Blockchains and voting: somewhere between hype and a panacea

Yomna Nasser*, Chidinma Okoye†, Jeremy Clark†, Peter Y A Ryan‡
** EFF/University of Waterloo, † Concordia University, ‡ University of Luxembourg*

Byline. Public intellectuals are forecasting a "blockchain revolution" [TT16]. Will verifiable voting be part of the revolution or will it pass it by? The case study of voting can teach us a lot about blockchain technology—both its potential and its limitations.

Introduction

The blockchain—a novel data structure and consensus mechanism designed for the decentralized currency Bitcoin—is currently being considered, exuberantly, for new applications in the financial and technology sectors. A steady stream of whitepapers from well-established banks (*JPMorgan Chase, Deutsche, Barclays, Citi*), professional service networks (*PwC, Deloitte, EY, KPMG*), and technology companies (*IBM, Microsoft*) explore how blockchains and distributed ledgers can create new digital assets, smart contracts, and provenance systems that enable direct interactions between participants, eschewing traditional intermediaries.

With all this attention (and some would say "hype") on new blockchain applications, it is unsurprising that a number of researchers, developers, journalists, and start-ups have posited blockchain technology as the missing link for transparent, verifiable election systems. Are they really? In this article, we approach this question in a systematic way from both directions: (i) beginning with a blockchain, what challenges arise from layering a voting system on top, and (ii) examining existing verifiable voting systems, how might a blockchain augment their properties? Our conclusion will not please those looking for "one-armed" researchers:[1] blockchains are a useful augmentation to verifiable voting in some circumstances and may introduce interesting ways of voting in non-traditional settings; but on the other hand, blockchains are not a panacea. For current public sector elections in particular, any holistic approach to security will at most use a blockchain as a supporting component in a much larger system, if at all.

In this article, *we do not propose a new system*. Rather we systemize existing knowledge on blockchain technology and verifiable voting technology, and showcase what the landscape looks like for their composition. It is not a survey, per se, because we are commenting on future work—informed by past work but not summarizing it. Our goal is to provide a justified research agenda by bringing into focus where there is potential, and pruning out what seem to be dead-ends.

[1] US president Harry Truman once quipped his desire for a "one-armed economist" that would not say "on the other hand."

An abbreviated history of the blockchain

At the 1990 *CRYPTO* conference, Haber and Stornetta introduced [HS90] and subsequently refined (alongside others [BdM91]) a data-structure for appending a series of temporally-ordered messages to each other. The later designs from this literature propose that a set of messages are aggregated into a single value using a cryptographic primitive called a Merkle or hash tree, such that one can efficiently prove a given message is fingerprinted by this value (called a Merkle root). If any message in the set is modified, the root is also necessarily modified, and so the root serves as a binding commitment to the exact set of messages. If the root is widely witnessed (companies like Surety have published such roots in the *Financial Times* and the *New York Times*), the messages are effectively locked in. If a new set of messages is produced, its root can be hashed together with the previous roots to produce a value, called the header, that locks in the entire sequence of events.

Two years later at *CRYPTO*, Dwork and Naor proposed a system for fighting spam where the sender of an email would be required to compute the solution to a moderately hard puzzle (e.g., something that would take a normal computer several seconds or minutes) before sending the email [DN93]. The recipient would be able to verify if the solution is correct quickly (e.g., in micro or milliseconds) and would ignore messages without a correct solution. Hashcash [Bac02] was an instantiation of this idea where the puzzle involved computing the hash of the email (and meta-information) over-and-over again, each time changing a meaningless counter included in the hash input, until the output of the hash function (considered a random mapping) happens to have a specified number of leading zeros.

In late 2008, Satoshi Nakamoto (the pseudonym of a not-yet-identified developer) posted a whitepaper to a cryptography mailing list outlining a new digital cash system called Bitcoin [Nak08]. By this time, many attempts at commercializing some form of cryptographic digital cash had been made, none reaching commercial success. The design of Bitcoin is quite different in several regards from previous work, but one particular design decision that distinguishes it from nearly all previous proposals is having a publicly available ledger that records every transaction. The ledger takes the form of the Haber-Stornetta data-structure: a set of transactions (called a block) are accumulated into a root, and the root is hashed with the previous block header to create a new block header (the chain of block headers being called the blockchain). Nakamoto's subtle but incredibly insightful twist is that each time a new block header is produced, it must integrate the solution to a moderately hard puzzle.

[Authors' note: if useful, we can include a sidebar with a technical specification of the hash-based computation puzzle, the Haber-Stornetta data-structure, and their combination in the Nakamoto blockchain]

A novel consensus mechanism

It is difficult to succinctly motivate and describe everything the addition of the puzzle enables. We refer the interested reader to a more thorough survey [BMC+15], but in short, Nakamoto's blockchain allows an open network of computers to update the blockchain in a way that (1) no node is in charge, (2) all

nodes agree on the contents of the ledger, and (3) nodes are incentivized to do the work of updating the ledger.

A blockchain of length L includes the solutions to L puzzles, and these solutions will add up to a considerable amount of computational work. The puzzle difficulty adjusts dynamically to keep the average solution time at 10 minutes. If nodes see more than one valid version of the blockchain, they will first prioritize the chain with the most work (roughly speaking, this is the longest chain), and then if chains with equal work co-exist, whichever they saw first. Once a node has prioritized a chain, it will assemble a block of valid transactions (e.g., ones that do not double-spend) and attempt to add it to the chain by solving the puzzle. All nodes are constantly doing this, and the first node to find a solution will broadcast it. This new chain should now be prioritized by all nodes (since it is one block longer than any other prioritized chains). The node that solves the block is allowed to keep seigniorage and certain fees affiliated with the transactions they chose to put into the block. To ensure they capitalize on this, they must ensure their chain (both their block and each previous block) contains only valid transactions or the network will reject it. This incentivizes every node to validate every block of the chain they prioritize.

If a malicious node were to change or drop a past transaction in a past block, the root of this block would also change and, subsequently, every blockheader that follows it. Thus to get this modified chain prioritized, the attacker would have to "catch up" to the length of the current blockchain and could only do so if it had considerable computational power relative to rest of the network. There are important nuances to this process of forming a consensus, but generally it is the case that all nodes will agree on the contents of the blockchain from beginning to nearly the end, with some disagreement over the last few "tail-end" blocks [GKL15]. For this reason, it is advised that one waits until a transaction is six blocks from the last block before considering it effectively immutable [Nak08]. It is also generally the case that any valid transaction will eventually be included in the blockchain [GKL15].

To build a currency on top of a blockchain, users chose a set of unrelated signing keys, for a digital signature scheme (currently ECDSA), and use the public keys as "addresses" for receiving payments. No other identifying information is used. New currency enters the system via the nodes that solve the blocks (they specify an address they own in their solution) and currency can be sent in any amount to any other public key by digitally signing a transaction specifying the details (requiring the signature counters theft). The transaction is broadcast to the network and eventually added to the blockchain, which has a publicly verifiable history of every transaction (ensuring no double spending) and is generally immutable (preventing any future revocation of the transaction). This is a simplification of Bitcoin but a sufficient description for our purposes.

Distributed ledgers

It is useful to think of the blockchain as both a data-structure and a consensus mechanism that enables agreement on a ledger of messages amongst a set of mutually distrustful nodes, premised on the assumption that the adversary does not own a considerable fraction of the computational power on the network (e.g., half is sufficient to eventually rewrite the ledger and a quarter is sufficient to attack subtler properties [ES14]). The way consensus is formed, through a network of devices solving a computational

puzzle as fast as they can, is considered by many to be wasteful and not desirable. Many of the companies exploring new blockchain applications are also considering how trusted entities can be added into the system to avoid computational puzzles, shorten block update times from 10 minutes (in Bitcoin) to seconds, and govern network join/write/read privileges. We use the term distributed ledger to refer to the family of technologies that includes both Bitcoin's blockchain and these relaxed variants. Ironically, most distributed ledger technology strips away all of Nakamoto's novel insights, and could have been deployed using 1990s' state-of-the-art knowledge.

Verifiable voting and digital cash are old siblings

Both digital currency systems and electronic voting systems have a long intertwined history in the cryptography literature. In fact, both originate with the same person: David Chaum. Chaum proposed the first cryptographic voting system in 1981, based on his mix network protocol [Cha81]. The next year, he proposed the first cryptographic payment system based on his blind signature primitive [Cha82]. Blind signatures were later studied extensively for voting (the hallmark paper being [FOO92]), and cryptographic mixing has been used in digital cash, recently to anonymize Bitcoin transactions [BNM+14, RMK14]. The shared history does not end there, with many cryptographic primitives and protocols being utilized in both payments and voting: zero knowledge proofs, ring signatures, digital credentials, homomorphic encryption, algebraic MACs, and on and on. We can now add blockchains to the list.

A cryptographic or end-to-end verifiable (E2E) voting system is one that provides a proof to each voter that her ballot was included, unmodified, in the final tally (even in the face of a corrupt election authority and malicious vote capture equipment). This proof generally consists of an obfuscated form of the voter's ballot choices, called a receipt, as well as public data about all recorded receipts and the tallying process that can be validated by anyone who wants (even those who did not vote). It is a strict condition of cryptographic voting systems that this proof leaks no information about how the voter voted, beyond what can be inferred from the tally alone, including the case where the voter deliberately crafts her ballot to leak how she voted. For in-person voting, we assume the voter's actions cannot be observed in the voting booth. For online or remote voting, no such assumption can be made and it becomes quite challenging (but theoretically possible [JCJ05]) to provide voter privacy when anyone can look over her shoulder and coerce/pay her to make certain choices.

A toy blockchain voting system

We now describe a simple voting system, layered on top of Bitcoin, to stack it up against the properties of a true E2E voting system. This is *not* a proposal for a system. Rather it is a pedagogic device we will tweak to showcase both the strengths and weaknesses of blockchains. It is not purely pedagogical however, as it is the core protocol of a number of blockchain voting projects including *Swarm, Ballotchain, BitCongress, Blockchain Voting Machine*.

Voters register a Bitcoin address with the election authority (EA). The EA publishes a list of addresses but does not list which address belongs to which voter. Each candidate also publishes an address. Voters

then cast a ballot by sending a small payment to their selected candidate. Any deviation from the voting rules (e.g., one vote per voter) can be seen by inspecting the blockchain, and the tally is visible by inspecting the candidate's received payments.

This scheme has one major benefit and several drawbacks. The main benefit is that when voters cast their ballots, they are doing so to a decentralized global network of computers where at least some will have no affiliation to the election authority and will include the transaction in their blocks.

The first drawback of the scheme is that the registration authority knows the mapping between voter identities and keys, and the mapping between keys and candidates voted for is public, so the registration authority can break voter privacy. Known cryptographic techniques can address this in the following way. The voter list consists of real voter identities and each voter adds a public key encryption of her address to the list beside her name. A set of mutually distrustful parties takes each encrypted address, with the real identity left behind, and shuffles it into a new location in the list and then "obfuscates" the value so that it cannot be recognized but still decrypts to the same value. Once the list is shuffled by each party, the complete set is decrypted with a decryption key shared amongst them. This system is essentially what Chaum originally proposed in 1981! In his scheme, voters chose random signing keys that are shuffled and revealed in the first phase, and used to sign ballots in the second phase [Cha81]. A modern variation could use ECDSA signatures and provably correct shuffling [HS11].

A fairly equivalent approach could have voters approach the registration authority in order to obtain a signature on their address. By using a "blind signature" scheme, the voter can obfuscate ("blind") the address such that the authority just sees random-looking data when it signs, and then de-obfuscate ("unblind") the message and signature in tandem such that the signature is properly formed relative to the original address. This is the basis of many voting systems [FOO92], and is largely equivalent to the voter list approach above with the drawback that once a signature is issued, it becomes invisible — it cannot be revoked (if a voter becomes ineligible) and no one can enumerate how many signatures an EA has handed out.

Tokens and coins

Because bitcoin payments can be traced on the blockchain, it is possible to flag a small amount of bitcoin to serve as a digital token for some other kind of asset. These tokens can be transacted between parties in the same way as Bitcoin itself (and possibly in new ways). Colored coins, Counterparty and MasterCoin rely on this idea. While a ballot can be tokenized, the difference is largely semantic. In the toy scheme, anyone can send bitcoins to the candidate addresses and the system will have to use a list of authorized voters to filter out spam transactions from the real ones. Tokens do not solve this issue—the process for checking a token's validity is also consulting a list of coins that have been tokenized. In a closed distributed ledger, submitting a transaction could require authorization. This would eliminate spam but would reintroduce trust in the network for censorship-resistance and consensus.

A running tally

Even with one of the address anonymization add-ons, our toy blockchain voting scheme has the property that votes can be seen as they come in, enabling a visible running tally for each candidate. From a transparency perspective, this can be either a feature or a bug. Some blockchain voting projects (*Follow My Vote*) advocate this as a beneficial feature. They argue that when combined with the ability to change one's vote (this could be accomplished in the system above by giving the voter many tokens with the policy that the 'last one' counts), a new type of scoring protocol emerges where voters can cast a ballot for their most favored candidate and then manually intervene if it seems they are too far from winning midway through the voting period. In response, we would argue that ranked choice voting is a more direct way of achieving this, and one that removes both the voter's guesswork and manual intervention. Further, open tallies are a "bug" in one major sense: they are illegal in essentially all governmental elections today. A second issue concerns ballot secrecy: we know from paper audit trails (added to electronic voting machines) that writing votes into a ledger in the same order as they are cast can reveal how voters voted through simple timing analysis.

Hiding voter choices

These two drawbacks, legality and ballot secrecy, motivates us to consider how the tally might be kept confidential until voting closes. Once again, the E2E literature offers numerous solutions to this problem and we highlight two general approaches in applying the literature to this issue.

The first approach to hiding the vote itself is to have voters submit their votes as a message instead of moving tokens to a candidate's address. Bitcoin offers a transaction called `OP_RETURN` that allows a user to burn a small amount of Bitcoin to embed 80 bytes of data into the blockchain. The data could be a public key encryption of the vote, to be tallied after the election in a privacy preserving manner (e.g., using additively homomorphic encryption [CGS97] or with a verifiable mix-network [SK95]). Alternatively, the set of voters might engage in a pre-election protocol that produces for each voter a secret random number they can add to their vote to mask it, such that the sum of all mask values is zero [KY02]. When the last vote is submitted, the blinding factors all cancel out leaving just the sum of the votes, eliminating the need for an EA. In both cases, this is the core idea of a much larger protocol that needs additional steps and proofs to achieve all the desired properties of an E2E system. A system of the second type has been recently proposed and implemented [MSH17].

The key question this raises is if such a system is really a "blockchain voting" system any more given it is using the blockchain in a very limited fashion: for posting publicly viewable messages into a ledger. Toward an answer, consider a second approach that can leverage voting through financial transactions, as in our blockchain voting example, while preserving the privacy of the vote until after the election is complete. The main obstacle to accomplishing this is the known, one-to-one mapping between candidates and addresses. Instead, each ballot could have a unique set of one-time use addresses for each candidate. This is closely related to systems like Scantegrity [CCC+08] where each candidate on each ballot has a unique code associated with them: in our case, the code happens to be a Bitcoin address. A largely equivalent approach would be to distributed ballots that have the order of candidates shuffled, and have a

set of addresses corresponding to the first candidate on the list, the second, etc. Such a system would correspond to Pret a Voter or vVote [RBH+09]. The backend of any of these systems would be able to verifiably map votes back to the correct candidates while preserving ballot secrecy.

Our key observation can now be made. There does not seem to be any difference between a voter sending a token to an address affiliated with their selection with a standard Bitcoin transaction, and simply writing the selection in an `OP_RETURN` message. The latter can do everything the former can (and more). Thus the ability to conduct transactions does not seem necessary for securing a voting system. In attempting to build a blockchain voting system, we have to layer so much additional cryptography on top that it effectively becomes a normal end-to-end voting system from the literature with the one modification that public messages are recorded on a blockchain. This begets an important follow-up question: where do E2E voting systems propose storing their public messages and is it any better than a blockchain?

The mythical bulletin board

Virtually every E2E voting system assumes the existence of a "bulletin board," which was first formalized by Benaloh and Tuinstra [BT94] and is considered to be an append-only, broadcast channel. The vast majority of papers adopt this as an assumption without specifying exactly how a bulletin board should be constructed. When E2E elections have been run in the real world (e.g., Scantegrity, vVote, Helios, etc), the bulletin board is typically implemented as a website with data signed by the election authority, perhaps with some replication by other interested parties.

Such an implementation is neither strictly append-only nor a broadcast: the data can be modified or reordered at any time the EA choses, the EA can equivocate on what is stored on the bulletin board by sending different records to different voters, and the EA can play gatekeeper and drop messages it does not want published. Of course, the EA might get caught modifying the bulletin board, and could be held accountable because it signs every version, however this is a detection mechanism more than a prevention mechanism.

By contrast, embedding messages into a popular blockchain offers several improvements. First, there is no single entity to serve as gatekeeper and no successful cases of censorship have been observed in Bitcoin's history. Once a record is incorporated several blocks from the tail-end of the blockchain, it cannot be feasibly modified, and everyone holding the longest chain will see exactly the same records in this part of the chain (this is called a common prefix in the blockchain literature [GKL15]). For these reasons, it is our opinion that a blockchain is the best bulletin board implementation we are aware of.

Carbon dating messages

Blockchains have one more trick that a traditional bulletin board cannot do. Because updating the blockchain requires computational work, a message that is X blocks from the tail-end of the blockchain was embedded X puzzles ago in time. Computing the solution to X puzzles takes a measurable amount of time. We might not be able to be very precise, since the computational power of the network varies, but we cannot shortcut months of the entire network's computational work in a few days. Thus a message

embedded in a blockchain accumulates work over time, much like a fossil accumulates carbon, and this work cannot be faked even if the entire network is malicious.

This unusual property was observed by Clark and Essex [CE12], two contributors to the Scantegrity system. In Scantegrity, certain commitments are made prior to the election and if a corrupt election authority changed and backdated them after the election, without being noticed, the soundness of the tally is no longer guaranteed. Naturally voters may only become interested in verification after some unexpected election result, and such a voter has no way to know from a bulletin board whether the pre-election commitments were really made before the election or not. Thus, the researchers embedded the pre-election commitments for the 2011 municipal election in Takoma Park, MD (which was using Scantegrity) into Bitcoin's blockchain so that voters after the election could be reasonably sure that the pre-election commitments must have really been made before the election, since by the time of the election, they had acquired a month of computation work.

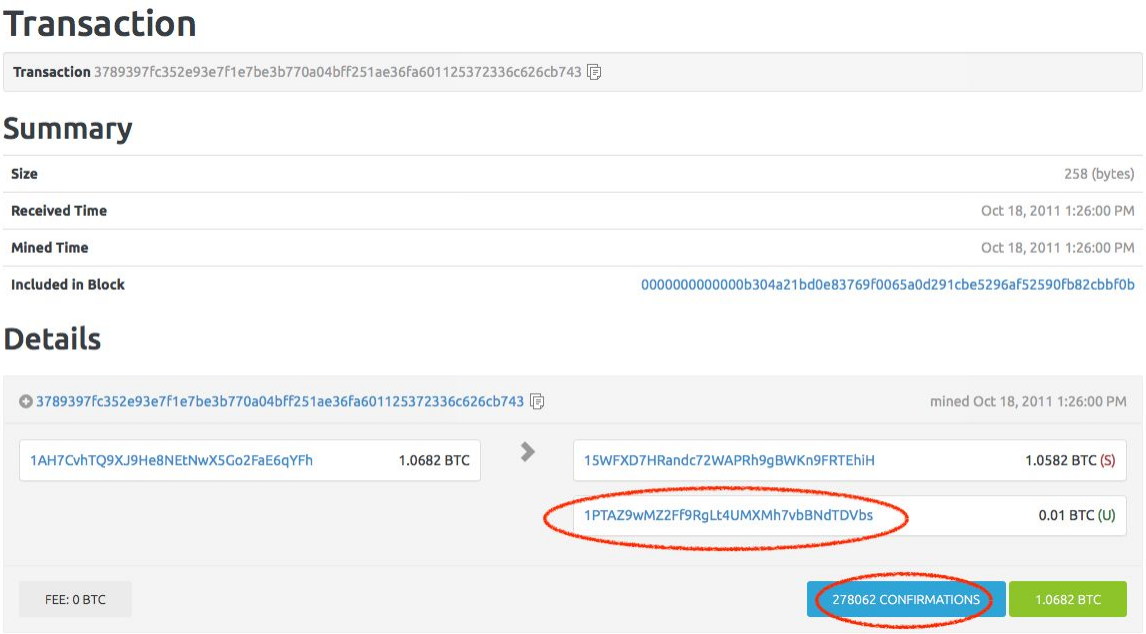

Figure 1: A screenshot from Blockexplorer showing the Scantegrity pre-election commitments (1PTAZ…) embedded in Bitcoin's blockchain prior to the 2011 municipal election in Takoma Park, MD. At the time of writing, this message has been extended by 278062 blocks (each block requiring an average of 10 minutes of computation work from the Bitcoin network). The commitment value itself was embedded as a Bitcoin address since `OP_RETURN` *was not supported at the time. The 0.01 BTC at this address is thus unspendable.*

What would deployment look like?

If a blockchain were deployed as a bulletin board in an election, how would that look? For in-person voting systems, voters would use standard voting technology. E2E systems have been built on both optical scan and direct-recording electronic (DRE) architectures. In this case, the EA equipment (whether the devices the voter operates or some central tabulator) would post the events to the blockchain. The voter would retain some sort of privacy-preserving receipt (e.g., a confirmation code, ciphertext, marked ballot position, etc.) that they would use to cross-check with the data posted on the blockchain. Thus, their

first and only interaction with the blockchain would come after the election for the purposes of an audit. This would require a computer program or website they trust. While the blockchain itself will not equivocate on what is stored, the voter's computational device can certainly display the wrong information (the voter cannot compute hashes in her head to detect something is amiss). The benefit of using Bitcoin's blockchain is the number of third party software and web-based tools for reading data from the blockchain.

In an online voting setting, voters cast their votes from their device which requires write access to the blockchain in addition to read access. The voter's device might not be able to obtain the correct software (or client-side scripts) due to a network attack and well-crafted malware can always interfere with a voter's ability to vote even if the network connections are secured with HTTPS. A malicious device can change a voter's selections, while simulating the process of casting the correct choices, and then simulate the receipt check should the voter perform one after the election from the same device.

The voting literature has a number of schemes, starting with Chaum's SureVote, where voters are mailed random-looking codes corresponding to each candidate to type into their devices so the device has no way of knowing how it might modify the vote. These solutions are largely tangential to the use of a blockchain but we make one remark. In an advanced code voting system like Remotegrity [ZCC+12], the voter and EA exchange codes until the voter is satisfied and then sends a final lock-in code. A blockchain can provide assurances that this lock-in will be eventually incorporated if it is successfully broadcast.

Can a human use that?

End-to-end voting systems have evolved over time to systems where as much cryptographic detail is swept under the carpet as possible, especially during voter interactions. In any voting system that requires the voters themselves to authorize a blockchain transaction, the voter will need to safely and reliably store a cryptographic key pair. Key management is known to be difficult for voters—whether for secure email or for Bitcoin itself [EBSC15]. It is generally not sufficient to use shorter passwords (cryptographically expanding them into keys) as the public keys will be placed on a public blockchain where they can be subject to offline brute-force attacks—in fact, many Bitcoin thefts have occurred because of password-derived keys. Any system that authenticates voters using cryptographic keys needs to carefully consider usability factors.

DRE audit logs

In E2E voting systems, it is sufficient to only write critical data like voter receipts to the bulletin board. However if an incident were to trigger an E2E system to not validate the final result, election officials would appreciate a finer-grained log of events to uncover issues. In fact, non-verifiable voting technology like DRE (direct-recording electronic) machines record a detailed electronic log of every operation they do across an election. DRE logs are critical in post-election audits. An issue is that the potentially faulty or malicious DRE generates and maintains its own log itself. In 2007 and in subsequent work, Sandler and Wallach propose that DREs broadcast logged events to a private network of nodes (called VoteBoxes) for

recordings in a Haber-Stornetta type of data-structure [SW07]. It would be quite natural to implement their Auditorium system with a private blockchain run between VoteBoxes.

Voting through smart contracts

Blockchain projects like Ethereum enable verbose smart contracts to be expressed and enforced on a blockchain. This opens eligibility to anyone matching a specified algorithmic description. It is hard to imagine all the possible forms this could take but it is one key area for future research to explore. So far in this article, we have concentrated on the use-case common to all governmental elections but other elections are imaginable.

Assume ownership shares in a company are issued on a blockchain (as was recently approved by the United States Security Exchange Commission for Overstock) and a shareholder vote occurs, shareholders could cast ballots that are weighted by how many shares they own (their *stake*) and they could even do so without revealing which shares belong to which shareholder. The outcome of a shareholder election might trigger certain outcomes on the same blockchain: financial transactions, stocks to split, new shares, or a dividend to be paid out. NASDAQ is experimenting with this type of system in Estonia.

A voting system could also be designed to give one vote per unit of computational work performed. Some Bitcoin users generate for themselves "vanity addresses" where their otherwise random Bitcoin address contains a certain prefix of characters. They do this by generating new addresses as fast as they can until one happens to have the right prefix. An election could be held where votes are only counted from addresses containing a certain prefix (the longer the prefix, the exponentially more work it requires to generate). This could also be layered onto a 'one vote per voter' scheme as a spam deterrent. The idea of eligibility based on computation effort is actually used indirectly in Bitcoin today. The Bitcoin community sometimes holds polls on protocol changes and the network nodes that solve the blocks include their vote in their solved blocks, which skews the result toward the side doing the most work solving blocks.

Some countries, like Canada, have a per-vote subsidy where a set amount of money is paid to political parties for each vote they receive — this could be automatically redeemable after an election. A country with mandatory voting might collect a deposit fee that is later refunded upon receiving a vote. More generally, voters might post some money as a surety that is returned only if they behave correctly within the protocol [MSH17]. This blend of voting and payments could also be used for nefarious purposes, like automating vote buying (c.f., [JKS16]). Bets on an election outcome could be automatically settled. While the outcome of an election can always be reported to a blockchain by a trusted party (called an "oracle" in the Bitcoin community), having an on-blockchain election can enable direct observation of the outcome by other scripts running on the blockchain.

Conclusion

Satoshi Nakamoto's blockchain is undeniably a breakthrough for decentralized digital cash, and it may prove useful in disintermediating other protocols and procedures. However when it comes to voting, it

seems the most useful contribution a blockchain can provide is a broadcast channel for cryptographic voting systems. In this role, blockchains are not only sufficient but actually provide some advanced properties, like carbon dating, that cannot be directly achieved with a standard bulletin board protocol. Blockchain-based voting systems also have the potential to entangle the eligibility for an election and the outcome of the election with other blockchain payments, assets and smart contracts in new and creative ways. While this is a somewhat abstract thought to end on, we are confident that researchers will recognize useful scenarios for this incredible flexibility.

Acknowledgments

We are thankful to members of the Bitcoin community for taking time to comment on our work and make suggestions, including Alice Cecile, Rolf Haenni, and Patrick McCorry.

Bibliography

[Bac02] A Back. Hashcash: a denial of service counter-measure. Tech Report, 2002.

[BdM91] J Benaloh and M de Mare. Efficient broadcast time-stamping. Tech Report, Clarkson University, 1991.

[BNM+14] J Bonneau, A Narayanan, A Miller, J Clark, JA Kroll, and EW Felten. Mixcoin: Anonymity for Bitcoin with accountable mixes. *Financial Cryptography* 2014.

[BMC+15] J Bonneau, A Miller, J Clark, A Narayanan, JA Kroll, and EW Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. *IEEE Symposium on Security and Privacy* 2015.

[BT94] J Benaloh and D Tuinstra. Receipt-free secret-ballot elections. *ACM STOC* 1994.

[Cha81] D Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[Cha82] D Chaum. Blind signatures for untraceable payments. *CRYPTO* 1982.

[CCC+08] D Chaum, R Carback, J Clark, A Essex, S Popoveniuc, RL Rivest, PYA. Ryan, E Shen, and AT Sherman. Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. *USENIX EVT* 2008.

[CE12] J Clark and A Essex. CommitCoin: carbon dating commitments with Bitcoin. *Financial Cryptography* 2012.

[CGS97] R Cramer, R Gennaro, and B Schoenmakers. A secure and optimally efficient multi-authority election scheme. *EUROCRYPT* 1997.

[DN93] C Dwork and M Naor. Pricing via Processing or Combatting Junk Mail. *CRYPTO* 1992.

[EBSC15] S Eskandari, D Barrera, E Stobert, and J Clark. A First Look at the Usability of Bitcoin Key Management. *USEC* 2015.

[ES14] I Eyal and EG Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Financial Cryptography* 2014.

[FOO92] A Fujioka, T Okamoto, and K Ohta. A practical secret voting scheme for large scale elections. *ASIACRYPT* 1992.

[GKL15] J Garay, A Kiayias, and N Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. *EUROCRYPT* 2015.

[HS90] S Haber and WS Stornetta. How to Time-Stamp a Digital Document. *CRYPTO* 1990.

[HS11] R Haenni and O Spycher. Secure internet voting on limited devices with anonymized DSA public keys. *USENIX EVT/WOTE* 2011.

[JCJ05] A Juels, D Catalano, and M Jacobsson. Coercion-resistant electronic elections. *ACM WPES*, 2005.

[JKS16] A Juels, AE Kosba, and E Shi. The Ring of Gyges: Investigating the Future of Criminal Smart Contracts. *ACM CCS* 2016.

[KY02] A Kiayias and M Yung. Self-tallying elections and perfect ballot secrecy. *PKC* 2002.

[MSH17] P McCorry, SF Shahandashti, and F Hao. A Smart Contract for Boardroom Voting with Maximum Voter Privacy. *Financial Cryptography 2017*.

[Nak08] S Nakamoto. Bitcoin: A peer-to-peer electionic cash system. Tech Report, 2008.

[RBH+09] PYA. Ryan, D Bismark, J Heather, S Schneider, and Z Xia. Pret a voter: a voter-verifiable voting system. *IEEE TIFS*, 4(4), 2009.

[RMK14] T Ruffing, P Moreno-Sanchez, A Kate. CoinShuffle: Practical decentralized coin mixing for Bitcoin. *ESORICS* 2014.

[SK95] K Sako and J Kilian. Receipt-free mix-type voting scheme—a practical solution to the implementation of a voting booth. *EUROCRYPT* 1995.

[SW07] D Sandler and DS Wallach. Casting votes in the auditorium. *USENIX EVT* 2007.

[TT16] D Tapscott and A Tapscott. Blockchain Revolution: How the technology behind Bitcoin is changing money, business, and the world. *Portfolio*. 2016.

[ZCC+12] F Zagórski, RT Carback, D Chaum, J Clark, A Essex, and PL Vora. Remotegrity: design and use of an end-to-end verifiable remote voting system. *ACNS* 2013.