Blockchain Voting: Implementation and Analysis.

Kimberly Villalobos, Christian Altamirano and Rishabh Chandra

***Abstract*— In this paper we provide a security analysis of the blockchain-based voting system proposed by HjÃąlmarsson and Hreiðarsson [1]. For this, we take the role of administrators of an election and implement a decentralized voting application. We then explain the implementation details and expose some of the possible attacks that make the system not optimal for running national elections. Lastly, we compare the security risks of this system to those of other online voting systems.**

I. INTRODUCTION

Electronic voting systems require a certain amount of certainty that is not even expected in other high-sensitive cybersecurity applications. This is mainly due to the architectural and philosophical challenges around the concept of voting. In most secure systems, there are very few users who actually operate it with credentials, in a centralized location (think major server for a corporation, crime record database for the FBI, etc). Voting systems necessarily need a high level of distribution, such that there are millions of voting machines operated by poll workers without the technical background to accurately assess the security vulnerabilities in the system. In addition, with voting comes the ability to influence policy and social outcomes for millions, if not billions of people. The stakes in a major election in a country like Brazil, India, or the United States are extremely high, and the economic benefit to a successful hack might literally be on the order of trillions of dollars (for example, if one candidate is more favorable to your corporation than another and is looking to give corporate tax incentive). This combination of high stakes and distributed control makes voting systems particularly difficult to secure electronically. The following are considered essential requirements that any voting system must satisfy:

1) Only eligible voters should be able to vote, and they can vote at most once. There needs to be, in other words, some form of authentication available for the system.

2) Each vote must be secret - i.e. it is impossible to determine how a voter voted in order to prevent bribing, extortion, or blackmail on the part of hostile actors.

3) The system must work under the assumption that adversaries with a lot of power exist and can try to attack the system.

4) The results of an election must be verifiable by auditing ballots or using some other statistical techniques that do not depend on the correctness of a software.

5) The system must have no part of the operation that is considered "trusted" - i.e. no actor or piece of software can be in possession of a secret key, or special administrative power that cannot be checked.

Paper-based voting systems have been the most common method used world-wide for running national elections. Voters usually need to go to a voting center, present a government ID to be authenticated and then proceed to cast their vote on a a paper ballot that is completely anonymous. The main advantage of this system is that it truly provides secret ballots: once a voter submits their vote it is basically impossible to differentiate his or her vote from the rest.

However, paper ballots do not necessarily satisfy all requirements listed above. In particular, a corrupt party with access to the ballots could insert multiple paper-ballots with fake votes and while it is possible to find out that extra votes were included, the system does not allow to verify which ballots are the invalid ones. Or in the opposite side, an adversary could destroy paper ballots before they are counted and the votes could not be retrieved.

Many people have tried to develop new voting systems that solve the disadvantages that paper ballots have; however, the main motivation for designing new systems seems to be solving the inefficiencies of the system like the manual work required or the amount of time needed to know the results, as opposed to solving its security concerns. Consequently, the incorporation of new technologies to automate and speed up the system is considered by many the perfect solution.

A. Internet Voting

Internet Voting refers to election systems in which ballots are electronic and votes are cast on a machine (computers, cellphones, etc). These ballots are sent to and from the voting device via web or email. Multiple forms of Internet voting have been designed, and while it has a significant amount of supporters, security experts believe that these systems should not replace paper-based ones since the former systems introduce multiple security risks.

Supporters of online voting believe in some, if not all of the following advantages of incorporating this technology in the election process:

- Election costs could be reduced: less people would need to be hired and there would be no need to print ballots or buy envelops.
- Simplifies the counting process and gives accurate results: a simple piece of code can quickly compute the total number of votes per candidate as opposed to waiting for people to manually count votes.
- Makes the system more accessible: Citizens could easily vote while abroad since the internet is ubiquitous.

However, security experts believe that implementing internet voting for important elections is very irresponsible since the system would be exposed to several threats that come inherently with the use of the internet. The major concerns are

- The entity in charge of the system might have enough power to corrupt the system: whoever has control over the software running the system could potentially change a vote without leaving any evidence. This would violate the requirements 1), 3), 4) and maybe 5) of voting systems.
- The system could be attacked: Online systems are vulnerable to cyber attacks that can either affect the functionality of the system, leak confidential information or modify the behavior of the system in unexpected ways that can be difficult to identify. This would violate the requirements 3) and 4) of voting systems.
- These systems might ease traceability of votes to voters: The voting system itself could contain information that reveals information about the voter like identification number, passwords or even the vote itself. This would violate the requirements 2) and 3) of voting systems.
- Machines used for voting can be hacked: This is perhaps the biggest concern since there is not much administrators of an election could do to prevent devices from being victims of hackers. If a machine used for voting is infected with malware the votes can be tampered or trace independently of how secure the voting system is. This would violate the requirements 2) and 3) of voting systems.
- Machines could fail to authenticate identities: If for example authentication relies on passwords or other secret information, adversaries could steal that information from voters and use it to vote with their identity. This would violate the requirements 1), 2) and 3) of voting systems.

The first governmental election that used Internet voting in the United States was the presidential election of the Reform Party in 1996 [7]. Since then, internet voting started to become a more popular idea but with its popularity more security risks also emerged. In 2000, the Democratic Party hired a private company to run its presidential primary election in Arizona with an internet voting system [6]. As a result, multiple security threats including denial of service attacks, voter identity thefts and hacking of voting machines were ubiquitous. The dangers of such an election were strongly exposed and a few years later the standards of voting systems

were strengthened by the Federal Election Commission of the U.S.

Currently, several states allow online voting through web portals or email. However, the majority of the citizens still do not approve this voting methods since votes cast in this form are exposed to security attacks from the administrators of an election, other citizens or even foreign adversaries.

II. BLOCKCHAIN-BASED VOTING SYSTEM [BB]

The blockchain techonolgy keeps a list of records that are very difficult to modify. Specifically, the blockchain data structure is a public distributed ledger formed by appending blocks that are linked as a chain: each block contains a hash value that is a function of the previous block. This design assures immutability of the ledger, since modifying the data in one of the blocks would immediately create inconsistencies in the hashes of the descendent blocks.

In a blockchain implementation, writes to the ledger are only allowed if the nodes of the network reach concensus and approve the write. Read accesses to the ledger are denoted as *calls* and write accesses as *transactions*. Transactions are then monitored and the consensus protocol used to approve or reject transactions varies with each implementation. These characteristics make the blockchain technology a distributed and decentralized system that does not grant complete power to any party individually but instead gives equal partial power to multiple parties.

The fact that data deployed onto the blockchain can never be modified and that there is no central entity in control of the ledger has naturally make many people believe that this technology could be the solution to the security problems present in online voting systems. And while it makes sense that appending votes onto the blockchain can protect votes from being modified, there is a lot of work that needs to be done to fill the gaps: how would authentication be successfully achieved? How would cyberattacks be prevented? Who should create and have control over the blockchain network? How do we guarantee that votes are secret?

In this section we will describe and analyze the election design that was proposed by HjÃąlmarsson and Hreiðarsson to implement a blockchain technology as part of a voting system [1]. We have taken the role of administrators of an election and we have implemented a voting web application following their design in order to better understand their proposal as well as to explore possible security attacks of the system. This design consists of smart contracts that are deployed onto a private blockchain and allow to represent votes as irreversible transactions.

A. Blockchain Network

Blockchains have several types of access control. For this implementation we need a private blockchain network in which the nodes of the network represent a partition of the voting population. While the design by HjÃąlmarsson and Hreiðarsson specifically focuses on liquid democracies and considers one node per voting district, we here generalize this approach by simply considering any partition of the voters, which could be more granular or more general than a district-based partition. In addition, in this design the nodes are assumed to be trusted parties from each partition set with high computing power.

The main advantage of private blockchains is that they restrict both read and write accesses to specific participants. As a consequence, private blockchains are composed of a limited number of fixed nodes, which provides other benefits like making transactions cheaper and faster as they need to be validated only by a few number of nodes. We then consider a network with a set of k nodes

$$N = \{n_1, ..., n_k\}$$

that are in one-to-one correspondence with the elements of a partition

$$P = \{P_1, ..., P_k\}$$

of the set of eligible voters for the election. The exact value of k for a specific election as well as the exact partition sets depend on the level of distribution desired for the system as well as other considerations like the physical location of voters.

Bootnodes: Besides the nodes in N, which keep an identical copy of the ledger and can read the blocks

or write on them, our network must also include other type of nodes called *bootnodes*, which do not keep the state of the blockchain. The unique purpose of these nodes is to allow the nodes in N (the partition nodes) to discover each other and keep the same copy of the ledger.

Concensus Protocol: The Proof of Authority [PoA] is the consensus protocol implemented for this blockchain network. Unlike the more common Proof of Work protocol used for the bitcoin blockchain, in which the nodes must solve complicated puzzles in order to append transactions to the blockchain and get bitcoin rewards, the PoA protocol relies on validator nodes that get payed to verify the validity of a transaction. Specifically, when a transaction is requested, each node that is declared as a validator in the blockchain network proceeds to either approve it or reject it. The transaction is then appended onto the ledger if and only if the majority of the validators approve it.

In this implementation, all nodes in N are declared as validators and therefore a vote transaction is successfully appended onto the blockchain if an only if the majority of the partition nodes approve it. Standard transactions include information about the sender, the receiver and the timestamp of the transaction; however, transactions in this design must only include the transaction ID, the block number, the district from which the vote was cast and the value of the vote. We also clarify that while it is possible that this partition nodes become corrupted and validate fraudulent votes, the reputation of these validators is at stake and at least half of them would need to become corrupt to approve a bad transaction.

B. Wallets

Users of a blockchain network need to own at least one account in the blockchain. An account consists of a public key that is used to identify transactions to and from the user, as well as a private key that the users use to prove that an account is indeed theirs. Because public keys tend to be very large, blockchain uses shorter strings called addresses that are representative forms of the public key for an account. A wallet is a common program used for managing users' accounts and their respective keys.

In order to allow voters in the election to cast their votes as blockchain transactions, the election administrators must create the blockchain network described above as well as one wallet with one account per eligible voter. If the system were able to match accounts to their corresponding owners, this election system would violate requirements 2) and 3) of voting systems. However, the blockchain network needs to somehow verify that a voter is indeed the owner of the account that he or she is using to cast the vote. To circumvent this problem, the use of Zero Knowledge Interactive Proofs can be used to generate and authenticate voter's accounts [1].

Zero Knowledge Proofs [ZKP]: This cryptographical tool allows one party to prove to another party that they know a specific piece of data without revealing any information other than their knowledge of that value. For instance, consider a signature scheme in which the parties involved have a secret key x and a public key g^x. Suppose Alice wants to prove to Bob that g^x is her public key. A round of a Zero Knowledge Proof would go as follows:

- Alice chooses a random number r and sends $m = g^r \mod p$ to Bob.
- Bob asks Alice either for the value of $x + r \mod (p-1)$ or the value of $g^{x+r} \mod (p-1)$.
- Alice sends Bob a value y
- Bob verifies that y satisfies either $g^x \cdot m \equiv g^{y \mod (p-1)}$ or $m \equiv g^y \mod p$ respectively.

If Alice does not know the value of x she has a $\frac{1}{2}$ probability of answering correctly Bob's request. This means that after multiple rounds Bob can confirm that Alice knows x if she answers consistently every time.

Non-Interactive Zero Knowledge Proofs [NIZKP]: This method is a variant of ZKPs in which interaction between the two parties is not necessary. Instead, a reference string that is common and shared between them is enough for reaching Zero Knowledge proof without any rounds of repetitive interactions.

Voter Authentication: In their implementation, HjÃąlmarsson and Hreiðarsson assume that each voter will need to register for the election by physically

presenting a government ID to election administrators. During registration, eligible voters would be assigned an electronic ID and would be prompted to choose a 6-digit PIN for the corresponding ID using a secure service provider for identity verification. This design assumes that NIZKPs are used to generate the wallets of eligible voters and to prove that a specific wallet belongs to a specific voter without revealing the identity of the voter.

Therefore, when a users want to vote they can simply use their electronic ID and PIN to authenticate themselves and get access to their wallet. The wallet must also contain the partition node that will be used by the user to interact with the ballot smart contract corresponding to the set where the voter belongs.

C. Smart Contracts

Smart contracts are pieces of code that self-execute in a decentralized application. The functions contained in the smart contracts must specify the agreements of the contract, which can be deployed onto a blockchain to make those agreements trackable and irreversible. After deployment the code cannot be changed and the parties involved are bound to follow the rules as written. A huge advantage of these contracts is that they are self-verifiable as their code specify requirements that trigger events when those are not satisfied. In addition, calling a public function in the smart contract corresponds to making a transaction in the blockchain. In our case, this means that a function can only be executed if it satisfies the contract's rules and the majority of the validator nodes in the blockchain network successfully approve that the account making the transaction has permission to do so.

In this implementation, smart contracts are used as ballots to enforce the election agreement. Specifically, we want the contract to contain a voting function that can only be called once per valid voter and that is executed as a transaction in the blockchain. Moreover, in order to facilitate the vote count process, we want our contract to provide functionality for checking the total number of votes that a specific candidate has. However, since voters should not have access to partial vote counts before casting their votes, only the administrators of the election should be able to call these functions.

For the system to be distributed and decentralized, as well as to optimize for performance, we use a ballot contract per partition set P_i and we restrict permissions for the i^{th} ballot contract so that only the corresponding node n_i can interact with it. Thus, voters that belong to P_i can execute the vote function in the i^{th} ballot contract by connecting to the network through node n_i. To enforce that voters vote in the center corresponding to the partition set they belong to, the voter's wallet contain information on the node they must interact with to make their vote transaction.

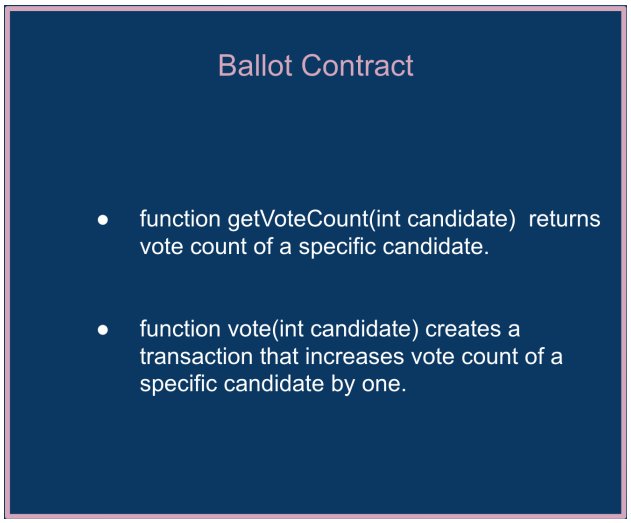

Fig. 1. Functionality of a Ballot contract.

In order to deploy all k ballot contracts at once, we use a factory smart contract that creates as many instances of Ballot contracts as necessary and deploys them onto the blockchain. We modified the factory contract proposed by HjÃąlmarsson and Hreiðarsson in order to make the application easier to implement. Specifically, we added functionality so that functions of a particular ballot contract can be called from the factory contract and not necessarily directly from the Ballot. This simplifies our implementations since nodes only directly interact with a single contract. These modifications do not affect the purpose of the distributed design because nodes can only call these intermediate functions if they have permission to interact with the corresponding ballot contract.

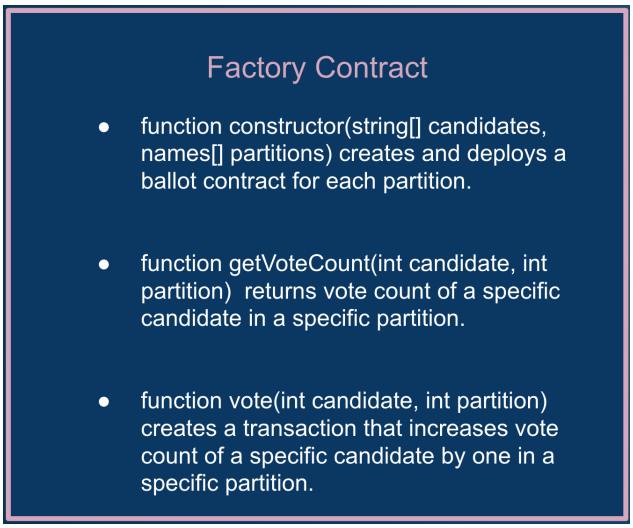

Fig. 2. Functionality of a Factory contract.

D. Implementation Software and Parameters

To run our election as a smart contract application we utilize one of the implementations of the Etherum protocol called Geth. Taking the role of administrators of an election, we used this interface to create our private blockchain with three nodes that implemented the Proof of Authority consensus protocol with a transaction rate of 5 seconds. We used only one bootnode for discovery of nodes since our resources were limited and it was enough for the security analysis of this system design.

Moreover, we wrote our smart contracts with the programming language Solidity and used the Truffle development environment to compile and deploy them onto the blockchain. The Truffle framework also allowed us to interact with the contracts after deployment via web3 in client-side JavaScript.

For this application we created a small election with only 3 candidates, 10 voters and 3 partition sets. We have left out the authentication part in our implementation, but we will assume the NIZKP distribution of wallets as indicated earlier for the security analysis. Similarly, our implementation does not modify the default data included in a transaction but we will analyze the system assuming that timestamps and senders' addresses are not recorded.

```
.
├── accounts.txt
├── boot.key
├── genesis.json
│   ├── election-harmony.json
│   └── election.json
├── node1
│   ├── geth
│   ├── geth.ipc
│   ├── keystore
│   └── password.txt
├── node2
│   ├── geth
│   ├── geth.ipc
│   ├── keystore
│   └── password.txt
├── node3
│   ├── geth
│   ├── geth.ipc
│   ├── keystore
│   └── password.txt
└── truffle
    ├── LICENSE
    ├── box-img-lg.png
    ├── box-img-sm.png
    ├── bs-config.json
    ├── build
    ├── contracts
    ├── migrations
    ├── node_modules
    ├── package-lock.json
    ├── package.json
    ├── src
    ├── test
    └── truffle-config.js
```

Fig. 3. File structure of our blockchain network.

E. Security Analysis

We will now proceed to analyze both the strengths and weaknesses of the BB system described above in terms of security. While we have used our own implementation to test some of the arguments that we will discuss, we have not implement all the attacks discussed in this section.

Security Analysis

- The administrators of the election have complete power over the system and could corrupt it: as administrators of the implemented election we are able to manipulate the blockchain network in multiple ways. For example, we can create more

wallets that do not correspond to eligible voter and use them to fake votes or to allow non eligible voters to vote. This is particularly dangerous because as a voting right no one should be able to match the address that committed a transaction with a particular voter, and therefore verifying that wallets belong to an eligible voter and not to a ghost voter becomes difficult.

- This distributed and decentralized system is more resistant to attacks: Most attacks would need to successfully attack multiple nodes in the network in order to affect the system's functionality. A DDoS attack, for instance, would need to make unavailable all bootnodes in order to affect the interactions between validator nodes. While this is still a possible attack, the proposed system is more resistant than other decentralized applications. In addition, other approaches like a Sybil attack are executable in this system because the use of a private restricts access to create new nodes. In addition, corruption of a single node in the network does not allow for corruption of the ledger since honesty from the majority of the nodes avoid corruption of the blockchain.

- Traceability of votes to voters is not allowed: The use of Non Interactive Zero Knowledge Proofs prevents the system from matching wallets to the voters' identities while still verifying that the voter casting the vote is the owner of the wallet from which the transaction is made. In addition, the fact that transactions do not include the address from which they were sent nor the time at which they were made prevents adversaries from using time data to figure out who made a specific transaction, not even when the adversary knows the address of the voters' wallet.

- The devices from where voters interact with the blockchain can be hacked: by inserting a simple script that records the moves, scrolls and clicks of the mouse in a device used for voting, we were able to find the candidate for which a specific voter chose to vote. Other possible attacks include modification of the interface so as to flip the order of the candidates and make voters believe they are voting for a different candidate, or even modifying the parameters when calling the vote function to select a different candidate.

- Failure to authenticate identities: in this system all what a voter needs to be authenticated by the system and gain access to the corresponding wallet is the electronic ID and a 6-digit PIN, and therefore adversaries could threat voters to steal their credentials and vote with their identities. Including other types of identity authentication like body metrics could help alleviate this risk, although it would still be possible to hack the authentication devices to make them fail.

The code for our implementation can be found at `https://github.mit.edu/kimvc/6.857-Voting-System.git` and a video of the application as well as an attack can be found at `https://www.youtube.com/watch?time_continue=6&v=RkYbPhdXGFI`.

III. OTHER ONLINE VOTING SYSTEMS

Currently, there are several online voting systems that are being used in the United States both for small and large elections. Here we will analyze the two of those that seem to be the most popular: Helios and Voatz. We will describe their designs and compare both them to the BB System in terms of security guarantees and risks.

A. Helios

Helios is an online voting system that claims end-to-end verifiability. This reflects in the system that is used to audit ballots and to track that a ballot has been submitted or tallied. Note that the makers of Helios themselves do not believe that their software should be used for major federal elections, because they believe voter's computers are not secure enough to be used in elections that are exposed to powerful attacks.

Helios security guarantees are based on El Gamal re-encryption, which is used to enforce anonymity of the votes. Voting in this system works in the following way:

(a) An administrator creates the election and inputs the names of the candidates and the exact times at which the election must begin and end. The

administrator next creates a list of emails to which credentials must be sent. These emails contain the username and password for each person who is an eligible voter and the administrator never has access to such passwords.

(b) A voter uses her credentials and casts her vote using the online interface. At the end, the voter has the option to audit her vote, in which case the signature is checked to demonstrate that the ballot is actually being cast correctly.

(c) At the end of the election, the tally takes place and the tracker associated with each ballot informs voters that their vote has been tallied. Since the tracker is a finger print of the encryption of the vote itself, it can be used to specifically reference the voter.

Security Analysis:

- The administrator has no power to corrupt the system since Helios provides unconditional integrity. This is a big improvement over the BB system, in which despite of the fact that the system was distributed and descentralized, the administrators had power to create extra wallets, and corruption of multiple nodes could corrupt the entire system.

- A disadvantage of this system is that Helios is vulnerable to attacks like web browser corruption. While the BB system did not specify user interface details, this could be an attack that affects that system as well. In general, Helios' major problem is that the server itself has complete control over the election and therefore voters' privacy and election honesty relies on Helios.

- Similar to the BB system, Helios guarantees that voters' ballots are completely secret. While we are able to see who voted, we are not able to see what each individual person voted for, nor indeed is it a tall possible to find out. The tally happens only at the end, and is conducted by combining the encryptions of the original votes, and only then decrypting.

- Like every other internet voting system, Helios' security cannot protect the election against attacks to the voters' machines.

- Like in the BB system, the authentication method in this system allows for violations of the requirements of a voting system. Using emails for authentication means that an adversary could vote with someone else's identity by hacking the voters' email or threatening them to show them their usernames and passwords.

B. Voatz

Voatz is a mobile elections platform that runs on only latest smartphones and makes use of a descentralized blockchain network. It relies on the security of smartphone technology and immutability of a blockchain. This implies that the voters need to have the latest smartphone and the Voatz app in order to be able to vote. Voters who do not own one of these devices will then not be able to vote through Voetz. Voting in this system works in the following way:

(a) Election officials grant Voatz access to a database containing information of all the voters.

(b) Before voters cast a vote, they must authenticate their identity by scanning a valid ID and sending a live snapshot as well as their their fingerprint. The Voetz app matches the voter's fingerprint to the device and then verifies the validity of the ID and the snapshot.

(c) Each vote is then represented as a transaction that must be verified by a specific set of servers.

Security Analysis

- Vaotz prints paper ballots for each vote. These ballots are then audited to verify that each voter voted exactly once and that all the votes were counted. This limits the administrator's ability to modify the votes or add fake ones.

- Similar to Helious, the server itself has complete control over the election and could corrupt it; Voatz gains access to databases of voters and the developers themselves could possibly be

collecting all the voting data they claim that is secure and anonimous. Even though Voatz has been largely used across the U.S. many people are still reluctant to accept Vaotz as an official system for government elections, since it is hard to trust a private company when there is a lot at stake.

- In terms of traceability, Vaotz ensures it on a different way than the BB system. The identity of each voter is protected twice: first by the app in the smartphone, and then in the blockchain. It also uses end-to-end encryption. Note that this system generates accounts for the voters and has a database for them, implying that the system knows which account corresponds to which voters. However, for each vote the name of the voter is anonymous and encrypted. Thus, there is no way of telling the voter's account from tracing a vote.

- In terms of security of the device used for voting, unlike the BB system, Voatz takes measures to ensure that any device used for voting is not hacked. In order to achieve that, it only allows last generation smartphones to be used. That way it takes advantages of their various security capabilities. For example, it detects malware or changes on the Operating System. This greatly lowers the risk of the device being attacked. However, this negatively impacts the acessibility of the system, as many voters do not own one of these devices.

IV. CONCLUSION

The blockchain system analyzed in this paper does not satisfy the essential security requirements because despite the cutting-edge technology used in the implementation, there are always security concerns inherent to Internet Voting. We believe the most important limitation for implementing secure internet voting systems is that the devices used for voting could be hacked independently of how secure the voting system is.

REFERENCES

[1] Friðrik Þ. HjÃąlmarsson, Gunnlaugur K. Hreiðarsson. *Blockchain-Based E-Voting System*. School of Computer Science, Reykjavik University, Iceland, (2018).

[2] Voatz, https://voatz.com/faq.html

[3] Electronic Voting, http://lorrie.cranor.org/pubs/evoting-encyclopedia.html.

[4] Ben Adida. *Helios: Web-based Open-Audit Voting*, Harvard University, (2008).

[5] Ronald L. Rivest. *Internet Voting—Seriously?*, EVN Conference, (2016).

[6] Dennis Berman. *We the E-People*, BusinessWeek, (2000).

[7] *Arizona Democratic Party Selects Votation.com to Hold World's First Legally-Binding Public Election Over the Internet*, The Free Library, (2014).