Blockchain Based E-voting

Shane Lockwood
Department of Computer Science
University of Colorado, Boulder
Boulder, Colorado 80309
Email: shane.lockwood@colorado.edu

Sham Prasad
Department of Computer Science
University of Colorado, Boulder
Boulder, Colorado 80309
Email: shpa5747@colorado.edu

Ankur Jain
Department of Computer Science
University of Colorado, Boulder
Boulder, Colorado 80309
Email: anja1834@colorado.edu

***Abstract*—Most common electronic voting systems rely on a centralized server and database. Centralized systems are less robust than decentralized systems, for instance centralized systems are more vulnerable to malicious users, have a lower fault tolerance, and arent as trusted by the users as decentralized systems. Unfortunately, there are very few functional decentralized voting systems. The ones which are functional are missing one or more key features voting systems rely on, such as ensuring voter eligibility, ensuring third parties cant modify voters votes, and ensuring a voters privacy is maintained. We attempt to implement a distributed voting system which could handle many of these features using blockchain technology.**

I. Introduction

Democratic societies and groups rely on voting systems to function. If that voting system is insecure or difficult to use, then the group or society cant function optimally. By creating a user-friendly voting system that can be used in the comfort of ones own home which also provides several key features many voting systems should have, then hopefully the group or society could function more optimally. These features include uniqueness (member may only vote once), authentication (only eligible members may vote), integrity (votes can not be changed by third parties), anonymity (third parties can not determine who voted and who voted who what/who), fairness (results are secret until voting period has ended), correctness (votes are accurately counted and published), robustness (system is tolerant of malicious attacks/votes), verifiability (voter can verify their vote counted), and coercion freeness (third parties have minimal/no effect on voters).

We propose an implementation which allows both large and small groups the ability to vote from anywhere with an Internet connection without the need for a central server. This design is being assisted by blockchain technology which is a form of replicated data type which can be shared with many peers connected over a peer-to-peer network as first mentioned in [1]. In theory, this implementation can be utilized by a small group of individuals, by a large government coordinating an election among its people, and anywhere in between. They need only coordinate the specifics of the vote, such as who or what they are voting for and a central peer for connecting to each other, through some other means then the voting system will allow a discrete and fair vote.

II. Related Work

Follow my vote is a non-partisan public benefit corporation [2]. They use blockchain and elliptic curve cryptography to provide transparency in election results while protecting voters privacy. Yi Liu and Qi Wang wrote [3] where they discuss some of the improvements that can be made to the e-voting system. Some of them include anonymising the IP addresses of voters by using TOR or other equivalent services. Unchain.voting was one of the earliest implementations of blockchain voting [4]. It was built on top of bitcoin blockchain. Publicvotes is an Ethereum based voting systems [5]. Limitation of this was that it was not decentralised, since the results were stored in a MongoDB for better performance of the website.

III. Design

Blockchains are the data structure used for this distributed voting protocol which will be shared with all voters over a Peer-to-Peer (P2P) network. The distribution over the P2P network allows reliability, in that there is no single point of failure. The Proof-of-Work protocol, which is yet to be implemented into our design, ensures that no small group of attackers can potentially control the blockchain.

Blockchains have the potential for becoming forked resulting in multiple versions of the same blockchain but unfortunately only a single blockchain is considered for the final tally. To ensure all peers have access to the accepted final blockchain, all blockchains must be passed between peers. Currently each peer passes every blockchain possessed by that peer to all peers which ask for them. This is sufficient for sharing the blockchains since all peers ultimately receive the to-be accepted blockchains. It's highly inefficient though since many forks, many of which are globally perceived as orphaned, could potentially be formed resulting in wasted memory and network traffic. The blockchain sharing daemon will be modified to share only the blockchains accepted by the sending peer and the longest blockchain which contains content pertinent to the requesting peer. This is sufficient in ensuring each peer receives the to-be accepted blockchains assuming each peer contacts every other peer. This is feasible for relatively small groups but would need to be optimized for larger groups. Peers and voters are used synonymously throughout this paper.

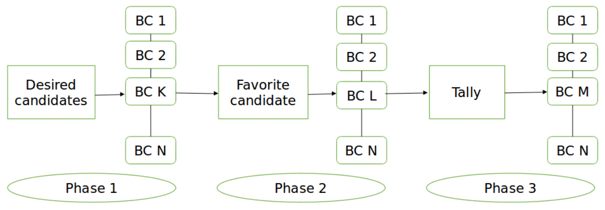

Fig. 1. A visualization of the phases during the voting process. Voters select their desired candidates and save them to a blockchain. Then given the selected candidates, voters select their favorite candidate and save them to a blockchain. Then when every voter has voted, they tally the votes and save their tally along with their name to a blockchain. The sets of blockchains for each phase are disjoint.

The actual voting process is separated into three phases. Phase one gives all the voters time to join the network and agree on a set of candidates. Phase two is where the actual vote takes place. Phase three tallies and publishes the vote. The measurement metric for blockchain acceptance is determined by the blockchain's length where longer is more likely to be accepted. Note that during these phases, the blockchains are being shared between the peers so different peers may have varying versions of the blockchains at any given time.

A. *Phase One*

Phase one sets up the vote to ensure all voters are voting on the same thing. Each voter saves their desired candidates to a block and appends that block to the longest blockchain with the same desired candidates. If no blockchain exists then they create a blockchain. Given the assumption that each group of voters with an agreed upon group of candidates will append the same blockchain, the largest group of voters must have appended the longest blockchain. This ensures the largest group of voters with the same desired candidates will have their blockchain accepted by all peers.

Multiple metrics for determining when to end phase one are relevant. We chose to end phase one when every voter appended the accepted blockchain. This was implemented by having each voter save a group of desired voters, which relates to the authentication feature as discussed in section 3.D.3, then comparing the voters who appended that blockchain with the desired voters. Other acceptable criteria for determining when to end phase one may be time based, waiting for a majority of desired voters to vote, or some hybrid of the two where the longer phase one lasts, the fewer the voters that need to be involved in phase one.

B. *Phase Two*

Phase two allows the voters to vote on their favorite candidate given the desired candidates in the accepted blockchain from phase one. Each voter saves the candidate they are voting for to a block and appends that block to the longest blockchain. Blockchains appended in phase two are different blockchains than those appended in phase one. If no blockchain exists then they create a blockchain. During this phase, there aren't multiple blockchains due to varying contents but simply due to the curse of distributed systems; multiple voters may cast their votes in unison resulting in multiple blockchain forks appended from the same blockchain. Upon sharing these blockchains, a voter who has already voted may receive a longer blockchain which they have not casted their vote to. They must then cast their vote to this longer blockchain. At the end of phase two, the longest blockchain is the accepted blockchain.

The same metrics for determining when to end phase one hold for determining when to end phase two.

C. *Phase Three*

Phase three allows the voters to tally the votes in the accepted phase two blockchain. Each voter will begin tallying the longest blockchain in their possession. If they acquire a longer blockchain during this process, they will restart with the longer blockchain. Upon completing their tally, each voter saves their tally along with their name to a block and appends that block to the longest blockchain with the same tally. They save their name to the block to ensure unauthorized voters don't cast tallies since a large enough group of unauthorized voters could cast fake tallies and ultimately alter the result. The multiple blockchains is made possible both by the curse of distributed systems and through varying contents; different tallies are appended to different blockchains. Same as during the first two phases, the longest blockchain is accepted and the contained tally of that blockchain is the official tally of the vote.

The metrics for determining when to end phase three are slightly different than those for determining when to end phases one and two. Since only a majority of votes are required for determining a selected candidate, phase three can end when the longest blockchain contains a tally from a majority of the voters. Otherwise, the same metrics apply as those from the first two phases.

D. *Feature Support*

The description of the system as explained thus far is a bare bones implementation and does not include any of the features described in section 1. Thus far we have made assumptions such as no voter will vote more than once, unauthorized voters will not vote, no one will attempt to modify other voter's vote, along with many other naive assumptions. These features aim to ensure all attacks on the integrity of the voting system are blocked.

1) Uniqueness: In many democratic voting processes, a voter may only vote once, and their vote only counts for a single vote. In phase two, we can have every voter save their name, or some identifier, to their block. A voter voting repeatedly isn't viewed as multiple votes but simply changing their vote. This is ensured in phase three during the tally process where each voter will only count the last vote provided by each voter.

2) Authentication: Many voting processes have restrictions on who can vote. To ensure only authorized voters may vote, during phase one, along with agreeing on the candidates, voters will also agree on the voters. Voters will only append their blocks to blockchains with the same desired voters as well as desired candidates. This will not affect phase two, but during phase three, voters will only count the votes of the voters in the accepted phase one blockchain.

3) Integrity: Once votes are cast, they shouldn't be modifiable by third parties. Votes are modifiable by the voter through simply recasting their vote, but to ensure third parties can't modify the blocks, some derivation of Proof-of-Work should be implemented. Proof-of-Work is essentially a puzzle which must be solved before a block can be appended to a blockchain. Proof-of-Work protocols are usually implemented such that multiple peers aid in appending a single block to a blockchain. So if more peers aid in appending the blockchain than modifying the blockchain, then the votes are protected from being modified by third parties.

4) Anonymity: Voters may wish to be discrete regarding who or what they voted for. By simply not saving their name or any personal identifier to their block, this can be ensured. Unfortunately this would interfere with other features such as authentication so we attempt to solve this issue while still saving personal identifiers to the blocks.

5) Fairness: If voters can determine the current trend of votes before the end of the voting period, it may influence their vote. To counter this, we wish for the votes to be kept secret until the voting period has expired. This can be accomplished through encryption. Each voter will encrypt their vote during phase two. During phase three, each voter will broadcast their key so each other voter can read and tally the votes. The difficulty with this is knowing which block each key belongs to. If the voters saved their name to the block then their name can be passed along with the key but this violates anonymity. If each block is assigned an unique identifier, then the identifier can be passed with the key but if someone can determine which voter transmitted the identifier then it still violates anonymity. Through transmitting a random subset of known keys and identifiers to the other voters, it will make it difficult for anyone to determine which voter casted which vote since it's unclear if the voter is passing their key and identifier or if they're simply relaying someone else's.

6) Correctness: It's a common ordeal that someone doesn't trust the tally was counted correctly and demands a recount. This assumes that a tally can be miscounted. Phase three is designed to ensure accuracy and honesty. Each voter tallies the votes and saves their tally to a blockchain with similar tallies. The longest blockchain all with the same tally is assumed to be correct. Since everyone has access to this tally, a false tally is unlikely to dominate all other tallies.

7) Robustness: Simply by being a distributed system, this system is more fault tolerant than a centralized systems with few points of failure. The major weakness in this system is invalid blockchains don't dominate the valid blockchains. This is ensured through some derivation of Proof-of-Work which ensures malicious attackers are unlikely to modify the blockchain faster than the honest voters assuming less than a majority of those with access to the blockchains are malicious.

8) Verifiability: Given most voting systems, it is unclear whether or not your vote counted. Phase three ensures all voters can determine their vote counted through the self-tally process. Each voter tallies the votes and during that process, assuming they voted, they'll tally their own vote. Unless they tally incorrectly, if their tally matches the accepted tally then their vote must have counted.

9) Coercion Freeness: Voters could feasibly be coerced into voting for something or someone out of fear or temptation. Some of the previous features aim to minimize this coercion. Anonymity aims to alleviate fear out of others knowing who or what you voted for. Fairness aims to alleviate temptation into voting for something or someone they wouldn't have voted for had they not been given that information. Unfortunately there's potential for a third party standing over a voters shoulder as they vote. Voters could setup aliases or shortcuts for voting for a certain candidate ahead of time. This would allow the voter to essentially enter an encrypted vote even in the presence of a third party where as the third party would have no idea as to who or what the voter voted for.

IV. Implementation

The design as described in section 3 sound good in theory but it's always a different story when implementing and testing. We're in progress of implementing this system but we still have a ways to go.

A P2P network has been implemented. Since peers can't access a P2P network without prior knowledge of at least one peer, or broadcasting to the whole world, a server runs which stores the addresses of all known active peers. The server is launched with no known peers. The server's address is hard-coded into each of the peers so when the peer launches, it queries the server for known peers and saves their addresses. The peer routinely queries the server for any updates on known peers. The server also routinely queries the peer to ensure the peer is still active. If the peer doesn't respond to the server, the server assumes the peer is no longer active. When the peers have addresses with other peers, each peer will routinely request all blockchains from every other known peer. Upon request, the peer would gather all their blockchains and transmit them to the peer who requested. Upon receipt, the peer will overwrite all blockchains with the same identifier which are shorter. This process runs in the background while the voting process is taking place.

The voting process unfortunately doesn't contain all the features described above yet but it's a work in progress. All 3 phases function individually don't smoothly transition between phases without user intervention. The desired voters and desired candidates are hard-coded and not input by the user. Upon launch, the user is prompted for their name. The program then automatically creates a block with the desired voters and desired candidates and either appends the largest known blockchain with similar candidates and voters or it creates a new blockchain. This process blocks in phase one until all desired voters have agreed on a single set of desired voters and desired candidates then the program stops with saved blockchains. The user then manually starts phase two. The program prompts the user for a candidate, searches for the longest known blockchain and appends a block with the candidate. The program then exits. The user must manually start phase three. This program then searches for the longest phase two blockchain and tallies the votes then saves the tally to the longest blockchain with the same tally. [1]

V. Evaluation

The P2P network is functional but inefficient. The peer queries the server and the server queries the peer which is redundant. The same result can be achieved by the peer requesting peers from the server which can count as a check-in from the peer to the server. When blockchains are transferred between peers, every blockchain is transferred which is unnecessary. Given large polls, many older forks may exist which are no longer relevant. Also, during phases one and three, if the contents of the blockchain don't reflect the voter's block contents, then those blockchains will never be appended anyway. This process can be streamlined to only transfer the sending peer's longest blockchain and the longest blockchain with content equivalent to receiving peer's block content.

The voting system is also fully functional but the most obvious downfall is the need for the user to manually start each phase. Only phase one transitions smoothly but the metric for determining if phases two and three have not been implemented yet so the program simply exits after a appending the voter's block. Fortunately, some of the features have been implemented, although in a fashion which may affect other features when implemented in the future. Authentication is partly achieved in phase one with the voters agreeing on a set of authorized voters. This has yet to be extended to phase three where the voter's ensure only authorized voters are voting. Correctness is implemented in phase three. By having the voters tally the votes, they can agree on a tally and the tally is likely the correct tally. This assumes that a majority of those tallying the votes are not malicious and possess the accepted phase two blockchain. Verifiability is implemented in phase three by having the voters tally the votes. If the accepted tally is equivalent to a voter's tally, then that voter's vote likely counted unless the voter miscounted or doesn't possess the accepted blockchain.

VI. Future Work

Goals for the P2P network involve optimizing data transfer efficiency. The server need not query each peer since each peer already queries the server. Currently every blockchain is transferred, even dated forks. A heuristic should be used to decide which blockchains need to be transferred. A simple heuristic includes only transferring the largest blockchain and the largest blockchain containing content related to the requesting peer's block content. If the requester already possesses those blockchains, then no blockchains shall be transferred. Also blockchains are passed as a single message. As blockchains grow, this will become infeasible. The blockchains should be partitioned so that they can be passed as a series of smaller chunks. Also, it may be feasible to request chunks of the same blockchain from separate peers such that it transfers more quickly. This may be a necessity given really large blockchains. To increase robustness, each peer can save a database of previously connected peers so if there is any issue with contacting the server, the peer could query those saved peers and hopefully they're active.

Goals for the voting system include implementing the metric for determining when a phase has finished. This will allow smoothly transitioning between phases without user intervention. Modify phase three to ensure only valid voters may vote and only the last vote counts. Design, implement, and test some derivation of Proof-of-Work to ensure no one can control a blockchain and no one can modify votes. Implement a encryption technique for encrypting a voter's vote then passing the key along with a block identifier to their peers upon entering phase three so the vote can be tallied. Implement an algorithm to randomly pass a subset of known keys and block ids to ensure the owner of the key and block id are kept anonymous. Implement the option for voting candidate aliases or shortcuts.

References

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.
[2] "The Online Voting Platform of the Future - Follow My Vite", *Follow My Vote*, 2018. [Online]. Available: https://followmyvote.com/. [Accessed: 06-Apr-2018].
[3] Y. Liu and Q. Wang, *An E-voting Protocol Based on Blockchain*, 2017.
[4] "unchain.voting", *Unchain.voting*, 2018. [Online]. Available http: www.unchain.voting/. [Accessed: 07-Apr-2018].
[5] "PublicVotes: Ethereum-based Voting Application - Dominik Schiener - Medium", *Medium*, 2018. [Online]. Available: https://medium.com/@DomSchiener/publicvotes-ethereum-based-voting-application-3b691488b926. [Accessed: 07-Apr-2018].

[1] https://github.com/shamprasad/teamEvoting