Towards Blockchain-Based E-Voting Systems

Chiara Braghin[1(✉)], Stelvio Cimato[1], Simone Raimondi Cominesi[1], Ernesto Damiani[1,2], and Lara Mauri[1]

[1] Dipartimento di Informatica, Università degli Studi di Milano, Milan, Italy
{chiara.braghin,stelvio.cimato,simoneraimondi.cominesi, lara.mauri}@unimi.it
[2] EBTIC Laboratory, Khalifa University, Abu Dhabi Campus, PO Box 127788, Abu Dhabi, UAE
ernesto.damiani@ku.ac.ae

Abstract. Electronic voting is one of the most challenging cryptographic problems, since the developed system should guarantee strong and sometimes contrasting security properties. Blockchain technology can be of help providing for free some important guarantees such as the immutability and transparency of the votes using a distributed ledger. In this paper we propose a blockchain based e-voting system, which is lightweight, since it does not rely on strong cryptographic primitives, and efficient, since it improves over previous proposals in terms of both execution time and associated cost for the required infrastructure. We provide the description of a proof of concept system together with the cost and performance analysis.

Keywords: Cryptographic protocol · Blockchain · Ethereum · E-Voting

1 Introduction

Elections play a vital role within the context of a democratic society. In particular, systems for electronic voting (e-voting) are a pivotal technology currently subject of research, opening new opportunities for the development of e-democracy. As a matter of fact, e-voting can help increase the level of citizen participation in the decision-making process by reaching a wider public and allowing individuals to express their opinion more easily. For example, compared to traditional paper-based systems, it has the obvious benefit of making long-distance voting easier, especially for armed forces and other voters overseas, and of improving accessibility for elderly and physically impaired, thus increasing the voter turnout. The approach to voting through digital systems is a continuous evolving domain with the overall goal to make the entire election procedure secure, verifiable and transparent. A second but not secondary purpose is also to increase efficiency, minimise the negative factor of human error and reduce cost of elections.

W. Abramowicz and R. Corchuelo (Eds.): BIS 2019 Workshops, LNBIP 373, pp. 274–286, 2019.
https://doi.org/10.1007/978-3-030-36691-9_24

In general, e-voting protocols rely on the existence of a public bulletin board to support verifiability, since all voters can transparently access and check the correct submission and counting of the votes [7]. E-voting is a natural area where blockchain technology has been exploited since its first appearance. Most of the proposals adopt blockchain framework as an immutable centralised database where votes can be stored ensuring a number of security guarantees.

Currently, there are several commercial remote e-voting protocols, namely BitCongress, FollowMyVote, and TIVI; some of them have been applied for informal and consultative voting, some others have been deployed for city or national voting [8]. Recently, online voting has been adopted in the USA, where West Virginian residents serving overseas were able to cast federal election ballots using a smartphone app [14]. Voters registered by taking a photo of their government-issued identification and a selfie, and then uploaded them via an app that has been developed by Voatz, a Boston company in charge of the development of the voting infrastructure. Using facial recognition software, voters used the app to cast their ballots, that were anonymised and stored on the blockchain.

The proposals above have some scalability and performance issues, restricting the number of participants, or requiring strong cryptographic primitives that are computationally expensive. In all cases, the resulting system fits to small scale elections or has lower performance in terms of time and/or cost. Debate on the advantages coming from the replacement of traditional balloting and on the potential risks of mobile voting technology is still ongoing [17].

However, we think that most of the issues can be well addressed by proposing simple voting frameworks where usability and security guarantees are well balanced, using the technical resources available. To this aim, in this paper we propose a remote e-voting system based on the deployment of standard cryptographic techniques, in particular we adopt chameleon hash as a means to add coercion-resistance property to the resulting system.

Our contribution

- We propose a lightweight and efficient contract-based e-voting framework and discuss the security properties it satisfies. Our proposal is implemented using a single smart contract and can be scaled to manage from small community to country-wide elections.
- As a proof of concept, we describe a prototype implementation relying on the Ethereum blockchain. For this purpose, we describe the implementation on Ganache, a local blockchain framework, and give a detailed representation of the associated costs for generic elections, evaluating also the performance in comparison with previous proposals.

2 Related Works

Since their first appearance, blockchain frameworks have revolutionised the financial sector creating a large number of crypto-currencies available in place of traditional currencies, and paving the way for deep transformation in the digital

economy [15]. The success of blockchain is mostly derived from the possibility to remove the role of the banks that usually play the role of the central authority managing a financial ledger and ensuring the correctness of all the financial transactions. On the contrary blockchains provide a distributed ledger managed by a peer-to-peer network where all the members interact and where a consensus mechanism gives the possibility to verify the transactions and validate the state of the shared ledger [9]. Taking advantage of the distributed consensus mechanism, practical applications of blockchain technology beyond the financial sector have been started in different fields such as health, science, government, culture and art [1].

Several digital voting systems have been proposed to improve the public electoral process in terms of costs and time efficiency, and achieve more direct form of democracy. Recently, different proposals consider blockchain frameworks as a means to get transparency and security guarantees. Some proposals use smart contracts to perform the voting phases.

Mc Corry et al. in [10] present an implementation of the Open Vote Network over the Ethereum blockchain. Open Vote Network [5] is a decentralized two-round protocol designed for supporting small-scale boardroom voting, where in the first round registrations of the voters are collected, while in the second round all voters can cast their vote. The system allows voting yes/no by collecting zero-knowledge proofs according to the Cramer et al. technique [2]. In the Ethereum implementation, two smart contracts are deployed, one for the voting and one for the cryptography computations devoted to the creation and the verification of zero knowledge proofs needed in the protocol. It is worth to notice that the current limitation in the Ethereum platform and the cost of the deployed contracts limit the usability of this approach to yes/no elections including a restricted number of voters (less than fifty).

In [3], the Broncovote framework for university-scaled elections is presented. The system is deployed on the Ethereum blockchain and relies on the Paillier homomorphic encryption to achieve voters' privacy. The implementation include three contracts: one for setting up the ballot and defining the candidates for the election; the second is the Registrar contract used by the administrator to allow potential voters to register; the third contract allow voters to cast their votes using homomorphic encrypted ballots. To encrypt the votes and to update the vote count, Broncovote interacts with an external server which perform the needed operations. Also in this case, considerations on the costs in terms of gas needed to perform the transactions limit the adoption of the system to elections involving a small number of participants (about thirty).

The implementation of a national e-voting system (examining the characteristics of Iceland, home country of the authors) based on blockchain has been considered in [6]. Different roles for the actors have been distinguished, and different blockchain frameworks analysed for the implementation, among Exonum, Quorum and Geth. The presented election scheme requires each voter to go at a voting district and makes use of a private Ethereum blockchian.

3 Background

3.1 Blockchain and Smart Contracts

A blockchain system [13] is a distributed peer-to-peer framework where participants are involved in transactions without trusting themselves, not relying on any trusted intermediary, but still having a way to verify the exchanges. Transactions are registered in a *distributed ledger* that does not need a central repository of information, but realises a distributed data structure replicated and shared among all the members of the network. All transactions that have been finalised in the blockchain are registered in a permanent and verifiable way. In a blockchain, each block is connected to the rest of the chain using the cryptographic hash of the previous blocks, being in this way resistant to any modification, since once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks. Transactions are verified and inserted in the chain by special nodes which are called *miners*. Their work consists in checking the sender and the content of the transaction and in generating a new block of transactions only after that a computationally expensive task, the so called *Proof of Work*, has been solved. The generated block can be then propagated to the rest of network where the other nodes can validate its correctness.

Some blockchain framework give the possibility to define and execute *smart contracts*, that are executable pieces of code stored and running on the blockchain to facilitate, execute and enforce the terms of an agreement. A smart contract executes independently and automatically according to the data that was included in the triggering transaction, and the blockchain network acts as a distributed VM.

3.2 Chameleon Hash

Chameleon hash functions are particular kinds of collision resistant hash functions which allow the existence of a trapdoor. We will use them to allow voters to check if their vote has been recorded correctly in the blockchain (see Sect. 4), avoiding coercion. If the trapdoor is not known, the function has the same security properties of ordinary collision-resistant hash functions, while the user can use the the trapdoor to easily find a collision.

A *chameleon hash function* is composed by three procedures:

- Gen takes as input a security parameter 1^k and outputs the evaluation key ek
- CH takes as input the evaluation key ek, a message m and a random value r and outputs a hash value h
- CH^{-1} takes as input the trapdoor tk, two messages m, m' and a random value r and returns a value r', such that $CH(ek, m, r) = CH(ek, m', r')$.

Here, we use the instantiation of chameleon hash based on the discrete logarithm problem similarly to what reported in [4]. The procedure Gen selects a

group G of prime order q of elements in Z_p^* with generator g. After selecting an element x in Z_q and computing the value $h = g^x$ the evaluation key ek is defined as $ek = (G, g, h)$ and the trapdoor key tk is defined as $tk = (ek, x)$. The procedure CH is defined for a message m and the random value r to output $ch = g^m * h^r$. The procedure $\mathsf{CH}^{-1}(m, r, m')$ outputs the value r' such that $r' = (m - m') * x^{-1} + r$.

4 A Blockchain-Based E-Voting System

In this section, we describe the architecture of a blockchain-based remote e-voting system, abstracting from the blockchain on which it is based on. The only assumptions made on the actual system are that smart contracts are supported and that users own a registered account to the system, with an associated public key. In this way, the security issues discussed in Sect. 4.1 remain valid in any scenario.

The main actors of the proposed framework are the *administrator* and the *voter*. The administrator represents the institution organising the election, thus in charge of configuring ballots with the list of candidates, registering eligible voters, deciding the lifetime of the election, and deploying the smart contract. An eligible voter, to cast her vote, just needs an Internet connection and a registered account to the blockchain system used (e.g., Ethereum or Bitcoin).

By following the classical high-level models of election systems, we focus on the three major phases of an e-voting process: *(i) Pre-election Phase*, in which candidates and eligible voters are registered; *(ii) Election Phase*, the actual voting phase in which only eligible voters are able to cast ballots from any location that is accessible through the Internet; and *(iii) Post-election Phase*, in which votes are published.

In our framework, the election process consists of the following steps:

1. *Election set-up - Phase I*: Each municipality configures the election ballots and includes in a white-list all the eligible voters.
2. *Voter registration*: An eligible voter must register with her municipality the public key of the blockchain account she will use to vote. The voter authenticates herself by presenting some personal data, such as social security number, ID number and address. The voter also deposits the evaluation key ek, keeping secret the element x needed to compute the trapdoor key. At the end of the registration phase, the voter is given an url that will be active during the election period (a sort of a virtual polling place), and that she will use to vote.
3. *Election set-up - Phase II*: Each municipality deploys a contract containing the list of the public keys associated with each voter, the list of candidates and an associated integer value representing the votes obtained by the candidate (and set to zero at the beginning). The contract also contains a *voting function* that is triggered only when the eligible voter casts her vote from the url she has been given during the registration phase.

4. *Voting*: The voter goes to the url and casts her vote by selecting the candidate from the (closed) list of candidates available for her municipality. The *voting function* checks if the voter is eligible and has not voted yet, then, if the candidate is valid, it records that the voter has voted and it increments the voting count (without connecting the vote with the voter). Moreover, it provides as receipt the result of the chameleon hash function computed using the evaluation key, a random value and the message containing the details of the vote she cast.
5. *Publication of Results*: At the end of the voting phase, the results are published, reporting the vote count for each candidate and the vote receipts as computed in the previous step.

4.1 Security Issues

Every voting system, either online voting or traditional paper-based voting, should satisfy specific security requirements. In this section, we list the major desirable security properties and discuss if and how properties are held. Since our system uses the blockchain to record vote counts and voting operations, the system inherits some of the properties "out of the box".

- *Eligibility:* only voters with the right to vote are allowed to cast a vote. This property states that only legitimate persons can vote and every vote cast must be counted only once.
  In our framework, only eligible voters will cast a vote: before recording the vote, the smart contract checks if the public key associated to the private key used to sign the voting transaction is the one presented to the administrator during the registering phase (thus recorded in the contract).
- *Correctness:* every valid vote cast is counted. This property implies that in order to correctly count submitted votes, those submitted by unauthorised or unauthenticated voters must be classified as invalid and hence not counted.
  In our framework, the smart contract (and the votes) are recorded in the blockchain, which is resistant to modification of data by construction.
- *Uniqueness:* no voter is able to vote more than once.
  In our system, double voting is prevented by the fact that the contract records (and checks) if a voter has already voted before counting the vote as valid.
- *Integrity:* no one should be able to modify, forge, or delete votes without detection.
  In our system, the votes (and the smart contract) are recorded in the blockchain, which is resistant to modification of data by construction.
- *Vote anonymity:* neither election administrators nor anyone should be able to determine how any individual voted.
  In our system, no individual vote is traceable back to the voter since the fact that a person has voted and the value of her vote are two separate pieces of information.
- *Auditing:* every voter can check whether his vote has been counted or not. This property refers to the ability of the voter to verify that his ballot choice

has been really counted, thus implying trust in the vote tallying process by all parties involved.
In our system, the voter can check if the receipt of her vote (computed using the chameleon hash) is registered in the vote count in order to be sure that her vote has been correctly recorded.

– *Coercion-resistance:* this requirement ensures that the voter can deceive a coercer into thinking that he has voted for some designated choices as instructed, when the voter has in fact cast a ballot according to her own opinion.
In our system, although the usage of the chameleon hash function allows the voter to check (and eventually prove) her vote, the coercer is not able to determine or not the targeted voter behaved as instructed.

5 Implementation

In this section, we describe *Chaincracy*, an Ethereum-based prototype implementation of the framework described in the previous section. It aims at reaching the highest number of eligible voters, since also users who cannot reach their polling places for different reasons, will be enabled to cast their vote by accessing a user-friendly web page showing the election ballot of his municipality.

5.1 Environment and Tools

At the moment, Ethereum is one of the most popular public blockchain platform for developing smart contracts, since it provides a built-in high-level Turing-complete language called Solidity (resembling common languages such as C++, Python and JavaScript). Accounts represent the main *entities* in Ethereum, since their configuration defines also the state of the Ethereum network. It is possible to distinguish between two types of accounts: *externally owned accounts* (EOAs), and *contract accounts*. The first category represents users interacting via transactions with the blockchain, while the second category represents the interactions due to the execution of smart contracts. A contract can change the state of the network on the basis of the transaction it receives, and usually can read or write data to its private storage, or store money into its account balance, or send/receive money from other users or other contracts, or, finally, send messages to other contracts to trigger their execution.

More in detail, for the implementation of our system prototype, we used Truffle web framework to write, compile and debug Solidity smart contracts; Ganache, a local Ethereum blockchain, to deploy the smart contract and run tests; and MetaMask to manage voters' accounts and to transact with a smart contract deployed on the blockchain from inside a JavaScript and a web application.

Truffle web framework [16] is a development environment, a testing framework and an asset pipeline for Ethereum, offering automated contract testing and some kind of debug feature.

Ganache is a local blockchain RPC server to test and develop against, integrated into Truffle and available both as a desktop application, as well as a command-line tool. It provides ten initial accounts pre-funded with 100 Ether along with a twelve-word seed phrase for re-generating those accounts. The seed phrase can be used to initialize a MetaMask client with the same accounts. The interface shows: the accounts generated and their balances, each block as mined on the blockchain, along with gas used and transactions, a list of all transactions run against the blockchain, and the logs for the server.

MetaMask [11] is a browser's plugin available for Chrome, Firefox, Opera and Brave allowing a user to access the Ethereum network. The plugin injects a JavaScript library developed by the Ethereum core team called web3.js into the namespace of each page loaded, thus providing the browser with APIs to make read and write requests on the Ethereum blockchain from regular websites. As a consequence, it allows users to make Ethereum transactions through regular websites, interacting with a local or remote Ethereum node, using a HTTP or IPC connection, without running a full Ethereum node. The tool also provides users with a secure identity vault, working as an Ethereum wallet, which allows anyone to manage identities across different websites and use them to sign blockchain transactions. Keys are stored encrypted on the browser, not on a remote server. So far, MetaMask has proven to be quite secure and there have been no successful hack attacks that have resulted in currency losses.

5.2 Implementation Details

In this section, we describe the most important components of the e-voting system we implemented, that are the Solidity smart contract, the web page presented to the voter, and the JavaScript file interacting with the server and the system to communicate data and update the status of the voter (see Fig. 1).

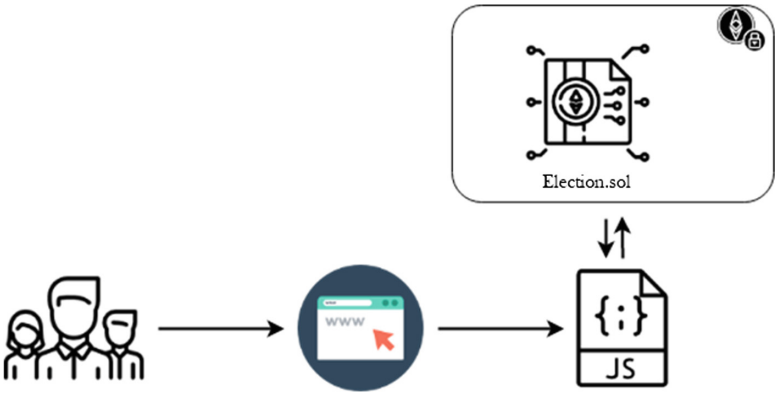

Fig. 1. *Chaincracy* voting framework.

More in detail:

- Election.sol: It is the (only) smart contract file, where the election candidates, the voters' public keys and the vote counters are recorded. Moreover, it contains a set of functions doing the vote count, controlling the eligibility of the voter, the correctness of the whitelist and of the candidates, avoiding double voting, etc.
- Index.html: This file represents the virtual polling place. It is the file requested by the url given to the voter after the registration phase. If requested during the election time, it calls the App.js file that retrieves the data from the blockchain and shows it to the voter.
- App.js: The file contains the configuration of web3 library, retrieves the candidates list and loads it in the web page displayed to the users. The file updates the page according to the status of the voter, for example showing a new page without the ability to vote if the voter has already cast her vote.

In the prototype implementation, from the point of view of the user, to participate to the voting process, each user should follow these steps:

1. Download and install the MetaMask plugin in her browser.
2. Either create an external user Ethereum account, or import an existing one from another wallet.
3. Before the election phase, register the public key of the chosen account to her municipality and record the voting url.
4. During the election period, visit the url she was given. The page (see Fig. 2) shows the list of candidates of her municipality retrieved from the blockchain thanks to the functions contained in the App.js file and her public key. In case she has already voted, she sees a page telling her that double voting is not permitted.
5. Select the candidate she wants to vote for, and cast her vote by clicking over the Vote button.
6. Confirm the transaction by means of the MetaMask interface.
7. At the end of the voting period, if the voter visits the url she was given during the registration phase, she is shown the election results for her municipality (see Fig. 3).

Fig. 2. The web page displayed to the voter (at her first visit), during the ballot.

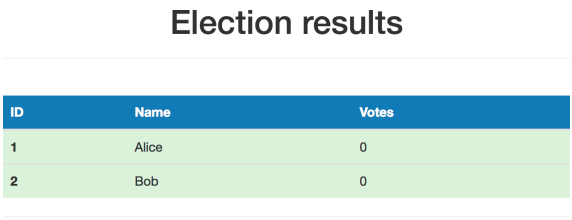

Fig. 3. The web page displayed to the voter when the voting phase is finished.

Extracts of the Contract. In the contract, the most important part is the vote function, where some controls are executed before incrementing the vote counter of the voted candidate.

In particular, the contract checks:

- that the voter has not voted yet (and updates her status after the vote):

```
mapping(address => bool) public voters;
...
require(!voters[msg.sender]);
...
voters[msg.sender] = true;
```

  In Solidity, mappings act as hash tables which consist of key types and corresponding value type pairs. A mapping is defined like any other variable type: here, we create a mapping called voters associating a unique Ethereum address with a boolean. It allows us to look up a specific voter with his Ethereum address, and check if he has voted or not.
- that the ballot time is still valid (in the fragment the starting time is set to December, 1st 2018 and the ballots lasts 24 hours):

```
// using unix timestamp
require (block.timestamp > 1543622400 &&
block.timestamp < 1543708800);
```

- that the public key of the voter is included in the contract list of eligible voters' public keys:

```
require(0x225f8F1c8 == msg.sender || ... )
```

  This means that the public key has been registered by an eligible voter during the registration phase at her municipality.

5.3 Cost Analysis of the System

The currency used within Ethereum network is called *Ether* (ETH). Computation within the blockchain and the EVM are repaid in ETH, although the execution fee is computed in terms of *gas*. In general, one unit of gas corresponds to the execution of one computational step and gas and ETH are deliberately

Table 1. *Chaincracy*: contract costs with a 8 candidates and 6 voters ballot.

Operation	Gas used	Cost (€)
Contract deployment	91,000	2.08
Vote	64,328	1.46

decoupled, such that fluctuation of the price of ETH is caused by external market forces, while the cost of gas is directly related to the computation costs.

A *transaction* in Ethereum defines the data that are signed by the entity starting the exchange and contains a message sent from an account to another account on the blockchain. Also contracts can send *messages* to other contracts, where each contract canbe conceived as *function calls*. The content of transactions and messages is similar: both contain a recipient, a value field indicating the amount of wei (1 *ether* is 1e18 *wei*) to transfer from the sender to the recipient, an optional data field hat is the actual input data to the contract, a gasLimit field representing the maximum number of computational steps the transaction or code execution is allowed to take to be used to compute the cost of the computation. A transaction contains also a gasPrice field, representing the fee the sender is willing to pay for gas.

The execution of a contract is triggered by a message or another transaction and every instruction is then executed on every node of the network. For every executed operation there is a specified cost, expressed in a number of gas units and each transaction has a maximum ether cost that is then equal to gasLimit $*$ gasPrice.

In Table 1, we show the *gas* costs and the corresponding prices in € for the deployment of the contract and for the voting operation. At the time of carrying out the experiments, November 2018, the *ether* exchange rate was 1 ETH = €114.35, and the median gasPrice was approximately 0.0000002 ETH (20 Gwei).

We considered a small ballot with 8 candidates and 6 voters in order to make a precise comparison with BroncoVote (see Table 2). Notice that the contract deployment cost is fixed, whereas the cost of the voting operation is per person. In Italy, this voting system could help the government to save money: in 2013, the government spent 389,50 million of Euros for national elections (including regional and local elections), with 46,905,154 eligible electors (and 35,271,541 effective voters). If we consider an average cost of around €40 every 20 people (i.e., €2 per person) and about 50 million eligible voters, Chaincracy could save near 300 million Euros.

Observe that the cost of the voting operation is charged to the voter within her MetaMask account. A possible solution could be to distribute at the end of the registration phase €1.5 to each voter's MetaMask account, using the public keys listed in the voters' white-list. However, due to the floating value of Ethereum currency, the price should not be fixed, but computed running the contract locally to check the actual price. Another solution could be the usage of tokens.

Comparison with Other Systems. In Table 2, we report the *gas* costs and the corresponding prices in € for the deployment of the contracts and for the voting operation for BroncoVote, the only alternative to our system with an available cost analysis. The cost values in *gas* are taken from [3] by merging the cost of the three different contracts used by the system. Notice that the usage of our simplified architecture gives a higher performance: €51 vs €2.

To encrypt the votes and to update the vote count, BroncoVote interacts with an external server which perform the needed operations. The operations are not counted in Table 2 since they are done externally to the contract, however they should also be take in consideration. Moreover, the server, being a trusted centralised server, may be subject to the classical form of attacks, such as DoS.

Table 2. *BroncoVote*: contract costs with a 8 candidates and 6 voters ballot.

Operation	Gas used	Cost (€)
Contracts deployment	2,263,132	51.75
Vote	813,977	18.60

6 Conclusions

To date, several e-voting protocols have been developed and used in various forms. However, in order to achieve a wide adoption of such systems it is necessary to improve their resilience against potential faults due to programming errors, hardware problems and malicious behaviours that are hardly detected.

Blockchain technology seems well positioned to address many issues related to digitalisation of voting process and provide enhanced security features without affecting usability, efficiency and reliability. In this paper, we proposed a simple and efficient e-voting system based on the Ethereum blockchain. We also described a prototype implementation, evaluating its performance in terms of costs, efficiency and scalability.

The framework improves over the previous proposals, and is scalable for country-wide elections, as the analysis of the associated costs has proved. Future works will address the formal evaluation of the security properties of these protocols, and the analysis of the trust assumptions needed during the whole voting process.

References

1. Braghin, C., Cimato, S., Damiani, E., Baronchelli, M.: Designing smart-contract based auctions. In: Yang, C.-N., Peng, S.-L., Jain, L.C. (eds.) SICBS 2018. AISC, vol. 895, pp. 54–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-16946-6_5
2. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19

3. Dagher, G. G., Marella, P. B., Milojkovic, M., Mohler, J.: Broncovote: secure voting system using ethereum's blockchain. In: [12], pp. 96–107 (2018)
4. Guasch, S., Morillo, P.: How to challenge *and* cast your e-vote. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 130–145. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_8
5. Hao, F., Ryan, P.Y., Zielinski, P.: Anonymous voting by two-round public discussion. IET Inf. Secur. **4**(2), 62–67 (2010)
6. Hjalmarsson, F. P., Hreioarsson, G. K., Hamdaqa, M., and Hjalmtysson, G.: Blockchain-based e-voting system. In: 11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, 2–7 July 2018, pp. 983–986. IEEE Computer Society (2018)
7. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45664-3_10
8. Kshetri, N., Voas, J.: Blockchain-enabled e-voting. IEEE Softw. **35**(4), 95–99 (2018)
9. Mauri, L., Cimato, S., Damiani, E.: A comparative analysis of current cryptocurrencies. In: [12], pp. 127–138 (2019)
10. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 357–375. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_20
11. Metamask (2018). Metamask. https://metamask.io/
12. Mori, P., Furnell, S., Camp, O. (eds.) Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP. SciTePress (2018)
13. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System, p. 9 (2008)
14. O'Sullivan, D.: West Virginia to introduce mobile phone voting for midterm elections (2018)
15. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media Inc., Newton (2015)
16. Truffle. Truffle suite: Sweet tools for smart contracts (2018). https://truffleframework.com/
17. Yurieff, K.: Can this technology modernize how we vote? (2018)