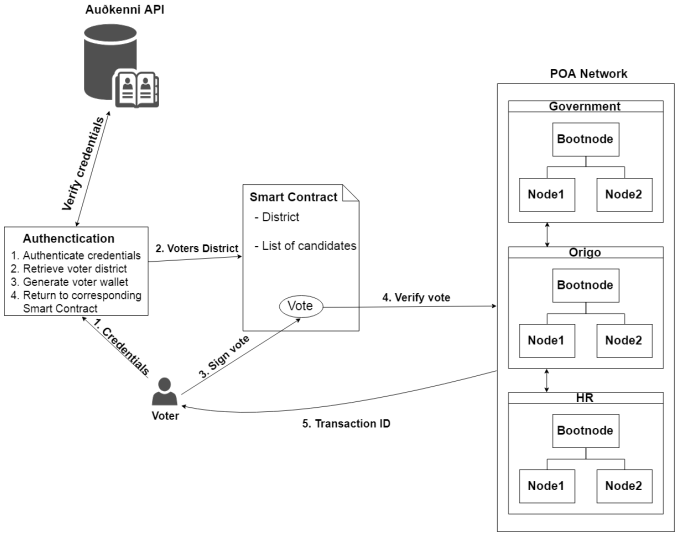

Fig. 3: Voter authenticates himself and casts vote

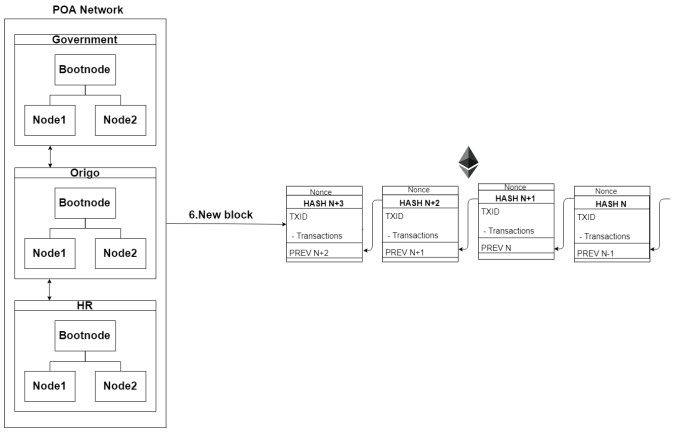

Fig. 4: Block added to the blockchain

TABLE III: Framwork Evaluation

	Exonum	**Quorum**	**Go-Ethereum**
Consensus	Custom-built BFT algorithm	QuorumChain, IBFT and Raft-based consensus	PoW, PoS and PoA
Transactions p/s	up to 5000 transactions p/s	Dozens to hundreds	Depends
Private support	Yes	Yes	Yes
Smart Contract Language	Rust	Solidity	Solidity
Programming Language	Rust	Go, C, JavaScript	Go, C, Javascript
Decentralized	Yes	Partially	Optional

option to print the transaction ID. When the vote is casted and has been verified, a function in the smart contract adds one vote to the party which was voted for. This functionality of the smart contract structure is utilized to determine the election result in each of the voting districts. Figure 3 is a visual representation of the steps we just ellaborated.

6) All transactions which were received and verified in the ongoing block time are deployed onto the blockchain after the block time has reached its time limit (see Figure 4). With each new block added to the blockchain, each district node updates his copy of the ledger.

D. Index of functionalities

Below, we will elaborate the functionalities of a novel ballot and election smart contract for an e-voting system, without the integration of a government indentity verification service.

```
contract ElectionCreation {
    address[] public deployedBallots;
    constructor (bytes32[] candidates,
    bytes32[] district, uint hours) public {
        for(uint i = 0; i < district.length;
        i++){
            address newBallot =
            new Ballot(candidates,
            district[i], msg.sender, hours);
            deployedBallots.push(newBallot);
        }
    }
    function getDeployedBallots()
    public view returns(address[]) {
        return deployedBallots;
    }
}
```

- ElectionCreation constructor: Takes in a list of candidates and districts along with the address of the wallet of the creator and the amount of hours the election will take. The constructor then creates a single smart contract for each district provided and puts the address of each smart contract created into the deployedBallots array.
- getDeployedBallots: returns an array with the address of each created smart contract in each index of the array.

```
contract Ballot {
    struct Candidates {
        bytes32 name;
        uint voteCount;
        uint creationDate;
        uint expirationDate;
    }
    Candidates[] public candidates;
    address public manager;
    bytes32 public votingDistrict;
    mapping(address => bool) public voters;

    modifier restricted() {
        require(msg.sender == manager);
        _;
    }
    constructor (bytes32[] candidateNames,
    bytes32 district, address creator,
    uint amountOfHours) public {
        manager = creator;
        votingDistrict = district;
        for (uint i = 0;
        i < candidateNames.length; i++) {
            candidates.push(Candidates({
                name: candidateNames[i],
                voteCount: 0,
                creationDate: now,
                expirationDate: now +
                amountOfHours
            }));
        }
    }
```

- Restricted modifier: This modifier is used to restrict functions in the manner of that only the creator of the election can access the information which the functions give.
- Ballot constructor: Sets the manager of the ballot smart contract to the address of the wallet which created the election, the voting district of the smart contract to the district which the ElectionCreation contract provided and then proceeds to fill the Candidates struct with the list of candidates provided and the number of votes for each candidate to 0. The constructor also stores the time of the creation of the contract along with the time when the contract is to expire.

```
    function vote(uint candidate) public{
        require(!voters[msg.sender]);
        if(now >
        candidates[candidate].expirationDate){
            revert();
        }
        candidates[candidate].voteCount += 1;
```

```
        voters[msg.sender] = true;
    }
```

- vote: This function allows voters to vote. The requirement for a voter to vote, is that the mapping of the address of the voter is set to its default, false. If that is the case, the function guarantees that the election time limit has not been reached. If both requirements are satisfied, the contract retrieves the index of which candidate was voted for and increases his vote count by 1 and sets the mapping to true, so that the voter can never vote again in this particular election.

```
    function getCandidateName(uint index)
    public restricted view
            returns (bytes32)
    {
            require(now >
        candidates[candidate]
        .expirationDate)
        }
        return candidates[index].name;
    }
    function getVoteCount(uint index)
    public restricted view
            returns (uint)
    {
            require(now >
        candidates[candidate]
        .expirationDate)
        return candidates[index].voteCount;
    }
}
```

- getCandidateName & getVoteCount: Both these functions retrieve the name and amount of votes a candidate has recieved from an index. These functions classify as helper functions to determine the election results after the election is finished

IV. Security analysis and Legal issues

In this section we analyze the security of the proposed e-voting system and the main legal issues.

A. Security analysis

1) DDoS: To successfully DDos a distributed system such as we have proposed, the attacker must DDoS every single bootnode in the private network. The individual or institution would be immediately located if that would occur. Each node is implemented with a Byzantine fault tolerance algorithm, which helps locating failed nodes in the system.

2) Authentication vulnerability: Each individual is identified and authenticated by the system by presenting an electronic ID from Auðkenni and the corresponding 6-digit PIN in the voting booth. Without supervision, an individual could vote for multiple people, if the individual had knowledge of the PIN for each corresponding electronic ID he has. To further address this vulnerability in the near future, a biometric scan could be introduced.

3) Sybil: Sybil attack[26] is known against centralized systems, where an individual creates a large amount of nodes in an attempt to disrupt network operation by hijacking or dropping messages. Since our proposal is running in a private network no individual has the access to create one. Even the consensus protocol that is used in our system is prone Sybil attacks.

Private blockchains solve many of today's security problems using strong cryptography features and the limited access to the ledger, without negating the transparency aspect the blockchain technology offers.

B. Legal issues

1) Remote voting: Remote elections provide no coercion resistance because of the non supervised factor in a remote election. Remote elections can therefore not guarantee the privacy that people have when they cast their vote in a voting booth. Family members or a coercer can watch over your shoulder while you're voting, which could lead to a misconfigured results. If elections are hosted on a website for example it could easily be taken down by people with good hacking skills and the mindset to do so. People could identify themselves as another person and therefore vote for another person and even multiple people.

2) Transparency: In the today's election scheme, no method of transparency can be offered to participants of the election. When an individual places his ballot in the box at his voting district, there is no guarantee from the scheme that his vote was counted and counted correctly. Any individual vote can be misplaced, counted incorrectly because of human error or simply because the party which the voter voted for could be disliked by the individual which counted the vote. This transparency is non-existent because no ballot has information on who casted aforementioned vote. To introduce transparency in the process of an election would require a new law which would allow government officials to provide the services which allow such method of transparency

3) Voter privacy: In every pen and paper election scheme, voters privacy is a key element. The law forbids any individual or entity to be able to know from a single vote, who gave aforementioned vote. If such information could be gathered for each vote, such information could then leak to the public which would allow for listing every single individual who voted for a single party/candidate. To satisfy the privacy of each voter, no individual vote should be traceable back to the voter.

V. Related work

In this chapter we will be examining various research papers and thesis which explored similar fields of study, i.e electronic voting systems.

Anonymous voting by two-round public discussion, proposed an addition of a self-tallying function to the 2-Round Anonymous Veto Protocol (called AV-net). The AV-net provided exceptional efficiency compared to related techniques, the paper was focused on the dining cryptographers network

(DC-net) and its weaknesses and proposed the AV-net as a new way to tackle that problem.[7][8][9]

The new protocol, like the AV-net requires no trusted third party or private channel. Participants execute the protocol by sending two-round public messages, but is significantly more efficient in terms of the number of rounds, computational cost and bandwidth usage. In general, the new protocol divided electronic voting into to two classes:

1) Decentralized elections where the protocol is essentially run by the voters.
2) Centralized elections where trusted authorities are employed to administer the process.

The protocol proposed was focused on the first class, where strong voter privacy was the primary objective which had two challenges. First challenge was that there exists no trusted third party. With a trusted third party, many security problems can be easily solved, but could lead to the 'trusted' third party to become the one who breaks the security policy. The goal therefore was to eliminate the use of a trusted third party altogether. The second challenge was that there would be no voter-to-voter private channels to ensure dispute freeness, i.e everybody could check whether all voters had followed the protocol faithfully.

These challenges were fulfilled in the AV-net, but the new protocol proposed a new solution which solved the downside of the AV-net, heavy computational load for each voter, which increased linearly with the number of voters. The new protocol proved as secure as the AV-net and can be seen as a generalisation of the AV-net protocol, but significantly more efficient. The first round in the two-round protocol consisted of every participant to publish his public key and a zero knowledge proof (ZKP) for his private key. When the round finished, each participant checks the validity of the ZKPs and computes. In the second round, each participant needs to demonstrate that the encrypted vote was one of the valid voting choices without revealing which one. The self-tallying function then works in a way that the public keys of all participants are combined is such a structured way that the random factors of the private keys immediately vanish, thus revealing the tally. There are limitations with this protocol concerning national elections. The protocol requires a collaborative effort from all voters, otherwise the protocol will not work. For example if some voters refuse to send data in the end of round 1, the tallying process will fail, everyone will know who those voters are and can expel them and restart the protocol. This leads to the voting process to be delayed, which would be severely costly for a large scale election. Another limitation is the lack of resistance of coercion because of the remote voting, a voter can be forced to reveal his secret value, therefore revealing how he voted.

A Smart Contract For Boardroom Voting with Maximum Voter Privacy[10], proposed the first implementation of a decentralized and self-tallying internet voting protocol with maximum voter privacy using the Blockchain, called The Open Vote Network (OVN). The OVN is written as a smart contract for the Ethereum blockchain. In its general idea the OVN is an implementation of the Anonymous voting by two-round public discussion we previously discussed.

The creators of the OVN came to the conclusion after implementing the system, that the cost of running such system on the Ethereum blockchain was 0.73$ per voter. The safe upper limit of voters was 50 voters, but the cost could be considered reasonable as it provided maximum voter privacy and is publicly verifiable. The limitation of number of voters was recommended because of the gas limit on the public Ethereum blockchain.

By examining their research paper, the limitations of the previous protocol are unchanged. The OVN does not provide any coercion resistance with the public verifiability in the way that the voting is conducted in a unsupervised environment, i.e the coercer cand stand over the shoulder of the voter. The OVN is also vulnerable to denial of service attack because it is implemented on the Ethereum blockchain, which has had numerous DOS attacks through its lifespan. The Ethereum blockchain could also be throttled by major traffic of transactions at the time of an election, which could delay the voting process immensely. The implementation could therefore be optimal for small boardroom voting, with the downside of having each individual voter downloading the full Ethereum blockchain to confirm the voting protocol is being executed correctly.

A Secure and Optimally efficient Multi-Authority Election Scheme, proposed a multi-authority secret-ballot election scheme which would guarantee privacy, universal verifiability and robustness, where voters would participate using a PC, where the main consideration is the effort required of a voter[11].

In this model, voters cast their vote by posting ballots to a bulletin board. The bulletin board works as a broadcast channel with memory to the extent that any party can access its content but no party can erase anything from the bulletin board. The ballot does not reveal any information on the vote itself but is ensured by an accompanying proof that the ballot contains a valid vote. The final tally, the sum of all votes, which occurs when the deadline is reached, can then be obtained and verified, by any observer, against the product of all submitted ballots. Which would ensure universal verifiability, due to the homomorphic properties of the encryption method used.

While this proposal can scale up to large elections better than the previous ones, it does have limitations. The proposal doesn't provide any coercion resistance, as the election takes place on a PC at home, like the Ethereum smart contract, the coercer can stand over the shoulder of the voter.

Netvote[12] is a decentralized blockchain-based voting network on the Ethereum blockchain. Netvote utilizes decentralized apps for the user interface of the system. The Admin dApp allows election administrators to set election policies, create ballots, establish registration rules and open and close voting. The Voter dApp is used by individual voters for registration, voting and can be integrated with other devices(such as

biometric readers) for voter identification. The Tally dApp is then used to tally and verify election results. Netvote supports three types of elections:

1) Open Election: Anyone may vote
2) Private Election: Only authenticated and authorized individuals may vote
3) Token-Holder Elections: Only voters who operate accounts that have a balance of a designated compliant token may vote.

In Netvote, each election is represented by a set of smart contracts which is instantiated on the Ethereum blockchain by an election administrators through the Admin dApp. Each ballot smart contract references one ballot. Multiple ballots can be listed together through a voter pool smart contract where each voter pool smart contract can f.x. represent each polling place. An individual voter therefore registers at the polling place and then interacts with the voter pool smart contract through the voter dAapp. In a private election, Netvote utilizes a so called Vote Gateway to provide voters privacy. Through the voter dApp, each individual voter transmits a cryptographically signed vote token to the Vote Gateway for verification. The Vote Gateway then retrieves the Vote ID secret key from a vault, where the secret key is specific to each election and is destroyed when voting closes. The Vote Gateway then does a SHA3 hash of the vote token with the secret key to generate an anonymous Vote ID. The Vote Gateway then submits the vote payload in a blockchain transaction to the voter pool smart contract with the vote payload mapped to the anonymous Vote ID. This Vote Gateway is used in a manner of a Zero-knowledge proof to guarantee voters privacy.
Netvote is an optimal solution for national election, while there is a limitation concerning scaleability since netvote utilizes the public Ethereum blockchain which we discussed earlier can be throttled and have immense traffic of transaction while the election is taking place.
Agora[13] is an end-to-end verifiable blockchain based voting solution designed for governments and institutions. Agora uses their own Token on the blockchain for elections, where governments and institutions purchase these tokens for each individual eligible voter. This voting system is a multi-layer architecture which include the blockchain, called the Bulletin Board, which is based on the Skipchain architecture. The data on the Bulletin Board is cryptographically tied to the Bitcoin blockchain through the Cotena layer, which provides a high level of immutability and decentralization of the data. The Bulletin Board utilizes permissioned collective authority nodes, which confirm transactions, where each node in the network maintains a copy of all transactions and approves them into blocks as part of the networks consensus mechanism. The Skipchain architecture[14] provides a proactive Byzantine consesus mechanism. This architecture enables software clients to navigate arbitrarily long blockchain timelines both forward and backward, providing proof of transaction validity without the need for a full record of the blockchain. The Cotena schema[15] is a tamper-resistant logging mechanism built on top of the Bitcoin blockchain. This layer links the Bulletin Board and supporting cryptographic proofs to the Bitcoin blockchain, which provides decentralized immutability to the permissioned layers data. Cotena was created to leverage the data security of the Bitcoin blockchain while introducing a design that has minimal data storage requirements and reduced Bitcoin transaction costs. Agora's voting process consists of six distinct steps:

1) Configuration: Election administrators create a new election event.
2) Casting: Voters cast their encrypted ballots to Agora's network.
3) Anonymization: Agora's network anonymizes all voter ballots.
4) Decryption: Agora's network decrypts the anonymized ballots.
5) Tallying: All votes are counted.
6) Auditing: Auditors post their reviews confirming validity of the election results.

Agora's voting system was partially used for the presidential election in Sierra Leone on march 7th in 2018[16]. In this election representatives of Agora went to poll stations and manually registered the vote data of paper ballots onto their blockchain to do a private tally of the votes. Agora is also an optimal blockchain-based electronic voting system solution, the only limitation found of this system is that Agora is a token-based solution, for every eligible voter, a token must be purchased to enable every voter with a voting right. This can lead to cost inefficiency with low voter turnout, f.x. if a government purchases 3 million tokens for 3 million eligible voters, but the voter turnout is 50%, then 1.5 million tokens would have sufficed. There would be no possibility for the government to purchase only 1.5 million tokens, since that would lead to the individuals that would not vote to not having a right to vote, which is illegal and predicting voter turnout precisely is impossible.

VI. Conclusion

The idea of adapting digital voting systems to make the public electoral process cheaper, faster and easier, is a compelling one in modern society. Making the electoral process cheap and quick, normalizes it in the eyes of the voters, removes a certain power barrier between the voter and the elected official and puts a certain amount of pressure on the elected official. It also opens the door for a more direct form of democracy, allowing voters to express their will on individual bills and propositions.

In this paper, we introduced a unique, blockchain-based electronic voting system that utilizes smart contracts to enable secure and cost efficient election while guaranteeing voters privacy. We have outlined the systems architecture, the design, and a security analysis of the system. By comparison to previous work, we have shown that the blockchain technology offers a new possibility for democratic countries to advance from the pen and paper election scheme, to a more cost- and time-efficient election scheme, while increasing the security

measures of the todays scheme and offer new possibilities of transparency. Using an Ethereum private blockchain, it is possible to send hundreds of transactions per second onto the blockchain, utilizing every aspect of the smart contract to ease the load on the blockchain. For countries of greater size, some measures must be taken to withhold greater throughput of transactions per second, for example the parent & child architecture[28] which reduces the number of transactions stored on the blockchain at a 1:100 ratio without compromising the networks security. Our election scheme allows individual voters to vote at a voting district of their choosing while guaranteeing that each individual voters vote is counted from the correct district, which could potentially increase voter turnout.

REFERENCES

[1] Sos.ca.gov. (2007). *Top-to-Bottom Review | California Secretary of State. Available at: http://www.sos.ca.gov/elections/voting-systems/oversight/top-bottom-review/*.

[2] Nicholas Weaver. (2016). *Secure the Vote Today. Available at:https://www.lawfareblog.com/secure-vote-today.*

[3] TechCrunch, (2018). *Liquid democracy uses blockchain to fix politics, and now you can vote for it* [Online]. Available at: https://techcrunch.com/2018/02/24/liquid-democracy-uses-blockchain/

[4] Geth.ethereum.org. (2018). *Go Ethereum.* Available at: https://geth.ethereum.org/

[5] Vitalik Buterin. (2015). *Ethereum White Paper. Available at: https://github.com/ethereum/wiki/wiki/White-Paper.*

[6] Nca.tandfonline.com. (2015). *Pirates on the Liquid Shores of Liberal Democracy: Movement Frames of European Pirate Parties*. [Online]. Available at: https://nca.tandfonline.com/doi/abs/10.1080/13183222.2015.1017264#.Wr0zCnVl8YR

[7] Feng Hao, P.Y.A. Ryan and Piotr Zielinski. (2008). *Anonymous voting by two-round public discussion.* Available at: http://homepages.cs.ncl.ac.uk/feng.hao/files/OpenVote_IET.pdf

[8] Feng Hao and Piotr Zielinski. *A 2-Round Anonymous Veto Protocol* Available at: http://homepages.cs.ncl.ac.uk/feng.hao/files/av_net.pdf.

[9] The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Available at: https://users.ece.cmu.edu/~{}adrian/731-sp04/readings/dcnets.html.

[10] Patrick McCorry, Siamak F. Shahandashti and Feng Hao. (2017). *A Smart Contract for Boardroom Voting with Maximum Voter Privacy* Available at: https://eprint.iacr.org/2017/110.pdf.

[11] Ronald Cramer, Rosario Gennaro and Berry Schoenmakers. *A Secure and Optimally Efficient Multi-Authority Election Scheme* Available at: http://www.win.tue.nl/~berry/papers/euro97.pdf

[12] Jonathan Alexander, Steven Landers and Ben Howerton (2018). *Netvote: A Decentralized Voting Network* Available at: https://netvote.io/wp-content/uploads/2018/02/Netvote-White-Paper-v7.pdf

[13] Agora (2017). *Agora: Bringing our voting systems into the 21st century* Available at: https://agora.vote/Agora_Whitepaper_v0.1.pdf

[14] Kirill Nikitin, Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, Justin Cappos and Bryan Ford (2017). *CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds* Available at: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-nikitin.pdf

[15] Alin Tomescu and Srinivas Devadas(2017). *Catena: Efficient Non-equivocation via Bitcoin* Available at: https://people.csail.mit.edu/alinush/papers/catena-sp2017.pdf

[16] Michael del Castillo (2018). *Sierra Leone Secretly Holds First Blockchain-Audited Presidential Vote* Available at: https://www.coindesk.com/sierra-leone-secretly-holds-first-blockchain-powered-presidential-vote/

[17] Ethereum Blog. (2018). *On Public and Private Blockchains - Ethereum Blog.* Available at: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

[18] Bitfury.com. (2018). *Digital Assets on Public Blockchains* Available at: http://bitfury.com/content/5-white-papers-research/bitfury-digital_assets_on_public_blockchains-1.pdf

[19] Steve Ellis, Ari Juels and Sergey Nazarov. (2017). *ChainLink: A Decentralized Oracle Network* Available at: https://link.smartcontract.com/whitepaper

[20] Konstantinos Chalkias, (2017). *Demonstrate how Zero-Knowledge Proofs work without using maths* Available at: https://www.linkedin.com/pulse/demonstrate-how-zero-knowledge-proofs-work-without-using-chalkias

[21] Manuel Blum, Alfredo De Santis, Silvio Micali and Giuseppe Persiano (1988). *Non-Interactive Zero-Knowledge* Available at: https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Zero\%20Knowledge/Noninteractive_Zero-Knowkedge.pdf

[22] Amot Fiat and Adi Shamir. (1986). *How to Prove Yourself: Practical Solutions to Identification and Signature Problems* Available at: https://www.math.uni-frankfurt.de/~dmst/teaching/SS2012/Vorlesung/Fiat.Shamir.pdf

[23] Mihir Bellare & Phillip Rogaway (1995). *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols* Available at: https://cseweb.ucsd.edu/~mihir/papers/ro.pdf

[24] Ethdocs.org. (2018). *What is Ethereum? — Ethereum Homestead 0.1 documentation*. [online] Available at: http://ethdocs.org/en/latest/introduction/what-is-ethereum.html

[25] Auðkenni.is (2018). [Online] Available at: https://www.audkenni.is/en/

[26] Vincent Gramoli. (2018). *On the Danger of Private Blockchains*. [Online] Abailable at: https://www.zurich.ibm.com/dccl/papers/gramoli_dccl.pdf

[27] Salanfe, "Setup your own private Proof-of-Authority Ethereum network with Geth", Hacker Noon, 2018. Available at: https://tinyurl.com/y7g362kd.

[28] Jelurida, "Jelurida", 2017. Available at: https://www.jelurida.com/sites/default/files/JeluridaWhitepaper.pdf