An E-voting Protocol Based on Blockchain

Yi Liu and Qi Wang

Department of Computer Science and Engineering
Southern University of Science and Technology, Shenzhen, China
liuy7@mail.sustc.edu.cn,wangqi@sustc.edu.cn

Abstract. Because of the properties such as transparency, decentralization, irreversibility, nonrepudiation, etc., blockchain is not only a fundamental technology of great interest in its own right, but also has large potential when integrated into many other areas. In this paper, based on the blockchain technology, we propose a decentralized e-voting protocol, without the existence of a trusted third party. Furthermore, we provide several possible extensions and improvements that meet the requirements in some specific voting scenarios.

Keywords: E-voting, Blockchain, Blind signature

1 Introduction

Electronic voting (e-voting), which uses electronic systems to aid casting and counting votes in an election, has been a research topic of interest for the past few decades in cryptography (for more on e-voting, see [1, 2]).

In comparison with the traditional paper-based voting, remote e-voting is environmentally friendly, real-time counting and processing, less error-prone. Meanwhile, as the time and efforts to vote decrease, the overall voter turnout may increase [3]. Nevertheless, e-voting has been used in only a few countries, for example, Estonia [4], Canada [5], Australia [6]. E-voting systems have been analyzed for security reasons and some flaws have been found [7–9]. In some countries, e.g., Germany, online voting has been abandoned due to its insufficient security and vulnerability.

E-voting can be views as special cases of secure multi-party computation (MPC) [10]. Because of the properties such as transparency, decentralization, irreversibility nonrepudiation, etc., blockchain has a large potential when integrated into many areas. It was remarked that it is very likely to solve MPC problems based on blockchain, of which the requirements could not be fully met by traditional solutions [11], such as e-voting [12].

In this paper, we investigate this idea, and propose an e-voting protocol based on the blockchain technology. The rest of the present paper is organized as follows. In Section 2, we present some related work on e-voting. We summarize the main contributions and properties of our protocol in Section 3. In Sections 4 and 5, we describe briefly the two techniques utilized in our protocol, and the notations used, respectively. As the main contribution, in Section 6, we present

the details of the e-voting protocol based on blockchain. We discuss some possible improvements and extensions of our protocol in Section 7, which could be used for some specific scenarios. In what follows, we analyze the privacy and security of this protocol from different aspects in Section 8. Finally, Section 9 concludes this paper by some future work.

2 Related Work

There has been a lot of work on remote e-voting protocols using cryptographic tools, such as [13, 14], etc. In some cases, a trusted third party (TTP) is involved to make e-voting systems more easily implemented and controlled. However, a powerful TTP may also become the vulnerable spot of the whole system. A few efforts have been made to combine an e-voting protocol with the blockchain paradigm to design a voting protocol without a TTP, which provides anonymity and verifiability as well [15].

Zhao and Chan proposed a voting protocol [16] in 2015, which introduces a reward/penalty scheme for correct or incorrect behaviors of voters. Although the protocol has some limitations, this is the first attempt to combine e-voting with blockchain. Later in 2016, Lee, James, Ejeta and Kim proposed an e-voting protocol [17], which involves a TTP into blockchain to preserve voters' choices. Very recently, using Bitcoin [18], Bistarelli, Mantilacci, Santancini and Santini proposed another e-voting protocol [19]. This protocol divides the organizer of elections into two different parts - the Authentication Server (AS) and the Token Distribution Server (TDS), to protect voters' privacy. However, there remain some problems in this protocol, for example, it is difficult to inspect these two parts' behaviors, and it limits the extension of the voting scheme.

3 Main Contributions

The main contributions of our work in this paper are presented as follows.

1. We integrate the blockchain paradigm into e-voting procedure and come up with a feasible and general e-voting protocol without a TTP, which provides a secure and flexible voting mechanism, satisfies almost all of the main requirements for an e-voting system and weakens the power of the election organizer.
2. According to the protocol, we discuss several improvements and extensions to meet the requirements of some specific scenarios.

More precisely, we list the properties that our protocol satisfies as follows.

Public Verifiability Everyone involved the election, including spectators, who can see the voting process (recorded on blockchain), can verify the whole election's procedure and its outcome.

Individual Verifiability Each voter is able to verify individual voting procedure, e.g., whether his/her ballot has been cast and recorded successfully, counted in the final tally, etc.

Dependability Guaranteed by the cryptographic algorithms and the practical consensus mechanisms of blockchain, the protocol protects the voting procedure against dishonest behaviors and attacks.

Consistency Supported by the practical consensus mechanisms of blockchain again, all participants involved in the election, hold the same record of the voting procedure, and thus accept the same outcome of the election.

Auditability The whole voting procedure recorded on blockchain is auditable after the election.

Anonymity Only voters themselves know the information of their votes, and all ballots in the ballot box have no connection with their voters.

Transparency Due to the transparency of blockchain, the whole procedure is open to the public. This leads to more fairness and validity.

4 Main Techniques in Our Protocol

In this section, we introduce two main techniques used in our protocol - blind signature and blockchain. Blind signature is used to preserve voters' choices during the election. Contrast to "secret use" of blind signature, blockchain, a data structure derived from Bitcoin as "public use", guarantees the transparency of the election procedure.

4.1 Blind Signature

Blind signature [20] is used for signing encrypted messages with no need for decrypting them. In our protocol, it plays a crucial role in hiding voters' choices on the ballots while getting signatures.

Blind signature can be implemented in several ways, for example, see [21–23]. Assume that Alice is the message provider and Bob is the signer. The blind signature scheme is described in the following.

Bob owns a signing function S'_{Bob} only controlled by himself. The corresponding publically known inverse S_{Bob}, which satisfies $S_{Bob}(S'_{Bob}(x)) = x$, but gives no clue about S'_{Bob}. To obtain Bob's signature of the string s without revealing it, Alice relies on a computing function C_{Alice} and its inverse C'_{Alice}, both of which belong to her only, and satisfy the condition that $C'_{Alice}(S'_{Bob}(C_{Alice}(x))) = S'_{Bob}(x)$ while C_{Alice} and S'_{Bob} give no clue about x. The signing procedure is presented as follows.

1. Alice sends $C_{Alice}(s)$ to Bob.
2. Bob receives $C_{Alice}(s)$ and signs it using S'_{Bob} to obtain $S'_{Bob}(C_{Alice}(s))$. Then he sends $S'_{Bob}(C_{Alice}(s))$ back to Alice.
3. Alice uses C'_{Alice} to obtain $S'_{Bob}(s)$ according to $C'_{Alice}(S'_{Bob}(C_{Alice}(s))) = S'_{Bob}(s)$.

Following the steps above, Alice obtains $S'_{Bob}(s)$, the signature of s signed by Bob, without revealing s.

4.2 Blockchain

Blockchain is a data structure in which data is organized as blocks, and blocks connect together to form a chain. Each block's creation is based on the latest block of the most current chain, and these creations are processed by nodes in the blockchain Peer-to-Peer (P2P) network. Every creation is required to follow the consensus mechanisms, such as Proof-of-Work (PoW) used in Bitcoin and Proof-of-Stake (PoS) used in PPCoin [24]. If more than one block are created in the same short period, the whole P2P network will only accept the longest chain, which may lead to a block creating competition. This competition makes sure that the network always maintains a unique chain. According to the consensus mechanisms, nodes' dishonest behaviors, such as deviation from the original chain, will be detected and refused by other nodes.

Now we introduce the message transmission on blockchain. Every user is associated with an asymmetric key pair, i.e., a public key and a private key. As the abstract message structure depicted on the left side of Fig. 1, senders fill out the area with their public keys, receivers' public keys and message contents. Afterwards, senders use their private keys to sign messages and send them to the blockchain P2P network. In this setting, messages are collected and packed into a block during a time period. As a message spreads over the network and is recorded on the blockchain, it is obtained from the blockchain by the receivers.

Message	
Sender	
Receiver	
Message	
Sender's signature	

msg	
Sender	pk_{Alice}
Receiver	pk_{Bob}
Message	*message content*
$sign(hash(msg), sk_{Alice})$	

Fig. 1. Message Structure and An Example

We claim that our protocol can be adapted on either public blockchain[1] or permissioned blockchain[2]. We may simply hold our elections on some existing pubic blockchains[3], since the security of such blockchains is assumed to be high. It is widely believed that it is almost impossible to wield a large fraction of computational resources to seize the control of such blockchains. Alternatively, if we hold elections on a permissioned blockchain, it is possible to customize it to meet some specific requirements. However, additional programming is needed and may

[1] All user nodes have opportunities to create blocks and maintain the blockchain, e.g., Bitcoin and Etherem [25].

[2] It is controlled by an individual or a group of user nodes, and only nodes permitted by the owner of the blockchain can see the blockchain and create new blocks

[3] To deploy our protocol on them, a mechanism is naturally required to carry messages, such as Bitcoin's *OP_RETURN* instruction.

lead to improper design. In such a case, unlike existing public blockchain, there might exist security flaws and functional defects.

5 Notations

In this section, we introduce some notations that are needed in the protocol for further description. First of all, we define the participants of an election as a 3-tuple:

$$(Voters, Organizers, Inspectors)$$

- **Voters:** a set containing all eligible voters.
- **Organizers:** the set of the election organizer. Here $|Organizers| = 1$. The organizer's duties are to hold the election, verify and record eligible voters' information and interact with voters during the election.
- **Inspectors:** the set of all inspectors. Here $|Inspectors| \geq 1$. We introduce inspectors in order to limit the organizer's power and inspect the organizer's behaviors. Inspectors also interact with voters during the election.

Then we present the definition of the data owned by participants in the following. Assume $voter \in Voters$, $organizer \in Organizers$, and $inspector \in Inspectors$, collections of data, including keys and functions, owned by different types of participants are presented by UML-like diagrams with explanations in the following.

Voters
$+\ pk_{voter}$ $-\ sk_{voter}$ $-\ pk'_{voter}$ $-\ sk'_{voter}$
$-\ C_{voter}()$ $-\ C'_{voter}()$

Organizers
$+\ pk_{organizer}$ $-\ sk_{organizer}$
$+\ S_{organizer}()$ $-\ S'_{organizer}()$

Inspectors
$+\ pk_{inspector}$ $-\ sk_{inspector}$
$+\ S_{inspector}()$ $-\ S'_{inspector}()$

- C_{voter} and C'_{voter} are a pair of functions for blind signature as mentioned in Section 4.1.
- pk_p and sk_p, here $p \in \{voter, organizer, inspector\}$, are a pair of asymmetric keys for the blockchain. The public key pk_p is used for participant's identification, while sk_p is the private key preserved by p secretly. This pair of two keys is mainly for communicating with others upon blockchain.
- pk'_{voter} and sk'_{voter} are another pair of asymmetric keys. Yet unlike pk_{voter} and sk_{voter}, both of the two keys are kept secret, and are mainly used for casting the ballot anonymously.
- S_p and S'_p, here $p \in \{organizer, inspector\}$, are a pair of functions used for signing ballots as mentioned in section 4.1. The organizer/inspector keeps the signing function S'_p secret, and makes its inverse S_p public.

In what follows, we define two operations, which may occur in participants' message communication via the blockchain, as well as voters' identity verification.

- **verifyVoter(voter):** a function used by the organizer and inspectors to check whether a voter $\in Voters$, has not voted yet and sent the ballot on time.
- **hash(m):** a secure hash function, such as SHA-256 employed in Bitcoin.

For simplicity, we use public keys to represent both senders and receivers and the notation $\rightarrow$ to represent message transmissions in the later. For instance, we use $pk_{Alice} \xrightarrow[msg]{message\ content} pk_{Bob}$ to represent a message msg from Alice to Bob. The right side of Fig. 1 shows the detail of this transmitted message.

6 Details of the Protocol

In this section, we introduce our protocol by stating the three phases in detail. Without loss of generality, we suppose that there is an election and three participants - Alice, Bob and Carol, where Alice is a potential voter, Bob $\in Organizers$ and Carol $\in Inspectors$. For now, we assume that there is exactly one inspector in this election.

6.1 Pre-voting Phase

First of all, Alice, as one of the potential voters, registers as an eligible voter with the organizer Bob via the channel he provides, by submitting her personal information together with her public key pk_{Alice}. When Alice finishes registration, Bob will put Alice with some of her information, including her id and pk_{Alice}, into the eligible voter set $Voters$.

After the registration procedure, Bob publishes the set $Voters$ of all eligible voters, each of whom has finished registration and is deemed as an eligible voter in the election, on the official website for inspection.

6.2 Voting Phase

The voting phase consists of two sub-phases: *ballot preparation* and *ballot casting*.

Ballot Preparation A correct voting message can be formulated by the following string (vote string):

$$\overbrace{\underbrace{\text{Choice Code}}_{x}\underbrace{0000\cdots0000}_{y}\underbrace{\text{Random String}}_{z}}^{n}.$$

The length n of the vote string varies depending on specific elections. The first x bits are the choice code, which represents the voter's choice, followed by

a y-bit zero string, which is an indication of a well-formed vote. The last part is a z-bit random string, which distinguishes different votes containing the same choice code.

Suppose that in a two-candidate election, there are two candidates X and Y. We may simply use two bits to constitute the choice code, i.e., 01 for X, 10 for Y, and 00 for abstention. This coding scheme could be accordingly extended when there are more than two candidates, or voters are allowed to choose more than one candidates. In this two-candidate scenario, Alice takes the following steps to vote for Y, and we stress that all message transmissions will be recorded on blockchain.

1. Alice creates the vote string V based on her choice (10 as the choice code) and a z-bit pseudo-randomly generated string:

$$\overbrace{\underbrace{10}_{2}\underbrace{0000\cdots0000}_{y}\ \underbrace{0110\cdots1110}_{\text{z bits random string}}}^{n}.$$

2. Alice uses the hash function to obtain $hash(V)$, and then gets $C_{Alice}(hash(V))$.

3. Alice creates two messages containing $C_{Alice}(hash(V))$, then signs **one of** the messages $msg_{Alice\ to\ Bob}$ and sends it to Bob. We can describe this transmission in the diagram below:

$$pk_{Alice} \xrightarrow[msg_{Alice\ to\ Bob}]{C_{Alice}(hash(V))} pk_{Bob}.$$

4. Bob receives this message and check whether $verifyVoter(Alice)$ returns *TRUE*. If so, he will sign $C_{Alice}(hash(V))$ using S'_{Bob}, create a new message $msg_{Bob\ to\ Alice}$ and send it back to Alice. Otherwise, he will ignore the message. The diagram is presented below:

$$pk_{Bob} \xrightarrow[msg_{Bob\ to\ Alice}]{S'_{Bob}(C_{Alice}(hash(V)))} pk_{Alice}.$$

5. Once Alice receives the message sent from Bob in Step 4, she will send **the other** message created in Step 3 to Carol via blockchain. Then Carol uses $verifyVoter(Alice)$ as Bob has done and checks whether Alice has sent the same $C_{Alice}(hash(V))$ to Bob. If both are *TRUE*, Carol will sign the $C_{Alice}(hash(V))$ and return it to Alice too. If not, she will ignore this message. Two diagrams of this step are in the following:

$$pk_{Alice} \xrightarrow[msg_{Alice\ to\ Carol}]{C_{Alice}(hash(V))} pk_{Carol}\ , \qquad pk_{Carol} \xrightarrow[msg_{Carol\ to\ Alice}]{S'_{Carol}(C_{Alice}(hash(V)))} pk_{Alice}.$$

6. According to *blind signature*, Alice uses C'_{Alice} to obtain $S'_{Bob}(hash(V))$ and $S'_{Carol}(hash(V))$ based on the messages from Bob and Carol, respectively.

At this time, Alice owns $S'_{Bob}(hash(V))$ and $S'_{Carol}(hash(V))$, which constitute the main information for ballot creation. These two signatures indicate that Alice has been verified as an eligible voter and her choice have been confirmed by both Bob and Carol.

Ballot Casting We call a ballot *valid* if it contains a voting string in a correct format, together with the organizer's and all inspectors' signatures. Now Alice holds all components of a valid ballot - V, $S'_{Bob}(hash(V))$ and $S'_{Carol}(hash(V))$. In this part, she needs to create a ballot and cast it anonymously. This ballot casting sub-phase contains the following two steps:

1. Alice creates a new message $Ballot$, which includes V, $S'_{Bob}(hash(V))$ and $S'_{Carol}(hash(V))$.

2. Alice sends the message $Ballot$ using the pk'_{Alice} and the sk'_{Alice}, a pair of asymmetric keys for the blockchain owned by herself secretly as mentioned in Section 5, to Bob via blockchain. The diagram is showed below:

$$pk'_{Alice} \xrightarrow[Ballot]{V||S'_{Bob}(hash(V))||S'_{Carol}(hash(V))} pk_{Bob}.$$

Once the $Ballot$ is recorded on the blockchain, Alice's vote is then completed. Fig. 2 illustrates the message transmission of the protocol.

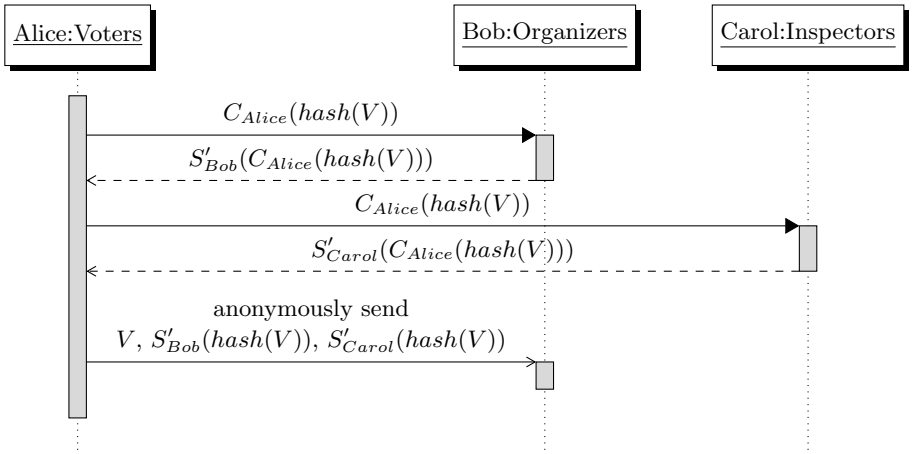

Fig. 2. Sequence Diagram of The Protocol.

6.3 Post-voting Phase

After the voting phase, the organizer Bob is required to collect all valid ballots. To this end, Bob creates a set $AllBallots$, which includes all ballots he has received. Then Bob runs Algorithm 1 to obtain the set $ValidBallots$ that includes all valid ballots.

When obtaining the set $ValidBallots$ of all valid ballots, Bob starts the tally, which produces the result of the election. We remark that the tally procedure is specified for different election scenarios. Then Bob publishes $AllBallots$ and

Algorithm 1 To Obtain All Valid Ballots

Input: $AllBallots$: the set of all ballots Bob has received
Output: $ValidBallots$: the set of all valid ballots

```
for each b ∈ Ballots do
    if isCorrectFormat(b) & hasAllSignature(b) & isCastOnTime(b) & hasNotBeenCounted(b) then
        ValidBallots ← ValidBallots ∪ {b}
    end if
end for
```

$ValidBallots$, together with $Result$, the election result. All the information published can be used to verify the election procedure, including whether all valid ballots are collected in $ValidBallots$, whether $Result$ is correct, etc. The verification can be done by all participants, and those who have the permission to see the blockchain. This is guaranteed by the properties of blockchain, since all message transmissions are recorded on blockchain, which also provides information for the latter audit procedure.

7 Improvements and Extensions

In this section, we discuss some possible further improvements and extensions when applying the e-voting protocol in special elections and scenarios.

7.1 Privacy of Data Transmission

In our protocol, the communication through the blockchain network may divulge voters' IP addresses, which may lead to the exposure of connections between voters and ballots via network analysis. To enhance voters' privacy, we recommend voters to use anonymity services like proxies or TOR [26], with which voters can hide their IP addresses.

7.2 Data Confidentiality and Neutrality

According to our protocol, because of the transparency property from blockchain, ballots are visible when they are cast to the blockchain network. This exposes the progress of the election during the voting phase, and may greatly influence the outcome of the election. Here, we provide two possible solutions for this problem.

A direct solution is to control the access of blockchain, by simply employing a permissioned blockchain for the election. The permissioned blockchain is more flexible and there exist several promising solutions of access control (like [27]). However, providing certain extent of data confidentiality, transparency is somehow lost.

To keep transparency, we may also introduce a ballot encryption mechanism here. The basic idea is: voters encrypt the message $Ballot$ using one new public

key provided by organizer, and the organizer open the corresponding private key for ballot decryption before the post-voting phase. The election then becomes confidential, and all ballots are enclosed until the end of the voting phase.

7.3 Dishonest Behaviors from the Organizer and Inspectors

Corruption may happen if the organizer and inspector conspire together, since both of their signatures are components of a valid ballot. To avoid this kind of dishonest behaviors, we can introduce more inspectors such that the corruption cost is greatly increased.

8 Security Analysis

In general, the security of our e-voting protocol mainly relies on that of blind signature and blockchain. In the following, we discuss several security issues on this protocol.

8.1 Voters' Privacy

Voters' privacy is mainly protected by the blind signature and hash functions. From $C_{Alice}(hash(V))$, $S'_{Bob}(C_{Alice}(hash(V)))$ or $S'_{Carol}(C_{Alice}(hash(V)))$, an attacker has no knowledge of the messages V. In addition, it is also difficult to figure out the connection between voters and ballots if Alice casts her ballot at a random time slot, since the second pair of keys pk'_{Alice} and sk'_{Alice} is not public.

8.2 Ballot Manipulation and Forgery

In our protocol, manipulated ballots will be rejected by the network due to wrong signatures and incorrect formats of ballots. Meanwhile, for potential dishonest organizer and inspectors, it is impossible to return a wrong signature with the purpose of invalidating voters' ballots, since wrong signatures associated with the original messages can be detected on blockchain. When the attack aims to forge ballots, it could not succeed if there exists at least one honest organizer or inspector.

8.3 Network Attack

When there are enough honest nodes in the P2P network, the intercepted ballot will be resent, and then accepted by those honest nodes and recorded on blockchain. In conclusion, malicious nodes can hardly influence the voting procedure. We also note that replay attacks do not work to forge multiple ballots in this protocol, because if two ballots have the identical voting string, they will be counted only once.

8.4 Ballot Collision

Ballots are identified by the *choice code* and the *random string* in the voting string. If it happens that different voters produce the same string, a collision occurs and one of the two ballots will be invalid. According to the *Birthday Attack*, for 128-bit voting strings, the probability that collisions occur is less than 10^{-18}. Therefore we can ignore the existence of collisions provided that the random string is long enough.

9 Conclusion

Using blind signature and blockchain, we proposed an e-voting protocol, which introduces a lot of desirable properties from blockchain. It would be nice if some details of this e-voting protocol could be further optimized and implemented. For example, because of intentional transparency of blockchain, it seems difficult to satisfy *coercion-resistance* (Voters should not be able to prove how they voted.) unless we implement *access control* using permissioned blockchain. Meanwhile, transparency might be reduced due to the tradeoff. How to balance the two properties: transparency and coercion-resistence better may constitute a possible direction of future work.

References

1. Fouard, L., Duclos, M., Lafourcade, P.: Survey on electronic voting schemes. supported by the ANR project AVOTÉ (2007)
2. Gritzalis, D.A.: Secure electronic voting. Volume 7. Springer Science & Business Media (2012)
3. Carter, L., Bélanger, F.: Internet voting and political participation: an empirical comparison of technological and political factors. DATA BASE **43**(3) (2012) 26–46
4. Madise, Ü., Martens, T.: E-voting in estonia 2005. the first practice of country-wide binding internet voting in the world. In Krimmer, R., ed.: Electronic Voting 2006: 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting.CC, August, 2nd - 4th, 2006 in Castle Hofen, Bregenz, Austria. Volume 86 of LNI., GI (2006) 15–26
5. Goodman, N.J.: Internet voting in a local election in canada. In: The Internet and Democracy in Global Perspective. Springer (2014) 7–24
6. Brightwell, I., Cucurull, J., Galindo, D., Guasch, S.: An overview of the ivote 2015 voting system (2015)
7. Ryan, M., Grewal, G.S.: Internet voting: coming to a computer near you, though more research is needed to eliminate the risks. Democratic Audit Blog (2014)
8. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security analysis of the estonian internet voting system. In Ahn, G., Yung, M., Li, N., eds.: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014, ACM (2014) 703–715

9. Halderman, J.A., Teague, V.: The new south wales ivote system: Security failures and verification flaws in a live online election. In Haenni, R., Koenig, R.E., Wikström, D., eds.: E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings. Volume 9269 of Lecture Notes in Computer Science., Springer (2015) 35–53
10. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982, IEEE Computer Society (1982) 160–164
11. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014, IEEE Computer Society (2014) 443–458
12. Wüst, K., Gervais, A.: Do you need a blockchain? IACR Cryptology ePrint Archive **2017** (2017) 375
13. Adida, B.: Helios: Web-based open-audit voting. In van Oorschot, P.C., ed.: Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA, USENIX Association (2008) 335–348
14. Ryan, M., Grewal, G.S., Chen, L.: Du-vote: Remote electronic voting with untrusted computers. In Cortier, V., Robbana, R., eds.: Proceedings of the Formal Methods for Security Workshop co-located with the PetriNets-2014 Conference , Tunis, Tunisia, June 23rd, 2014. Volume 1158 of CEUR Workshop Proceedings., CEUR-WS.org (2014) 4
15. Zou, X., Li, H., Sui, Y., Peng, W., Li, F.: Assurable, transparent, and mutual restraining e-voting involving multiple conflicting parties. In: 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014, IEEE (2014) 136–144
16. Zhao, Z., Chan, T.H.: How to vote privately using bitcoin. In Qing, S., Okamoto, E., Kim, K., Liu, D., eds.: Information and Communications Security - 17th International Conference, ICICS 2015, Beijing, China, December 9-11, 2015, Revised Selected Papers. Volume 9543 of Lecture Notes in Computer Science., Springer (2015) 82–96
17. Lee, K., James, J.I., Ejeta, T.G., Kim, H.J.: Electronic voting service using blockchain. JDFSL **11**(2) (2016) 123–136
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
19. Bistarelli, S., Mantilacci, M., Santancini, P., Santini, F.: An end-to-end voting-system based on bitcoin. In Seffah, A., Penzenstadler, B., Alves, C., Peng, X., eds.: Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017, ACM (2017) 1836–1841
20. Chaum, D.: Blind signatures for untraceable payments. In Chaum, D., Rivest, R.L., Sherman, A.T., eds.: Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982., Plenum Press, New York (1982) 199–203
21. Chaum, D.: Blind signature system. In Chaum, D., ed.: Advances in Cryptology, Proceedings of CRYPTO '83, Santa Barbara, California, USA, August 21-24, 1983., Plenum Press, New York (1983) 153
22. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In Brickell, E.F., ed.: Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Volume 740 of Lecture Notes in Computer Science., Springer (1992) 89–105

23. Camenisch, J., Piveteau, J., Stadler, M.: Blind signatures based on the discrete logarithm problem. In Santis, A.D., ed.: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. Volume 950 of Lecture Notes in Computer Science., Springer (1994) 428–432
24. King, S., Nadal, S.: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, August **19** (2012)
25. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151** (2014)
26. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In Blaze, M., ed.: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA, USENIX (2004) 303–320
27. Zyskind, G., Nathan, O., Pentland, A.: Enigma: Decentralized computation platform with guaranteed privacy. CoRR **abs/1506.03471** (2015)