A Proposal for a Secure, Scalable, and Anonymous Voting System on a Private Blockchain Network

Chris Ficklin, Wid Sogata, and James Tsai

Abstract— **Voting systems continue to evolve as they have since Athenians lined up publicly to deposit stones into one of two erns. In modern times certain standards are accepted as being critical to a fair election, those being secrecy of the ballot, the ability for each eligible person to vote only once, the integrity of the votes cast, and mechanisms to verify results. Methods and technologies utilized to hold fair elections continue to evolve to better meet these requirements.**

Blockchain technology gained prominence through use in cryptocurrencies and is increasingly being developed for a broad range of applications. In this paper, we present a secure voting system design based upon a blockchain developed with the open-source blockchain creation project, Tendermint. A protocol utilizing a zero-knowledge proof is adapted from the ZeroCoin project and provides an additional step wherein the voter's ballot, or VoteCoin, is anonymously exchanged providing complete privacy.

I. INTRODUCTION

A blockchain is a recent database evolution common to cryptocurrencies like Bitcoin, but with a wide range of possible uses. All changes to a blockchain are maintained in a long list, or ledger, that extends back to its initial creation [1]. To understand the current state of the database, it's necessary to iterate through the entire list to track all changes. The blockchain is maintained by a distributed network of nodes that form an agreement on the state of the ledger and updates to it.

One thing common to all blockchain databases is the unique and very secure way that changes are made to the blockchain. Every event is securely recorded in a permanent, unalterable sequence that allows for close scrutiny at any time. At the same time, features can be implemented to obfuscate the identities of participants, providing for transaction privacy. There is significant cryptography at all points that make it practically impossible for one person to transact as another person or for unauthorized changes to be made, even by a central maintainer of the blockchain.

These features, and others, are useful in many applications from issuing stock, transferring titles, establishing the order of events such as the creation of intellectual property [2, 3], and make blockchain databases a viable technology upon which new voting systems can be implemented, while continuing to maintain compliance with the accepted standards critical to fair elections.

A blockchain database is ideally suited to election voting due to the integrity required of the process and the high stakes frequently at play. Having a trail that is both verifiable yet anonymous, while being resistant to tampering by third parties adds validity to the result. At the same time, costs can be reduced through much easier tabulation and fewer recount disputes.

A basic model of a blockchain voting system may look something like the following: A voter is sent a token called a VoteCoin when they sign in to vote. Next they announce their desire to transfer the token to the public key address associated with the candidate they wish to support. That transaction is recorded and, when the voting window closes, the candidate with the most VoteCoins is declared the winner [4].

This simplistic model provides a useful framework and some important advantages over current voting methods, but on its own fails to satisfy all of the requirements of a fair voting system. Our proposal builds on this model and introduces additional features that not only bring it in line with current expectations, but creates a system superior to anything currently in use.

We note that many aspects of a voting system are not fixed and can be implemented differently according to the needs and regulations of whatever entity sponsors a particular election. The client, or voting interface, and the hardware requirements are of particular note. A version of our system could be deployed on smart devices and computers anywhere a voter has internet access. Conversely, voting could be limited to polling places running tightly controlled hardware. Those details are left to election commissions and election oversight bodies. What we propose simply makes either option more attractive.

The remainder of the paper is organized as follows. In Section II we expand on the required features of a fair voting system. Section III provides background information with a brief introduction and contrast between the blockchain technologies Bitcoin and Tendermint. Section IV presents an overview of the specific technologies underpinning our proposed blockchain voting system including Tendermint. Next we follow up in Section V with a step-by-step walkthrough of a sample election to illustrate how our proposal satisfies the stated requirements. Section VI provides further technical specifications of our proposed implementation. Section VII addresses other considerations

such as potential limitations and topics of ongoing research, with a final conclusion in Section VIII.

II. Required Features of Voting Systems

There are four requirements generally regarded as necessary to hold fair elections.

1) Secrecy of the Ballot

There are two parts to ballot secrecy. First is the guarantee than no one else, especially the candidates or governing authorities, can link a voter to their vote. This is easy to understand. If the Sherriff knows an individual didn't vote for him, he might be tempted to retaliate while favoring his allies.

The other aspect of ballot secrecy concerns the voter's inability to prove to anyone else how they voted. This is to prevent vote buying or undue pressure, such as by a spouse or boss.

2) Eligible Voters Cast One Ballot Only

Eligibility is determined by rules, identity is checked for eligibility, and one non-transferable and non-identifiable ballot is given to a voter.

3) Integrity of the Vote

Votes are accurately recorded, implying that a ballot is easy to understand, not altered or misread, and correctly tallied. Further, the voter has confidence that the vote is counted without relying on trust. In a typical system, representatives of interested parties are involved at various stages as a check against corruption, but there are still potential vulnerabilities such as human error or proprietary electronic machines that could be hacked or purposely loaded with biased software.

4) Ability to Verify Results

Upon dispute, suspicion of impropriety, an extremely close vote, or as a random audit, there should be an ability to verify results after the fact. Paper ballots, when they work as intended, are good at this. Entirely electronic voting machines still in use are not as able to satisfy this.

III. Blockchain Technologies

Bitcoin

The Bitcoin cryptocurrency protocol first introduced the concept of a blockchain in 2008 [5]. The Bitcoin protocol uses a decentralized peer-to-peer network to propagate proposed updates to the ledger. Special voluntary participants called miners are the only nodes capable of committing the proposed changes. To maintain decentralization, miners are financially incentivized to compete against each other for the right to update the ledger. This update, or block of new information, is then announced and all participants add it to the end of the blockchain. This competition, or proof-of-work [6], limits any one party from gaining too much control over the ledger, thus maintaining its decentralized nature.

Bitcoin's proof-of-work function, called HashCash [7], requires miners find a sequence matching a specified SHA265 hash value. The difficulty is adjusted by the protocol so that this occurs approximately once every ten minutes. Combined with the current 1MB block size, the Bitcoin protocol is limited to about 7 transactions per second [8].

Furthermore, because of the large and highly decentralized nature of the peer-to-peer network, it is possible for some discrepancy to exist with respect to the most recent changes to the blockchain. These discrepancies are worked out over the next several rounds, but combined with the relatively long mining interval, confidence in the permanent state of the ledger can take as long as an hour, depending on the level of certainty required.

Tendermint

In the time since Bitcoin's inception, many other blockchain schemes have been conceived. Many applications don't require decentralization, yet still benefit from features like the historical record that is maintained and able to be scrutinized.

Tendermint is an open-source project to set up and deploy blockchain networks and databases. Networks can be private or public, but even with public distributed networks Tendermint uses methods other than proof-of-work functions to maintain the ledger [6]. It also avoids the need for each client to possess a full copy of the blockchain, which is benefit we discuss in section IV.

With private networks, which we utilize, there is little less concern that maintainer nodes behave properly since the distributed servers are maintained by the election commission. Instead, the primary concern is for fast and efficient agreement and resolution of the blockchain. This system is easy to deploy, provides incredible scalability, and the robust transaction processing necessary to handle national elections.

IV. Overview and Features of our Proposed Blockchain Network

Bitcoin and similar blockchain implementations rely on a peer-to-peer protocol to exchange transaction updates, include them in blocks, and establish the validity and order of the blocks by miners, this model has several significant limitations in a voting context.

In order for these peer-to-peer systems to function, they rely on many peers being connected to the network doing their part to pass around and agree upon information as the blockchain database is updated. While it is possible to have client-only nodes that just access the network to send updates, most need to be active participants. In order to do this, they must first download a full copy of the entire blockchain containing all past transactions, which can be quite large. This is a time

consuming step and requires many participants to remain connected to the network. During the course of an election this would not be desirable as people want to connect, vote, and disconnect quickly.

Another concern is the time it takes for transactions to be validated and the number of transactions that can be supported per second. The Bitcoin network takes around 10 minutes to an hour to verify a transaction and ensure against a double spend of a coin, depending on how much certainty you desire [9]. Compounding problems is a transaction rate limited to about 7 per second. In a large election that would not be nearly sufficient. Some gains could likely be made with tweaks to a similar protocol, but not significantly [10].

Further, the addition of blocks to the chain is handled by mining nodes competing to complete mathematical puzzles in exchange for extra currency or transaction fees. Elections do cost money, so we can imagine some scheme whereby miners are compensated for their work, but it's unnecessary and relinquishes an important role in the operation of the network to third parties. This is an unacceptable risk as elections would no doubt encourage bad actors who may invest heavily to attempt to disrupt or corrupt the results [11].

Instead we propose that the distributed blockchain database be setup using an open source project called Tendermint [12], which simplifies blockchain deployment and delegates responsibility of blockchain maintenance to authorized validator nodes that are kept online through the election. This system supports up to ten thousand transactions a second with very fast validation times. It is highly flexible, relatively easy to use, and can be spun up on a variety of platforms including virtual servers allowing large scalability.

Tendermint also supports many advanced features useful in a voting context. Their socket layer called TMSP (Tendermint Socket Protocol) allows programs embedded in transactions to run on the blockchain network with the same security and verification as simple transfer transactions. Various programming languages are supported with many exciting possibilities.

Validator nodes are points of centralization, a feature commonly frowned upon in cryptocurrency spheres, but here the trade-offs are not significant and the benefits are great. As for their security, they can be audited by interested parties, and the code embedded and secured in the transactions is able to check that it was completed successfully and report its status, all of which reduce the trust required of a centralized authority. If desired by the election commission, authorized parties representing various interests can be allowed to run nodes on the network to keep track of the state and integrity of the blockchain during and after the election. These nodes would not need to be maintainers that actually commit changes to the blockchain, but rather observer nodes that participate in the transmission of votes and the sharing of the updated blockchain to other maintainers.

V. Stepping Through the Voting Model

To illustrate the functionality of our proposed system at each stage of an election and its ability to satisfy the requirements laid out in section III, we now walk through a sample voting scenario.

1) Sign in at a Polling Location and Receive a Ballot.

A voter runs client software (similar to a cryptocurrency wallet) to connect to the blockchain and simultaneously communicate with an authentication server. She provides required identification proving she's eligible to vote. The wallet supplies a public key address to which a "VoteCoin" is sent from the authentication server. The connection to the authentication server is then disconnected. In reality, no transactions are ever 'sent' anywhere, their ownership is just updated in the ledger. Still we use this common convention of referring to transactions as transfer coins.

In this step the so-called VoteCoin is equivalent to being handed a ballot. If the voter were to send her VoteCoin to a particular candidate's public key address, it would be easily traceable to her by anyone who knew what coin was originally sent to her. While likely limited to those with access to the logs of the authentication server, it is a clear violation of the first privacy requirement defined above. Though the server could be configured to dump relevant logs, our goal is to minimize steps wherever possible where anyone, even the most powerful entities, could potentially compromise the privacy of the ballot. As such, we propose a method to exchange the original VoteCoin with a different one that cannot be traced to the physical identity of the voter.

2) The Ballot is Laundered and Replaced

This issue is a frequent concern of cryptocurrency users who want extra anonymity and protection from traffic graph analysis of the blockchain. The most common solution is to use "mixers" that take in coins from multiple users, mix then around, and the redistribute them. This is a suboptimal solution because it again relies on centralized, trusted third parties not to keep logs or get attacked. In the cryptocurrency context there are several other concerns such as fees, non-returned coins, and insufficient mixing [16]. In our model, however, control of the network is maintained privately and there exists an authority to remedy errors, so the main concern here remains the presence of centralized points of attack whether intrusion or retained logs.

A Bitcoin adaptation called ZeroCoin [13] provides an interesting feature in its protocol that we adopt to mix and anonymize VoteCoins without any trusted, centralized mixers. This is commonly called a laundry step, and despite the negative connotations of that terminology in a currency context, we use that term for simplicity. The ZeroCoin protocol employs a zero-knowledge proof to exchange and mix coins, the technical details of which are discussed later in section VI.

3) Cast the Ballot

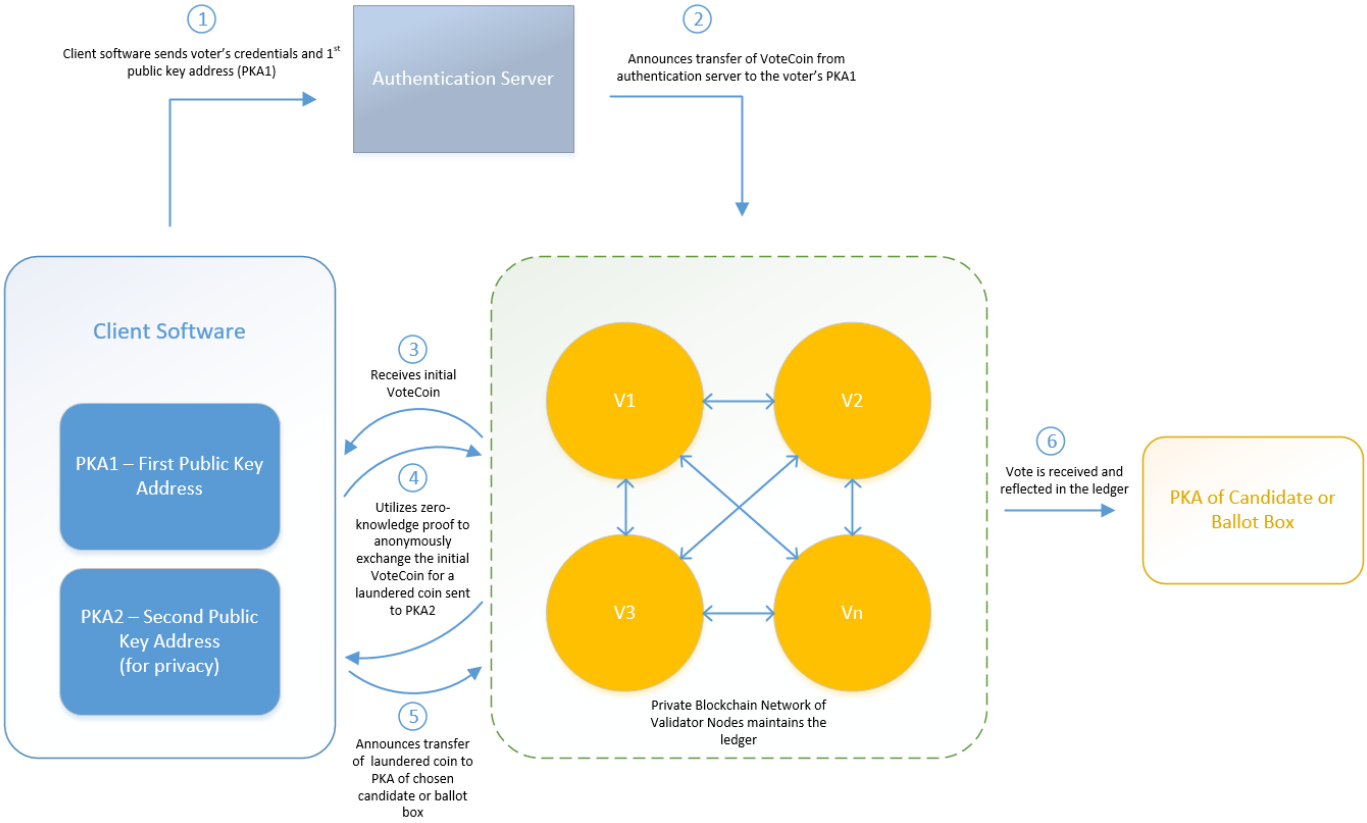

Figure 1: Blockchain Voting Process

In this scenario our voter is now in a virtual voting booth (or an actual one if required) with a ballot that cannot be traced to her physical identity. We introduce two general methods to cast a vote, one of which we expand on considerably later. In a simple scenario where only one choice is being considered, such as a referendum or single seat election, a VoteCoin can be transferred to a public key address corresponding to the voter's preference. At the end of the election the address with the most VoteCoins wins.

We refer to the example above as "simple" because there is no information or instructions embedded in the VoteCoin. Its transfer from a voter to their preference carries all the information required. However, our proposed system also allows the use of more advanced coins that can be programed to facilitate complex ballots that allow votes on multiple seats or proposals. They can also run instructions facilitating "smart contracts" with built in requirements in order for a transaction to complete. The area of smart contracts is fascinating and is presently getting a lot of attention. No doubt there are potential uses we've never considered, but we present one possible scenario now.

It's important that the user experience with the client is as simple as possible. It's likely they'd never see the hash value of the coins, the step where a new public key is created, or the exchange for a laundered coin. Yet all of those steps need to be enshrined and protected in the process to prevent tampering. Those steps could be programmed into the client, along with the entire ballot, and other instructions, but then a new client would need to be deployed for every election in every jurisdiction. Assuming the client was available for download from app stores like iTunes, it would get unwieldy to have so many different versions.

Our proposal is to develop lightweight clients for the typical ecosystems, keeping them maintained as needed for compatibility. Presumably people all over the world will use the same tried and tested (and trusted) clients if their local election commission doesn't want to write their own. And of course it will be open source so anyone can audit it.

As mentioned before, our proposal allows code to execute upon various conditions, such as receipt of a VoteCoin. The actual code to run the ballot would be stored and run from the VoteCoins. The coin will populate the client with the ballot options and any other instructions, execute the required steps to generate the two public key addresses, launder the coins, and transmit the votes. Various configuration parameters controlling the interaction with the blockchain maintainer nodes will also be embedded.

As for the security of the code, it will be verified beforehand according to local requirements, and its contents will be hashed and subsequently verified at every step. If a transaction were to fail, a warning will be reported by other nodes while attempting to add it to the chain. It will be returned to the sending client, which has had a protocol

established for a returned coin due to failed hash or other reason. Due to the short verification time, presence of a problem will be quickly known and resolved. Assuming all goes well, everyone has a chance to vote within the specified window (which can also be defined in the VoteCoin), and the integrity of the process is secure according to its design. Now we follow the vote to the last step.

4) Count the Votes

In the simple example discussed earlier, the VoteCoin is transferred to the public key corresponding to the preference of the voter. In the case of a complex ballot discussed here, all coins are sent to the same recipient which, in this analogy, is equivalent to one big ballot box. According to the instructions embedded in each coin, the preferences are recorded for each option on the ballot. This is done by a smart contract that gives the ballot box authority to mint a corresponding number of new coins and transfer one each to the public keys representing each choice on the ballot. Once that is complete and a confirmation is returned to the waiting client, the contract is complete and the process is terminated.

There are many possibilities when it comes to the specifics of the embedded code. These are just some examples. In the next section we provide more information about the supported languages and their interaction with the blockchain network via the TMSP socket protocol

VI. Technical Discussion

Blockchain network based on Tendermint and the TMSP socket protocol.

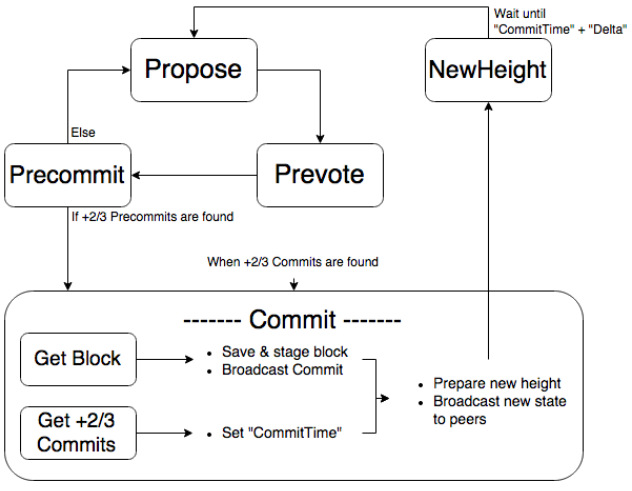

Figure 2: Tendermint Proof of Stake Consensus Algorithm [12]

Tendermint is a blockchain consensus engine that, along with a simple messaging protocol, TMSP (Tendermint Socket Protocol), enables the deployment of highly distributed Byzantine fault tolerant [14] applications that are efficient, secure and scalable.

The Tendermint core and applications run in different UNIX processes and communicate via TMSP. The socket protocol has been designed with support for many programming language, with more being adapted all the time. This provides maximum flexibility compared to competing solutions. As an open-source project, we anticipate a wealth of solutions made available to help meet the diverse needs of elections of all types and sizes. To that end, we expect the ability to accommodate existing development environments will help drive adoption and further refinement to what we hope will become a robust voting platform.

VoteCoin Laundering

ZeroCoin [15] is a cryptographic proof that a person possessed a VoteCoin and made it unspendable. The maintainer nodes can verify this proof, which gives the voter the right to redeem a new VoteCoin. The ZeroCoin is like a poker chip received at the cost of a VoteCoin and later redeemed for a different VoteCoin. ZeroCoin relies on a zero-knowledge proof, which is a way to prove a statement without revealing any other information.

We define several goals for the VoteCoin laundry phase. First, the laundered VoteCoin must be unlinkable to the original public key address, and thus potentially to the real identity of the user. Second, no party, even an entity with control of maintainer nodes, should be able to confiscate VoteCoins. Third, the process should be robust and succeed even with high transaction volume or network delays. To help ensure that, the process must be efficient and have a small impact on the network relative to the total network load.

We propose adapting the ZeroCoin cryptographic mixing protocol to our blockchain in order to exchange a voter's initial VoteCoin with a different coin sent to a second public key address not known to any party. This step breaks the association of the original coin with the physical identity of the voter.

The anonymizing process involves a separate coin minting and spending step. We'll continue to call this second coin type a ZeroCoin, after the protocol we're adapting. To create a ZeroCoin the client software generates a random serial number (*S)* which is then encrypted with a second random number (r) resulting in a ZeroCoin. Their VoteCoin and ZeroCoin are then sent to an escrow pool managed by the maintainer nodes.

To redeem a new VoteCoin to their second public key address thus completing the purpose of the laundry step, the voter (or client software running protected code on the blockchain network to automate the process) must prove two things using a zero-knowledge proof.

The first step is to prove that they know their original ZeroCoin is now in the pool, without revealing which one. This is achieved via a one-way membership function called a cryptographic accumulator [15].

Next the client proves that it knows a random number (r) and serial number (S) that correspond to a ZeroCoin in the pool. The proof is sent only the serial number (S) to the

maintainer nodes as a transaction. They then verify that the serial number (S) has not been used, after which a different VoteCoin is transferred to the voter's second public key address. After each serial number (S) verification, the accumulator needs to be recomputed. It is suggested [15] that this step represents a six-fold increase in necessary computation power needed to processes typical Bitcoin transactions, but with the fixed denomination size of ZeroCoins and VoteCoins, the additional load is expected to be managed, especially with the high transaction volume made possible by the efficient Tendermint blockchain and privately operated maintainer nodes. Furthermore, typical concerns regarding transaction fees that might be required by mining nodes is eliminated.

VII. Other Considerations

One obvious question election boards will need consider when deciding whether to allow voting outside polling places is the ability of a voter to prove their vote to someone else. Our system on its own would not prevent a person allowing someone else to view their vote as they make it or take control and vote for them. Similarly, during the registration process someone may attempt to sign in using another person's credentials, whether authorized or not. There are certain built-in safeguards against voter fraud, such as the inability to pass VoteCoins along to another person before being sent to the ballot box (not only should it be easy to prevent, it would be glaringly obvious upon analysis of the blockchain since all transactions are recorded), but we see no way around this aspect of ballot secrecy. These risks are easy to mitigate at designated polling places, but it would be for elections to weight the pros and cons and decide for themselves. Ultimately election fraud is still illegal, and widespread abuse, for instance an employer forcing everyone to prove their vote, could still be reported if the system is trustworthy. The main trade-off seems to be one of convenience, and the decision will be based on risk and logistics. Perhaps other technologies, such as biometric signatures or image capture at sign in, may offer some additional safeguards.

Another consideration for vote secrecy involves IP addresses. As we have defined it, our proposal removes the possibility to tie a vote to a person's physical identity through analysis of the blockchain or access to maintainer nodes. Some additional care will may be needed to ensure ISPs and other entities with the ability to monitor traffic are not able to connect voters to their VoteCoins. One solution might be to utilize the TOR network, but elsewhere we have made concerted efforts to retain as much control of the network as feasible. While it should not be easy to change a vote traversing TOR, it may be possible to disrupt the election in various ways. Ideally the client would be able to react and the vote not lost if there was a connection issue, but there are concerns. This problem remains ongoing for now as we continue assessing the risk and possible solutions.

VIII. Conclusion

Our proposal for a blockchain election system maintains the important features that ensure the sanctity of the democratic vote, while utilizing new technology to help further democratize the process by increasing security, privacy, accessibility, and accountability above currently available levels.

The cryptography underpinning a blockchain at every step helps guarantee the integrity of every vote and the process as a whole, and by extending that same security to the code that runs the election and processes the results, we not only help make elections easier and more accessible, we improve on even the best systems currently available, irrespective of cost and infrastructure.

The security and accountability provided by a distributed blockchain network and database add accountability and verifiability to current systems that are frequently proprietary and opaque. With a novel addition of a zero-knowledge proof to anonymize the ballot after distribution, we improve upon one of the largest threats to integrity that current blockchain voting proposals have failed to address.

References

[1] Sinclair Davidson, Primavera De Filippi, Jason Pots, Economics of Blockchain, RMIT University, Melbourne, Australia: March 9, 2016.

[2] Victoria L. Lemieux, *Trusting Records: Is Blockchain Technology the Answer?*, Paper Article, November 2015.

[3] Melanie Swan, *Blockchain: Blueprint for a New Economy*, O'Reilly Media, Inc, February 2015.

[4] Pierre Noizat, Handbook of Digital Currency, 1st Edition, Ch. 22, Blockchain Electronic Vote, Academic Press, 2015

[5] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, November 2008

[6] *Proof of Stake versus Proof of Work,* BitFury Group, 2015

[7] Adam Back, Hashcash - A Denial of Service Counter-Measure, 2002

[8] Joseph Poon, Thaddeus Dryja, The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Lightning Network

[9] Joseph Poon, Thaddeus Dryja, The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Lightning Network

[10] Ittay Eyal, Emin Gun Sirer, *Majority is not Enough: Bitcoin Mining is Vulnerable*, Department of Computer Science, Cornell University

[11] Yonatan Sompolinsky, Aviv Zohar, *Accelerating Bitcoins Transaction Processing: Fast Money Grows on Trees, Not Chains*, International Association for Cryptologic Research 2013

[12] Jae Kwon, Tendermint: Consensus without Mining, Tendermint

[13] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, Edward W. Felten, *Mixcoin: Anonymity for Bitcoin with accountable mixes (Full version)*, International Association for Cryptologic Research, 2014

[14] Miguel Castro, Barbara Liskov, *Practical Byzantine Fault Tolerance*, Massachusetts Institute of Technology, Cambridge, Massachusetts

[15] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin, *Zerocoin: Anonymous Distributed E-Cash from Bitcoin*, Johns Hopkins University Department of Computer Science, Baltimore, USA

[16] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in SP '14, 2014.