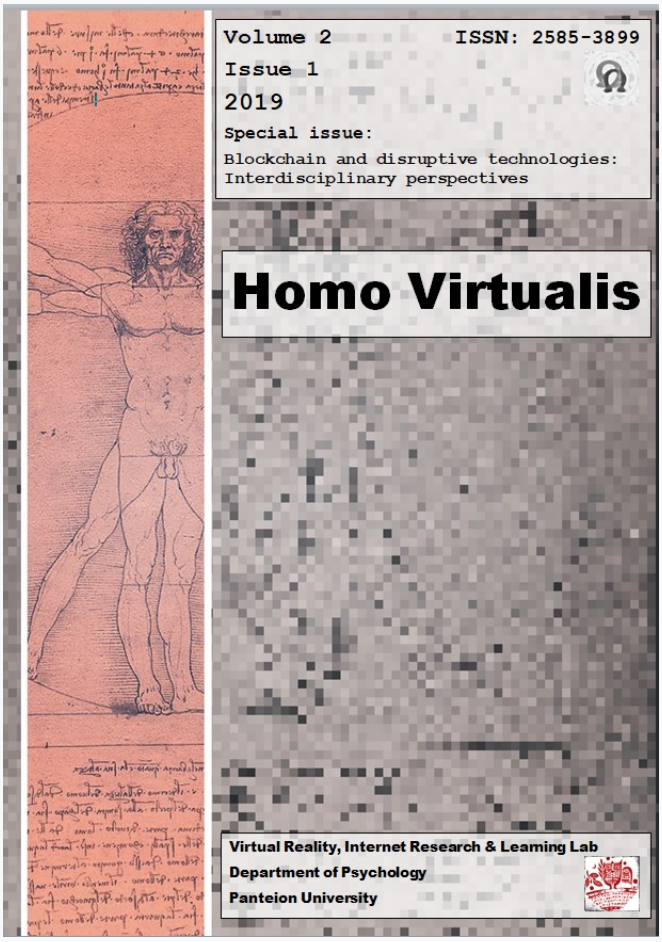
Homo Virtualis

Vol 2, No 1 (2019)

Blockchain and disruptive technologies in social sciences: Interdisciplinary perspectives



Blockchain, consensus, and cryptography in electronic voting

Panagiotis Grontas, Aris Pagourtzis

doi: 10.12681/homvir.20289

To cite this article:

Grontas, P., & Pagourtzis, A. (2019). Blockchain, consensus, and cryptography in electronic voting. *Homo Virtualis*, *2*(1), 79–100. https://doi.org/10.12681/homvir.20289

Homo Virtualis 2(1): 79-100, 2019
ISSN 2585-3899 ∫ doi.org/10.12681/homvir.20289

Blockchain, consensus, and cryptography in electronic voting

Panagiotis Grontas[1] **& Aris Pagourtzis**[2]

Abstract: Motivated by the recent trends to conduct electronic elections using blockchain technologies, we review the vast literature on cryptographic voting and assess the status of the field. We analyze the security requirements for voting systems and describe the major ideas behind the most influential cryptographic protocols for electronic voting. We focus on the great importance of consensus in the elimination of trusted third parties. Finally, we examine whether recent blockchain innovations can satisfy the strict requirements set for the security of electronic voting.

Keywords: *cryptographic voting, blockchain, electronic voting, trust, consensus*

Introduction

Voting is the cornerstone of democracy. This fact brings an immense weight to what is an otherwise simple distributed decision-making process, where a collection of agents input a set of choices into a computational process which outputs what is mutually preferred, according to some predefined rules. This process can be quite simple algorithmically - for instance the inputs can be simply counted and the choice that gathers most of the votes is outputted. However, the fact that the output binds all the agents – an entire nation in the most extreme case – for the time to come raises the stakes. This means that not only must the computation be correct, but all participants must reach *consensus* on its correctness. This applies especially to the 'losers', i.e. they must be convinced that the elections were conducted according to the rules and its result indeed expresses the majority of the agents (Schneier, 2018; Adida, 2019).

The generality of the process has made voting ubiquitous through history. As a result, elections have been conducted using many technologies according to the era and the importance of the result (Alvarez et al., 2010). For instance, a widely used method to vote is by a 'show of hands' where the voters raise their hands if they agree with a proposal. The hands are counted, and the collectively preferred result is the one for which most hands were raised. This method, while known from ancient times, is still being used today to reach

[1] PhD Candidate, School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece, pgrontas@corelab.ntua.gr
[2] Associate Professor, School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece, pagour@cs.ntua.gr

informal decisions of minimum importance. However, nobody would use it to conduct nationwide elections. The reasons are many. This method doesn't scale well, is error prone, but most importantly it cannot convince the losers that the participants expressed their true will, since the vote is visible to everybody, which makes coercion and retributions in the case of an 'incorrect' vote very possible.

The most prominent method used in elections today is the *Australian Ballot* where the voter casts her vote in a private space in organized polling stations. The ballots are sheets of paper, cast inside an envelope and collected in a sealed ballot box which is opened by a magistrate or trusted third parties. The tallying takes place publicly and is witnessed by representatives of various parties. The security of the process is safeguarded with rudimentary technical means (envelope, privacy screen, ballot box) but most importantly by *trust in institutions and the conflicting interests of the participants*. However, it is not without problems: Sometimes, long queues are formed which discourage voters, especially elderly ones. There are infrastructure and personnel costs, which must be incurred in every election. Tallying takes time, especially in large constituencies, is error prone and cannot easily accommodate for complex voting rules. For instance, to enable some forms of preferential voting – like runoff voting (Patuit, 2011), the elections must be conducted twice. Most importantly, voters cannot verify that their vote has been counted, themselves. The best they can hope for is that their trust in their representatives doing their jobs correctly is not ill founded.

Computers are considered as a possible solution to improve the traditional polling system mainly in terms of speed, efficiency, cost cutting and overseas voting. Note, that in many countries (e.g. the U.S.) the casting and tallying processes are already mechanized using lever machines (Halderman, 2012), so computerization is a logical next step. Moreover, even though the voter might use paper ballots, many voting related activities are done electronically. For instance, voter registration might be conducted separately through state agencies that use a database system. Communications of partial, hand counted, results between polling station during election day is done electronically. 'Since we can bank online, why can't we vote online' goes the argument (Simons & Jones, 2012).

Electronic voting comes in two forms: it can be done on site, combined with the traditional polling infrastructure. The voters come to predefined polling stations, enter the privacy screen but instead of casting a ballot in a ballot box, they will use a direct-recording electronic (DRE) voting machine – a computer with a touch screen – or an optical scanner to input their vote. The DRE usually records the vote in a memory module which is later extracted and inserted into a tallying machine. In some cases, the DRE prints a paper receipt which can be inserted in a conventional ballot box and used as a backup mechanism. A second possibility is remote voting. In some countries, the most prominent example being Estonia (Clarke & Martens, 2016), the voter can participate in elections using a PC or a mobile phone and cast their vote through the Internet into a government controlled database. A recent variation of this method records the vote in a public permission less blockchain instead. Finally, voting without a dedicated polling infrastructure can even occur on a small scale (boardroom voting), where few participants conduct the elections on their own by assuming extra functions apart from the role of the voter, by self-registering or self-tallying (Kiayias & Yung, 2002).

Electronic voting either done on site or remotely is not without problems. Whenever researchers could study electronic voting equipment or test an electronic voting system, they found critical security holes. Some of the most important ones are documented in

(Halderman, 2016). For instance, in 2010, researchers not only managed to bypass the built-in security mechanisms of a DRE voting machine, but also managed to reprogram it to play the game of PAC-MAN. In another case, researchers managed to find security holes in a voting system only within a few hours of looking at the source code (Wolchok et al., 2012). Recently, the Swiss Post conducted a public intrusion test of its e-voting system. The results were not encouraging, as the researchers managed to find flaws that could allow an attacker to alter or embed any number of votes, and produce a result that would not correspond to the actual voters cast (Lewis et al., 2019). Note that an attacker of voting system is rarely some lone hacker; it is more probable that it is a well-funded external or even internal government agency that is not hindered by physical restrictions such as distance. With electronic voting such an attacker can easily achieve cheating at a large scale without leaving evidence.

Security requirements for (electronic) voting

Some might rush to attribute allthe problems with electronic voting to specific characteristics of computer systems. Indeed, computer software usually functions as a 'ghost inside the machine'. The situation is very vividly described by David Dill in (Russell Michaels, 2006):

> [t]he voting booth is separated by a curtain and there is a guy behind the curtain that would write down your vote. You dictate the vote and once you're done you leave, without being able to look at the ballot. Most people in their right mind would not trust this process. The guy behind the curtain could be incompetent, hear the votes wrong and register it incorrectly or it could be that he did not like your political affiliation and prefer your vote would go to another party...

The reasons are more profound however and stem from the fact that voting is indeed a very hard problem, because it is characterized by many *conflicting* security requirements. The formalization of these into computer code might be incomplete and cause problems, but might also lead into novel and brilliant solutions, that have not arisen in real word elections, because they have more relaxed security needs, since they can make us of physical restrictions and can rely on trust in institutions which cannot be transferred to software.

We will proceed to analyse the mainsecurity properties that are expected from electronic voting systems following the framework of (Halderman, 2012) which we enrich by giving a simple description of the relevant formal definitions used in the cryptographic literature.

Correctness

When the voting system is run as designed i.e. without any adversarial manipulation, it must produce the result that corresponds to the actual votes cast by the voters. Furthermore, this means, that all the voters that check the result and the process by which it was reached receive accepting results. While this property is very simple to define and understand, it is notoriously difficult to implement due to the complexity of computer software

which is assembled from various components – built from different entities and for potentially different initial purposes – that interact in complex and often unanticipated manners.

The automated verification of these actions is impossible and software testing cannot be comprehensive enough. One approach for solving this problem is through *certification*. The producers of the voting system supply the hardware and software specifications to an independent auditor, who runs tests and decides whether the system meets certain criteria. The problem with certifications, in general, is that the auditor does not evaluate the system holistically, but tests whether specific claims made by the producer are substantiated or not. As a result, the system is only partly evaluated. A generalization of the process of certification is evaluation through *open source software*. The creator of the voting system releases the source code, and everybody can analyse it and test it as they wish. This approach was believed adequate in previous decades. The current consensus is that open source is a necessary but not sufficient condition for e-voting security. This is because it suffers from a very important flaw: namely that nobody can guarantee that the code released and scrutinized matches the code that is indeed executed during an actual election.

To escape this dead end, Rivest proposed the notion of *software independence* (Rivest, 2008) which states that the outcome of an election should not depend on the software that is used to run the election. More formally: a voting system is *software independent* if an undetected change or error in its software cannot cause an undetectable change or error in the election outcome.

The whole idea is that software problems should not propagate to the election results. In order to achieve this, the system must generate evidence that allow auditing that will detect deviations from the specified operations. A simple form of evidence is a paper trail generated as receipts from the DRE machines and checked by the voters before casting (Voter Verifiable Paper Trail – VVPAT) or statistically by independent auditors in post-election manner. A variation of this concept, called *strong software independence,* requires detection and recovery mechanisms to be in place, so that a detected error does not require re-running the election. Note that the paper trail satisfies even this stronger notion.

Another form of evidence that can be generated from a voting system utilizes mathematical tools invented by cryptography and will be the focus of the rest of this article.

End-To-End (E2E) verifiability (Chaum, 2004)

Since we cannot be sure about the correctness of a voting system its acceptance rests on our capability to verify it i.e. on its verifiability or auditability. In particular, the following processes should be verifiable:

- Votes should be *cast as intended*, which is officially called *ballot casting assurance* (Benaloh, 2006). In simple terms, this means that the user interface of the voting system should aid the voter to express his choice without interfering in any way.
- Votes should be *recorded as cast*, meaning that the voting system should internally represent the vote in a manner that does not deviate from the voter choice.
- Votes should be *counted as cast*, which means that the computation of the result should only depend on the cast votes. It should neither alter nor delete or inject votes that do not correspond to actual voters that participated.

In order to verify these actions, a voting system must allow the following audits:

- Individual verifiability. Each voter must be able to verify that his selection was correctly included in the outcome.
- Universal verifiability. Any interested party should be able to verify that the result computation was correct, i.e. all the valid inputs and only the valid inputs were included in the output without any change and that the computation adhered to the specification. A central component of universal verifiability is *eligibility verifiability* (Cortier et al., 2016), where it must be checked that every ballot cast originated from a voter with the right to vote. The observant reader should immediately see that a voting system that provides universal verifiability also provides administrative and individual verifiability.
- Administrative verifiability (Benaloh, 2008). The organizers of the election can verify that everything went as planned. This is because the election officials act as trusted proxies of the individuals, so their actions are on their behalf. This is a relaxation of universal verifiability that closely resembles real world physical elections. A goal of cryptographic voting is to eliminate the need for such checks.

There are many formal definitions of E2E verifiability – e.g. (Chaum, 2004; Juels et al., 2005; Kiayias et al., 2015; Cortier et al., 2016) and more. Their common characteristic is that the Election Authority (EA) is *considered an adversary*. Formally, E2E verifiability is modelled as a cryptographic game, where the election authority provides voters with receipts that allow them to check the casting, recording and tallying of the votes. However, its aim is to try to subvert the tally and its verification without cancelling the validity of the receipts. In this effort it can conspire with dishonest voters.

Accountability

The various forms of verifiability described above only deal with how to check that a participant complies with the protocol specification. However, this is only part of the story: How does a protocol react if a participant blames some other party, for tampered evidence? The answer to this question is subtle – as such a blaming might not be innocuous, but might aim to disrupt a normal run, instead. It must be considered though in order to enable strong software independence. The security property that deals with this problem is called *accountability* and was formally defined in (Küsters et al., 2010). Their definition concentrates around an entity, called a judge that makes sure that the honest participants are never blamed and that only deviations from formally defined security goals are accounted for.

Privacy

Another feature of great importance for the correct operation of an (electronic) voting system is the concept of privacy. According to it, *the actual choice of a specific voter should not be revealed to anybody even if the voter wishes to*. While, this property has seemingly little to do with correctness, it resembles an adversary changing the contents of a ballot, since a voter without privacy can be vulnerable to influence from external agents. A related yet dis-

tinct notion is that of voter *anonymity* or *unlinkability*, which might reveal the voter choice, but without any link to the voter identity. In physical elections, secrecy is enforced using the sealed envelope and the ballot booth, while anonymity is used during ballot counting, where the identities of the voters cannot be achieved since the ballot carry no such information and the ballot box is shuffled before opening them. Voter privacy is a complex notion and there are many different variations to it.

Ballot secrecy. The contents of a particular vote (i.e. one that is tied to an identified voter) should be protected from other voters (single or in groups) and tallies. This is usually defined in a formal manner by running two executions of the voting system with the same result, and trying to associate each execution with a particular voting behaviour by (a subset of) the honest voters (Cohen & Fischer, 1985). An important remark here is that ballot secrecy cannot be totally achieved since the result of the computation might leak some information on its own, as is the case with a unanimous vote, where we know for sure how everyone voted. However, even if a vote is not unanimous, when a party receives e.g. 40% of the total votes, then we know that for a random voter, there is a 40% probability to have voted for the particular party. Even if only a single voter begs to differ, she is entitled to it and must be protected. Generalizing the above observation, we must note that any corrupted subset of voters can collaborate and deduce the sub tally of all the honest ones. However no more should be deducible from the outcome. In the simplest case, privacy means that no further information is deducible from the votes and the tally alone, against corrupted tallies conspiring with other corrupted voters. These adversaries are assumed passive, which means that they adhere to the voting protocol, but perform extra computations with the data they obtain.

Receipt Freeness. The next level of privacy defends against a corrupted voter that wants to sell her vote. The vote buyer is considered passive, waiting to receive evidence of the vote selling to check his 'purchase'. In physical elections this is enforced by the screen in the polling station in a simple manner: even if the voter wants to sell her vote, she cannot prove that she in fact did. In electronic voting things get more complicated, because the evidence that are generated for verifiability purposes can serve as a proof that the voter voted in a certain way. As a result, they must be easy to fake in a manner that cannot distinguish them from real ones and may not be transferable (Benaloh & Tuinstra, 1994).

Coercion Resistance. This property defends against the most offensive attack scenario. The attacker in question is standing over the voter's shoulder and is dictating her behavior. He can make her vote randomly or abstain from voting. He can even impersonate her by controlling her credentials. This quite powerful attack is one of the greatest obstacles to the introduction of remote electronic voting. There is however a framework (Juels et al., 2005) that despite being counter intuitive, allows electronic voting systems to deal with coercion attempts. It is based on the simple observation, that a coercer has no incentive to carry out an attack, if he cannot tell if it was successful or not. A voting system can enable such a behaviour from voters, if it can allow them to vote multiple times, by using different credentials each time. These credentials must be indistinguishable. One of them is registered as authentic before the election begins through an untappable channel (i.e. in person). However this is not a problem as it can be used in many elections. During the elections, the voter is assumed to have the capability to generate her own credentials. Each ballot is accompanied with one of them. The vote that corresponds to the registered credential is the only one counted, while all others are discarded during tallying. It is

assumed that the real vote is used during a moment of privacy, were the coercer is not present, while the fake ones are used in the presence of an adversary.The difficulty in this scenario is to make the count verifiable, to everybody except the coercer, who must not tell whether the voter actually obeyed his order and voted or abstained. As a result, coercion resistance assumes voter anonymity.

This framework has been applied to the *Civitas* system (Clarkson et al., 2008). An alternative implementation appears in (Grontas et al., 2017). A weaker and simpler version of coercion resistance can also be found in Helios (Adida, 2008), where the voter casts many ballots with the same credential but only the last vote counts.

Everlasting privacy. Another aspect of ballot secrecy that must be considered is its duration. In many implementations that use cryptography, ballot secrecy is conditional on the premise that it depends on the solution of computational problems that are currently assumed infeasible. However, since there is no formal proof, some years into the future this might not be the case, as the advent of science and technology might render such computational assumptions obsolete. This possibility combined with the requirement that all auditing data are publicly available might allow the recovery of one's vote. In a future oppressive regime, this might have dire consequences for a present voter or her relatives. Consequently, privacy must be unconditional or to use another term, everlasting (Moran & Naor, 2006).

The conflict between verifiability and privacy is worth commenting. A verifiable voting system that lacks privacy is easy to obtain – it is the electronic equivalent of a snapshot during show of hands. However, it is not trustworthy, since the voters cannot be trusted to express their real preferences if they know that they will be recorded and potentially be used against them. On the other hand, a private voting system that is not verifiable is useless, since nobody will now if their vote really counted, so nobody will bother to use it. This conflict extends to other voting properties as well.

Authentication and authorisation

While voting may be a general way to reach a decision, not everybody should be able to participate. Various jurisdictions impose rules such as age or citizenship limits, which distinguish the voters with the right to cast a ballot. Thus, there must be a way for the eligible voters to stand out. Authentication serves this purpose. However, it requires some form of identification and this might contrast the anonymity requirement. There are cryptographic solutions that allow a voter to prove that she meets the voting criteria (i.e. prove that she is an adult or a citizen), while withholding her identity. Alternatively, an anonymity set can be created, where only eligible voters are contained, and the voter must prove that she belongs to this set without explicitly disclosing her identity. This can be related to the real-world fact that once the partial results of a polling station are public – the aggregate hides the particular voter choice.

Enfranchisement

All participants eligible to vote should be encouraged to participate. This is easier said than done. For example, in order to be authenticated, voters should use some form of credentials. These exclude the people that cannot use these credentials for some reason. In addi-

tion, voters should not be intimidated by the voting system and be discouraged to participate. This is particularly true for electronic voting, since technologically illiterate people might find it hard to use the system. Of course, possible solutions must respect the previous requirements. Assistance in voting for example might solve the problem, if ballot secrecy is maintained. Enfranchisement implies that voters trust the system to express their intent. Complex voting systems, such as the ones that are based in cryptography that we described in this paper, might succeed in implementing some or most of the requirements but might-fail to convince the voters, because they might be considered too complex to understand.

Fairness

A voting system must tally the votes without making public anything, but the final results. This is particularly important if the votes can be counted, while the voting period is still going on. Since, an electronic voting system must make available data for verifiability purposes, it must be impossible that they lead to partial results. Another aspect of fairness is that elections are a one-time event. If they are interrupted or cancelled there is not guarantee that the voters will not be affected and exhibit a different behaviour in the second run.

Availability and efficiency

A voting system should always be available to receive input from the participants and it must output the result of the computation in a reasonable time. Availability is crucial since the repetition of the vote casting phase, sheds doubts on fairness. Availability in the presence of an active and persistent adversary is often called robustness. The adversary might pose as a voter or as the authority or both. In addition, lack of availability hinders enfranchisement, since a voting system that has 'ups and downs' in its operation is perceived as untrustworthy. It is very important to point out that a voting system must be robust in its complete lifecycle and not only on the vote casting stage. For example, a verifiable voting system must have a detailed process to deal with 'alleged' verification problems, or else verifiability can be used against the system. Efficiency is one of the reasons put forth by electronic voting proponents, especially when the voting population is extremely large. It refers to the resources used by the voting system in all its workflow. Such resources are referred to as cost and might be time, money, requirements on infrastructure and people participation etc.

Cryptographic voting systems

Electronic voting systems can achieve software independence by utilizing ideas and concrete primitives from the field of cryptography. A cryptographic election system is modelled after voting with show of hands, where a snapshot of the hands is maintained for tallying. Cryptography is used in order to hide the choices of the voters but must importantly to obtain *trust* in the correctness of snapshot with minimal reliance to third parties. In other words the role of cryptography in voting is to replace the paper trail with mathematics (Adida, 2006). In what follows we shall review the necessary cryptographic components that make up a voting system in order to gain a better understanding on how they work.

Bulletin board and consensus

We begin the overview of the cryptographic voting paradigm by describing the dominant architecture to store the snapshots of the votes first proposed in (Cohen & Fischer, 1985). It involves a publicly available repository where every interested party posts all election related data (cryptographic keys, candidate slate, registration information, votes, auditing information etc.). This repository is called the *Bulletin Board (BB)* and is defined as a broadcast channel with memory. It supports only the insertion of new data (append-only) without allowing deletions or modifications. Data is retained for verifiability purposes. While all proposals in cryptographic voting utilize the Bulletin Board, there is no acceptable implementation. Many real-world systems utilize a shared database and a corresponding website; however, this requires trust in the database owner. In order to distribute the trust into peers, one must solve a *consensus problem* between these agents so that everybody agrees that a particular vote, for instance, must be appended to the election log. This problem is difficult to achieve in the presence of Byzantine (malicious players) and a definitive solution eludes computer scientists since its introduction in (Lamport et al., 1982) despite many attempts. To see why this is such a difficult problem, consider the trivial case illustrated in Figure ***1***. A corrupted party can stop two honest parties from reaching consensus, where the BB consists only of three players – one of which is malicious. One of them has received a ballot and all must collectively decide whether to accept (1) it or not (0) by exchanging opinions. In the first case, the ballot receiver is honest and proposes to accept the ballot. One of the participants is malicious and misreports the opinion of the receiver. As a result, the other honest player receives conflicting opinions. In the second case, the ballot receiver is malicious, and the initial messages are conflicting, leading to a dead end again. This impossibility result can be generalised in the case of n players, where the existence of more thantwo thirds of honest players is a necessary condition for consensus.

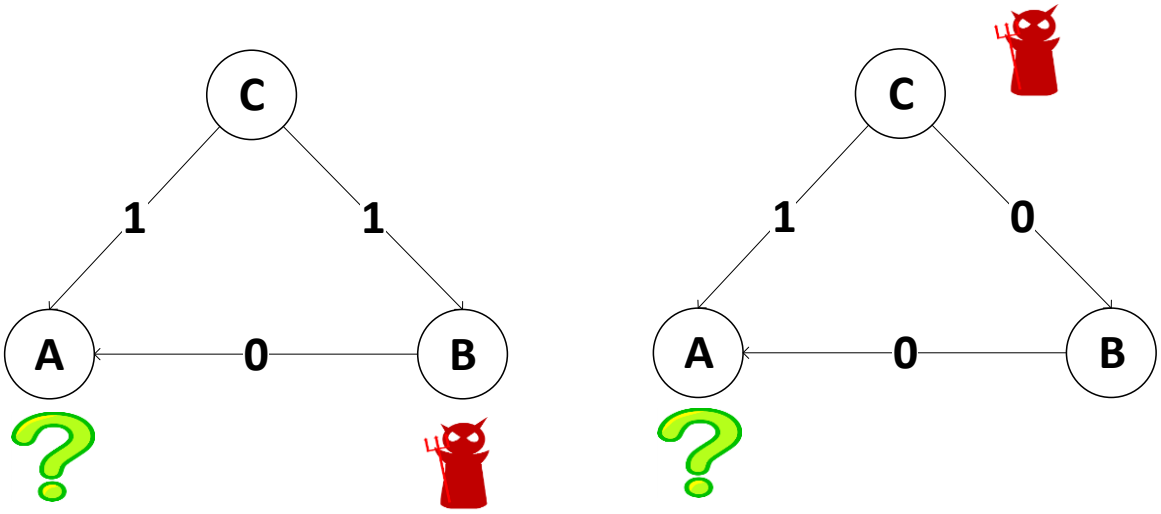


Figure 1. *A corrupted party can stop two honest parties from reaching consensus*

Many distributed algorithms have been proposed in order to implement a Bulletin Board applicable to electronic voting scenarios (Culnane & Schneider, 2014; Kiayias et al., 2018). Recently the blockchain concept has emerged as a trust less decentralized BB implementa-

tion. It has formally been proved that it solves the consensus problem, extended to accept a dynamic set of anonymous participants, assuming an honest majority (Garay et al., 2015) along with some cryptographic assumptions. The applicability of the blockchain as a replacement for the BB will be examined in the next section.

Semantically secure cryptosystems

In a cryptographic voting system, all voters post their ballots to the Bulletin Board where they are subsequently verified, counted and audited. Since the BB is publicly accessible, the votes must be hidden for ballot secrecy. As a result, some form of encryption must be used. A public key cryptosystem is preferred since it makes the management of cryptographic keys easier. Typically, the election authority (EA) posts its public key to the Bulletin Board and the voters receive it and encrypt the ballots. Since there are only a few preferences that can be selected in a typical voting system, the encryption scheme used must be *semantically secure,* i.e. it must not allow a potential adversary to discover the vote by brute forcing encryptions of all possible choices using the publicly available key of the EA and matching them with the encrypted ballot. This is usually achieved by embedding randomness in the encrypted message, so that there are many possible encryptions of a single choice (Goldwasser & Micali, 1984). Usually, the ElGamal cryptosystem (Elgamal, 1985) is used, which operates in a cyclic group G of prime order q generated by g. Encryption uses the public key h and decryption uses the private key $x \in Z_q$:

$$\mathrm{Enc_h(m, r)} = (\mathrm{g^r}, \mathrm{m} \cdot \mathrm{h^r}) \; and \; \mathrm{Dec}_x(a, b) = \frac{b}{a^{-x}}$$

A variation with useful properties is the exponential ElGamal cryptosystem, where the decryption involves solving a discrete logarithm problem. As a result, this version is useful only when the message can be encoded as a small number.

$$\mathrm{Enc_h(m, r)} = (\mathrm{g^r}, \mathrm{g^m} \cdot \mathrm{h^r}) \text{ and } \mathrm{Dec}_x(a, b) = \log_{\mathrm{g}} \frac{b}{a^{-x}}$$

Note that in both cases, possession of the randomness $r \in Z_q$used in the encryption must be protected since it can serve as a receipt, on how the voter voted.

Secret sharing. In a public key cryptosystem, the owner of the secret key has the power to decrypt. This power is critical in a voting, since encrypting with the EAs public key means that the EA can decrypt each individual vote and learn everyone's preferences. In order to avoid situations like these we need to split the secret key into pieces and give each piece to a shareholder. There are schemes that can provide this functionality, with the most prominent one being the one introduced in (Shamir, 1979). It also allows flexibility, as it does not require all shares of the key to be present in order to decrypt. A threshold can be defined, where the secret (key) be retrieved, only if more than a certain number of shares are combined.

Homomorphic encryption. Most cryptosystems have a useful property: they allow the combination of ciphertexts in such a way that a mathematical operation is transferred to the corresponding plaintexts. For instance, if one multiplies two ElGamal ciphertexts, the result

corresponds to the product of the original plaintexts. If one multiplies two exponential ElGamal ciphertexts, the result corresponds to the sum of the original plaintexts.

$$\begin{aligned}\mathrm{Enc_h}(m_1, r_1) \cdot \mathrm{Enc_h}(m_2, r_2) &= (g^{r_1}, g^{m_1} \cdot h^{r_1}) \cdot (g^{r_2}, g^{m_2} \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, g^{m_1+m_2} \cdot h^{r_1+r_2}) = \mathrm{Enc_h}(m_1 + m_2, r_1 + r_2)\end{aligned}$$

This property is very useful in cryptographic voting, as it provides a method for the EA to compute the election result without decrypting the ballots. For instance, consider the case where the ElGamal cryptosystem is used by voters to cast their ballots in a referendum. Their available choices are 1 for yes and -1 for no. The EA can multiply all the encrypted ballots cast and obtain a ciphertext that corresponds to the result of the election (number of *YES* votes – number of *NO* votes). If the elections supports more than two options then a different encoding might be used – e.g. (Baudron et al., 2001). The product of the ciphertext contains the election result and can now safely be decrypted. Notice that in this scheme, the election authority must be trusted only to decrypt the result and not the individual votes. Such trust can be lifted by applying a secret sharing scheme to the decryption key, where the shares are typically distributed to mutually distrusting authorities as is the case in the real world. This voting scheme has been originally proposed in (Cramer et al., 1997) and applied to the Helios election system (Adida, 2008). Its drawback is that the only operations that can be *efficiently* performed on encrypted data are currently only multiplication and addition. This means that it is not easy to incorporate complex voting rules on encrypted ballots.

Zero knowledge proofs of knowledge

The homomorphic method to compute results as presented in the previous section has a very important drawback. It allows a cheating voter to post a *mega-ballot* that hides more than a single vote. For instance, if a candidate is encoded as the value 1 then the voter can encrypt and post the value 100 instead, thus adding 100 votes for her preference to the final tally. To avoid this attack scenario each ballot must checked for its validity. This seems impossible, since the content of the ballot must be secret. However there is a cryptographic primitive that can realize it, namely *Zero Knowledge Proofs Of Knowledge* (ZKPoK) proposed in (Goldwasser et al., 1985).

A ZKPoK is a protocol that can prove the correctness of a computation without disclosing any information about it. It consists of two parties, namely the *prover* and the *verifier*. The prover wants to convince the verifier about the knowledge of a secret (often referred to as a witness), without disclosing (any part) of it. The concept seems counterintuitive, but a simple example can help illustrate how it might be achieved (Goldreich, 2010):

The prover holds two identical balls of different colour. The verifier is colour blind and wants to be convinced that the balls have indeed a different colour. This can happen using the following interactive sequence of actions:

- The prover gives the balls to the verifier (commit phase).
- The verifier hides the balls behind her back, one ball per hand.

- Randomly (with probability $\frac{1}{2}$), the verifier switches hands for each ball, in a manner invisible to the prover (i.e. behind her back).
- The verifier reveals the balls (challenge phase).
- The prover must guess whether the verifier switched hands or not (response phase).
- Since the prover might cheat, i.e. present two balls of the same colour and succeed in guessing whether the verifier switched hands or not, a single guess does not provide many guarantees. However, the experiment can be repeated n times to decrease the cheating probability to 2^{-n}.
- Verifier is convinced, that the boxes have different colour, without ever knowing what the actual colours are.

ZKPoKscan be made non interactive with the Fiat-Shamir heuristic (Fiat & Shamir, 1986). In voting systems, they can augment the homomorphic scheme by demonstrating that voters have correctly encoded their preferred candidates, or that the election authority has correctly decrypted the election result. In general, they can be used to show that the all participants in a voting system have correctly followed the protocol without disclosing their private inputs (preferences), thus enabling software independence without sacrificing secrecy. Moreover, similar ideas based on the commit-challenge-response concept – i.e. Benaloh challenges (Benaloh, 2006) – can be used to convince the voters that their choice has been cast as intended by a DRE voting machine. After the voter casts the vote, she is presented with two choices. She can either audit the vote or record it. The voting device must commit to the ballot representation without knowing if it is going to be audited or not. As a result, it is forced to follow the protocol and the voter can be assured that her ballot is cast as intended.

Digital signatures

Digital signatures, proposed in (Diffie & Hellman, 1976), are one of the most successful public key cryptographic primitives. A user submits a message to a signer, who applies a function of his secret signing key and generates a signature that can be verified by everybody that possesses the corresponding public verification key. They allow message integrity, authenticity and non-repudiation in a publicly verifiable manner, certifying that a ballot has not been altered during its transmission to the Bulletin Board for example. However, the public key might give away the identity of the voter, so they cannot be used as is. In fact, many variations have been proposed that can combine their properties with anonymity.

Blind Signatures. Blind signatures were presented in (Chaum, 1982) as a means of realizing anonymous credentials in electronic payments and other applications. The concept is very simple and can be presented with a simple real-world analogy. Consider a consumer who wants to pay a merchant some money issued by a bank to purchase a good. This can be done without the bank learning the identity of the consumer or what exactly was purchased using the following real-world protocol:

- Blinding: The consumer puts a blank sheet of paper into an envelope. On it, there is a carbon sheet attached.
- He sends the envelope into the bank requesting the amount of money that the good costs.

- Signing: The bank validates that the consumer has the money into his bank account and signs the envelope.
- Unblinding: The consumer validates the signature and empties the envelope. Due to the properties of the carbon paper the signature is transferred to the enclosed sheet.
- The merchant is paid with the signed sheet of paper.
- She presents it to the bank.
- Verification: The bank validates its own signature and pays the merchant the appropriate amount.
- The transaction is complete. Since the bank's signature is transferred to many identical blank sheets, the bank cannot trace who purchased which items.

This usage scenario can be easily applied to voting, resulting in a system, proposed in (Fujioka et al., 1992) that bears similarities to the operation of a polling station, if the registrars were in a different physical location than the ballot box.

- The voter submits a blinded version of the ballot along with identity information.
- The election authority verifies the identity of the voter and checks if she is eligible. If everything is OK, then it signs the blinded ballot.
- The voter verifies the signature of the election authority, unblinds the ballot, and posts it to the BB *anonymously*.
- The EA collects all the signed ballots, the signatures and computes the results.

Since its original publication, many variations of the blind signature concept have been presented. On variation of particular interest is *Conditional Blind Signatures* (CBS) proposed in (Zacharakis et al., 2017). This variation encapsulates the check made by the election authority in the election scheme described above, into a standalone cryptographic primitive. A blinded message is sent to an authority, which in turn tests a predicate and signs the message. The resulting signature is valid if and only if the predicate is true. However, this scheme has a serious drawback – the authority can cheat and provide an arbitrary signature disregarding the value of the predicate. In order to make the signature auditable a variation of CBS called PACBS (Publicly Auditable Conditional Blind Signatures) has been proposed in (Grontas et al., 2019) where the authority embeds the predicate inside the signature and attaches zero knowledge proofs that enable everybody to check its actions.

Mixnets

A *mixnet* (Chaum, 1981) is a general anonymity primitive that is constructed using cryptographic schemes like the ones we previously described. As such, it has been used in systems that provide anonymous services i.e. browsing, auctions and of course elections. A mixnet anonymises a set of input items, by garbling and shuffling them. It consists of entities called *mix servers*, where each one receives a set of inputs and breaks the link between input and output, by changing its exit position and form. More formally, an item entering as v_i will exit as $v'_{\pi(i)}$, where π is a random permutation. Since an itemtraveling through the mix net always changes exit position and form it cannot be tracked. The transformation of an item is achieved using cryptographic operations: Two variations have been proposed: *the decryption mixnet* (Chaum, 1981), where the sender encrypts the mixnet input using multiple lay-

ers of public keys and each mix server partially decrypts, and *the re-encryption mixnet* (Park et al., 1993) which utilizes the homomorphic properties of the underlying cryptosystem.

In the context of voting, the anonymised items using the mixnet are the votes and the mix servers communicate through the Bulletin Board. Traditionally the mix servers operate in sequence. However, on a practical level they can operate in parallel, where the votes are split into distinct groups. Each group is assigned to one mix server that performs the shuffling and transformation. After the operation, that group is assigned to next mix server. In the last phase the first group will be processed by the last server. More importantly, in electronic voting the actions of all mix servers must be universally verifiable (Kilian & Sako, 1995; Furukawa & Sako, 2001; Neff, 2004). As a result, they provide zero knowledge proofs that they processed their items successfully. This approach, however, is computationally expensive so there are other approaches that sacrifice accuracy for speed and robustness (Jakobsson et al., 2002). The research on mix nets is very rich and they can be considered a mature technology.

Blockchain voting

The popularity of Bitcoin (Nakamoto, 2008) and other cryptocurrencies has provided a novel approach to the implementation of Bulletin Boards for use in electronic voting.

Bitcoin basics. These electronic coins solve the double spending problem inherent in all forms of e-cash without resorting to trusted authorities like blind signature-based schemes. In particular, they do away with the token-based metaphor for money, replacing it with signed transactions. For instance, in Bitcoin, if one wants to pay a certain amount, say 5 bitcoins, it creates a transaction transferring the corresponding amount to the payee, digitally signs it and broadcasts it to a network of cryptocurrency users. For the transaction to be valid the payer must point to transactions, where he is on the receiving end, whose amounts total to the 5 bitcoins currently being spent. For this to be possible all transactions ever spent must be recorded. In Bitcoin, this takes place on a distributed ledger data structure that groups transactions in blocks. They are ordered in a precedence relation using links (cryptographic fingerprints) to previous blocks. This makes sure that no transaction can spend 'coins' that were never received and that no one can modify accepted transactions. This distributed ledger data structure, is called a *blockchain*. In our example, the payer will point to the past transactions residing in the blockchain, in which he received the 5 bitcoins he wants to transfer.

The novel aspect of the blockchain is that there is no central authority to maintain the transaction ledger. Every participant can in principle maintain it. However, this creates the problem of conflicting versions as each participant might not receive some transactions or receive them in different order. For the blockchain to be usable as a transaction ledger, all participants must reach consensus on the exact order and contents of all transactions. As we saw the consensus problem is one of the most difficult in computer science. The creators of Bitcoin came up with a novel solution to overcome it. In predetermined periods of time a random participant is selected with the aim to propose which transactions are to be included to the ledger. In order to thwart a sybil attack – one where an adversary controls many participants to increase its chance of being selected – each participant must solve a cryptographic puzzle that requires computational power. This mechanism is called *proof of work* and the participants that execute it are called *miners*. Mining is combined with a simple con-

flict resolution rule: In the case multiple valid blockchains are presented the one with the more invested computational power wins. The novelty of the scheme ties the proof of work mechanism to the validation of transactions. Moreover, it rewards the participants with coins, incentivizing them to contribute.

Many variations have been proposed for the architecture of a distributed ledger. If there are no restrictions on the participants, as is the case in Bitcoin the blockchain is called *permissionless*, otherwise it is called *permissioned*. In addition, in some cryptocurrencies like Ethereum (Wood, 2014) the participants can execute arbitrary code apart from the validation of the transactions. This code is called *smart contracts*, and the blockchain is used to maintain its state – creating in effect a global computer.

Blockchains in electronic voting. The idea behind blockchain voting comes from the conceptual similarity of the blockchain with the Bulletin Board. Indeed, in both cases, transactions need to be recorded in an append-only manner and the participants *much reach consensus on which transactions are valid and which are not*. In the case of cryptocurrencies these transactions represent the transfer of value, while in the case of elections these transactions represent the recording of preference.

An electronic voting Bulletin Board implemented as a blockchain, eliminates the need for a trusted third party to maintain the data needed to tally and audit an election. It can also, in theory, lead to truly decentralized elections where the voters themselves conduct the elections without any trusted parties to record, count and verify the votes. All parties accept the contents of the Bulletin Board where the auditing will occur, as they accept the recorded transactions in the Bitcoin blockchain.

As a result, many have proposed to replace the Bulletin Board of a voting system with a distributed ledger technology like the blockchain, resulting in several proposalboth purely academic – i.e. (McCorry et al., 2017; Panja, 2018; Yu et al., 2018) as well as in the industry – e.g. the *VOLT Project*, *Democracy.Earth*, *FollowMyVote* and more. All of them are built on the same scenario (Nasser et al., 2018): each candidate is represented by a Bitcoin address (account). When a voter wants to vote for a specific candidate, she sends a fixed small payment to the address of the candidate. This transaction is recorded on the blockchain. Consequently, the voters themselves need to be represented using addresses that function as pseudonyms. Both a permissioned and a permissionless blockchain could be used. When the election ends, everybody can check the blockchain and sum the amount of coins received by each candidate address and declare the winner.

This scenario is very similar to the snapshot of a show of hands that we repeatedly used in explaining the properties of voting systems. Since all the votes are represented as transactions in the publicly readable blockchain everybody can audit them and verify the tally. If a public permissionless blockchain is used, the voting system is integrated into the everyday operation of the cryptocurrency, and as a result the guarantees are even higher. However, there are many details that need to be considered.

Firstly, in order to enable eligibility verifiability, there must be a protocol that determines the mapping between voter addresses and their real identities. If it is executed by a registration authority, then a trusted third party is introduced to the voting system and the use of the blockchain resembles the permissioned case. If the elections are conducted in a small scale, then it is reasonable to assume that the voters can jointly agree to their eligibility, using real world information. On the other hand, in large scale elections, voters need to simply

'vote and go' and cannot be expected to run such protocols on their own. In fact, some aspects of real world voting especially at the national scale are claimed to be inherently uncentralizable as argued in (Heiberg, 2018). In our case, eligibility verifiability requires an identity provider that must bind real world identities to voting credentials granted only to the ones with the right to vote. This means that a registration authority is required, but it must be prevented from associating voters with bitcoin addresses. This can be done using cryptographic anonymity principles such as blind signatures and mixnets. While this binding can be made verifiable with tools such as PACBS, the identities are still maintained in a centralized database operated by a nation state – a trusted third party.

A second drawback is that the basic bitcoin voting scheme satisfies neither notion of ballot secrecy. The use of pseudonyms provides minimum protection, but the real identities might leak or they might be deanonymized by using advanced analysis techniques (Meiklejohn et al., 2013). A solution employs techniques used in homomorphic voting systems. The voters encrypt their choice of candidate and instead of sending transactions to each candidate they send transactions to a single address owned by a tallier. There are many ways to embed the encrypted ballot inside the transaction, depending on the type of blockchain used: In the case of Bitcoin, encryption will be done outside the system and the hidden ballot along with the zero knowledge proofs of validity will have to be stored on a separate data source and linked with the transaction with a construct like the `OP_RETURN` statement. This approach has the downside that the external data source becomes a trusted third party that has control over the actual data. Alternatively the voters can self-tally the elections using the recorded data. For this a blockchain that supports smart contracts must be used. This is implemented in the *Open Vote Network* (McCorry et al., 2017).

Receipt freeness and coercion resistance are also problematic in this basic scheme, as an attacker can easily ascertain whether their target followed their instructions by simply reviewing the blockchain. In fact, the forced abstention attack is trivial to perform. In order to solve these problems, more central authorities are required. On the voting end, there should be an authority that provides anonymous credentials (addresses) to the voters, in order to cheat the adversary from finding out if their target really took part in the election and what was their choice. In addition, another central authority must operate on the tallying side and filter out the coerced votes based on the fake credentials. Finally, the public availability of collection data makes everlasting privacy even harder to achieve.

Except for secrecy, the basic blockchain voting scheme does not support fairness, as anybody can calculate intermediate results by monitoring the transactions broadcast on the blockchain. This can affect the choice of late come voters (Nasser et al., 2018). This is easy to understand by recalling the show of hands analogy, where an initially generated momentum on a voting option is self-reinforced due to the transparency of the process.

The proposed blockchain solutions can be examined in many aspects, for instance if they are truly non-centralizable or whether they support large scale elections. An analysis of (Dricot & Pereira, 2018) finds that only the OpenVote network (McCorry et al., 2017), is a functionally decentralised platform. However, it has scaling problems, as it is meant to be used only for small scale or boardroom elections (for a maximum number of 50 voters as the authors themselves claim). In general, all public blockchains suffer from efficiency issues, both in the number of transactions that are cleared per second as well as in the wait time (Heiberg,

2018) for a transaction to be confirmed. These make them difficult for use in large scale elections.

It must also be stressed that, the decentralization arguments only hold for the application layer, i.e. the voting protocol itself. While the network layer – the blockchain – is considered decentralized, a closer look reveals that there exists concentration on mining power (Heiberg, 2018). For example, a recent work by (Gencer, 2018) finds that 90% of mining power is in the hands of 16 miners in Bitcoin and 11 miners in Ethereum. This is a troubling result, because it implies that the voting protocols are designed with on top of a leaky abstraction.

Finally, blockchain voting worsens a major problem faced by all electronic voting schemes especially cryptographic ones. Enfranchisement requires that the voter understands the process she participates in. This is difficult to do when the system is built on top of complex mathematical concepts that cannot be easily explained. Some claim that approaches like code voting (Chaum, 2001) aim to hide abstract such difficulties away from the voter, however we doubt their effectiveness. This situation is made worse by the probabilistic and the incentive-based nature of the security of many of the schemes we dealt with. This applies especially to blockchain voting. While their scientific analysis is sound, the average voter might not be confident with less than perfect solutions. Opponents of such system might even take advantage of such misconceptions in order to cast doubt on these voting solutions.

Conclusion

To sum up, in this paper we examined whether the blockchain is the future of voting as claimed by many authors of e-voting schemes. We found that this is still far from being the truth. Firstly, this is because voting itself is an inherently difficult problem. The accepted (physical) solutions that are currently being used do not solely rely on technical procedures, to settle the conflicting security requirements, but are built on layers of trust in institutions. Furthermore, there is no single technical solution of choice for e-voting systems. Agreement is reached only on that an electronic voting system should be software independent and E2E verifiable. Most experts propose using a paper trail for verifiability since the cryptographic solutions are still immature, but there is again no system that is totally accepted. Voting on the blockchain solves only a single part of the voting problem – how to reach consensus on the transcript of the protocol. However, this is only the tip of the iceberg. In order to be able to convince the loser candidates, as well as the electoral body and observers, a voting protocol must satisfy strict security requirements, for which the blockchain does little to help so far.

Electronic voting is still in its infancy and should be approached with caution as it can help attackers cheat at a large scale with ease. However, as all human activities are going to be conducted electronically in the long run, it is inevitable that voting follows suit. Cryptography and other techniques will evolve to support the security requirements. At the same time the public should gradually get accustomed to the new processes by participating in small scale elections where the stakes are not as high as in national governmental elections. This means that the introduction of electronic voting should be bottom up and gradual, providing security guarantees that are theoretically rigorous, yet easily conceivable by the average voter.

References

Adida, B. (2006). *Advances in cryptographic voting systems*. (Thesis (Ph. D.)). Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science

Adida, B. (2008). Helios: web-based open-audit voting. In *Proceedings of the 17th conference on Security symposium* (pp. 335-348). USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id=1496711.1496734

Adida, B. (2019). Convincing the loser: Securing elections against modern threats. Retrieved from https://www.youtube.com/watch?v=dy0_8A9U8Rs

Alvarez, R. M., & Hall, T. E. (2010). *Electronic elections: The perils and promises of digital democracy*. Princeton University Press.

Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., & Poupard, G. (2001). Practical multi-candidate election system. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing* (pp. 274-283). New York, NY, USA: ACM. https://doi.org/10.1145/383962.384044

Benaloh, J. (2006). Simple Verifiable Elections. In *EVT'06*.

Benaloh, J. (2008). Administrative and public verifiability: Can we have both? In *Proceedings of the Conference on Electronic Voting Technology* (p. 5:1-5:10). USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id=1496739.1496744

Benaloh, J., & Tuinstra, D. (1994). Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing - STOC '94* (pp. 544-553). New York, New York, USA: ACM Press. https://doi.org/10.1145/195058.195407

Chaum, D. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, *24* (2), 84-88.

Chaum, D. (1982). Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, & A. T. Sherman (Eds.), Boston, MA. https://doi.org/10.1007/978-1-4757-0602-4_18

Chaum, D. (2001). SureVote: Technical Overview. In *Workshop on Trustworthy Elections (WOTE '01)*.

Chaum, D. (2004). Secret-ballot receipts: True voter-verifiable elections. *Security & Privacy, IEEE*, *2*(1), 38-47.

Clarke, D., & Martens, T. (2016). E-Voting in Estonia. In *Real-World Electronic Voting: Design, Analysis and Deployment*. https://doi.org/10.1201/9781315371290

Clarkson, M. R., Chong, S., & Myers, A. C. (2008). Civitas: Toward a secure voting system. In *IEEE Security and Privacy Symposium*. Retrieved from http://dblp.uni-trier.de/db/conf/sp/sp2008.html#ClarksonCM08

Cohen, J. D., & Fischer, M. J. (1985). A robust and verifiable cryptographically secure election scheme (Extended Abstract). In *FOCS* (pp. 372-382).

Cortier, V., Galindo, D., Küsters, R., Müller, J., & Truderung, T. (2016). SoK: Verifiability notions for e-voting protocols. In *IEEE Security and Privacy Symposium* (pp. 779-798).

Cramer, R., Gennaro, R., & Schoenmakers, B. (1997). A secure and optimally efficient multi-

authority election scheme. *Transactions on Emerging Telecommunications Technologies*, 103-118.

Culnane, C., & Schneider, S. (2014). A peered bulletin board for robust use in verifiable voting systems. In *2014 IEEE 27th Computer Security Foundations Symposium* (pp. 169-183). IEEE. https://doi.org/10.1109/CSF.2014.20

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644-654. https://doi.org/10.1109/TIT.1976.1055638

Dricot, L., & Pereira, O. (2018). SoK: Uncentralisable ledgers and their impact on voting systems. Retrieved from http://arxiv.org/abs/1801.08064

Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, *31*(4), 469–472. https://doi.org/10.1109/TIT.1985.1057074

Fiat, A., & Shamir, A. (1986). *How to prove yourself: Practical solutions to identification and signature problems*. In A. M. Odlyzko (Ed.) (Vol. 263, pp. 186-194). Berlin, Heidelberg. https://doi.org/10.1007/3-540-47721-7_12

Fujioka, A., Okamoto, T., & Ohta, K. A. (1992). *Practical secret voting scheme for large scale elections*. (J. Seberry & Y. Zheng, Eds.), 718 Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings § (1992). Springer. https://doi.org/10.1007/3-540-57220-1_66

Furukawa, J., & Sako, K. (2001). An efficient scheme for proving a shuffle. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology* (pp. 368-387). London, UK: Springer-Verlag.

Garay, J., Kiayias, A., & Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-662-46803-6 10

Gencer, A. E. (2018). Decentralization in bitcoin and Ethereum networks. *CoRR*, *abs/1801.0*. Retrieved from http://arxiv.org/abs/1801.03998

Goldreich, O. (2010). *Foundations of cryptography*. https://doi.org/10.1017/cbo9780511721656

Goldwasser, S., & Micali, S. (1984). Probabilistic encryption. *Journal of Computer and System Sciences*. https://doi.org/10.1016/0022-0000(84)90070-9

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, *18* (1), 186-208. https://doi.org/10.1137/0218012

Grontas, P., Pagourtzis, A., & Zacharakis, A. (2017). Coercion resistance in a practical secret voting scheme for large scale elections. In *Proceedings - 14th International Symposium on Pervasive Systems, Algorithms and Networks, I-SPAN 2017, 11th International Conference on Frontier of Computer Science and Technology, FCST 2017 and 3rd International Symposium of Creative Computing, ISCC 2017* (Vol. 2017–Novem, pp. 514-519). IEEE. https://doi.org/10.1109/ISPAN-FCST-ISCC.2017.79

Grontas, P., Pagourtzis, A., Zacharakis, A., & Zhang, B. (2019). Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Sala, & P. Massimiliano (Eds.), *Financial cryptography and data security (FC 2018). Lecture Notes in Computer Science, vol 10958* (pp. 210-231). Springer. https://doi.org/10.1007/978-3-662-58820-8_15

Halderman, J. A. (2012). Securing digital democracy. Retrieved from https://www.coursera.org/course/digitaldemocracy

Halderman, J. A. (2016). Practical attacks on real-world e-voting. In *Real-World electronic voting: Design, analysis and deployment*. https://doi.org/10.1201/9781315371290

Heiberg, S. (2018). On trade-offs of applying block chains for electronic voting bulletin boards. *IACR Cryptology EPrint Archive*, *2018*, 685. Retrieved from https://eprint.iacr.org/2018/685

Jakobsson, M., Juels, A., & Rivest, R. L. (2002). Making mix nets robust for electronic voting by randomized partial checking. In *In USENIX Security Symposium* (pp. 339-353).

Juels, A., Catalano, D., & Jakobsson, M. (2005). Coercion-resistant electronic elections. In V. Atluri, S. D. C. di Vimercati, & R. Dingledine (Eds.), *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 6000 LNCS, pp. 37-63). ACM. https://doi.org/10.1007/978-3-642-12980-3_2

Kiayias, A., Kuldmaa, A., Lipmaa, H., Siim, J., & Zacharias, T. (2018). On the security properties of e-voting bulletin boards (pp. 505-523). https://doi.org/10.1007/978-3-319-98113-0_27

Kiayias, A., & Yung, M. (2002). Self-tallying elections and perfect ballot secrecy. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/3-540-45664-3_10

Kiayias, A., Zacharias, T., & Zhang, B. (2015). End-to-end verifiable elections in the standard model. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 9057, pp. 468–498). https://doi.org/10.1007/978-3-662-46803-6_16

Kilian, J., & Sako, K. (1995). Receipt-Free {MIX}-Type voting scheme - A practical solution to the implementation of a voting booth. In *Proceedings of {EUROCRYPT} 1995*. Springer-Verlag.

Küsters, R., Truderung, T., & Vogt, A. (2010). Accountability: definition and relationship to verifiability. *Proceedings of the 17th ACM conference on Computer and communications security*. https://doi.org/10.1145/1866307.1866366

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, *4* (3), 382-401. https://doi.org/10.1145/357172.357176

Lewis, S. J., Pereira, O., & Teague, V. (2019). Ceci n'est pas une preuve. Retrieved from https://people.eng.unimelb.edu.au/vjteague/UniversalVerifiabilitySwissPost.pdf

McCorry, P., Shahandashti, S. F., & Hao, F. (2017). A smart contract for boardroom voting with maximum voter privacy. In *Lecture Notes in Computer Science (including subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-319-70972-7_20

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., & Savage, S. (2013). A fistful of bitcoins: characterizing payments among men with no names. In *IMC '13: Proceedings of the 2013 conference on Internet measurement conference*. https://doi.org/10.1145/2504730.2504747

Moran, T., & Naor, M. (2006). Receipt-Free universally-verifiable voting with everlasting privacy (pp. 373-392). https://doi.org/10.1007/11818175_22

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Bitcoin*. https://doi.org/10.1007/s10838-008-9062-0

Nasser, Y., Okoye, C., Clark, J., & Ryan, P. Y. A. (2018). *Blockchains and voting: Somewhere between hype and a panacea*. Retrieved from https://users.encs.concordia.ca/~clark/papers/draft_voting.pdf

Neff, C. A. (2004). *Verifiable Mixing (Shuffling) of ElGamal Pairs*. Retrieved from http://courses.csail.mit.edu/6.897/spring04/Neff-2004-04-21-ElGamalShuffles.pdf

Panja, S. (2018). A secure end-to-end verifiable e-voting system using zero knowledge based blockchain. *IACR Cryptology EPrint Archive*, *2018*, 466. Retrieved from https://eprint.iacr.org/2018/466

Park, C., Itoh, K., & Kurosawa, K. (1993). Efficient anonymous channel and All/Nothing election scheme. In *EUROCRYPT* (pp. 248-259). https://doi.org/http://link.springer.de/link/service/series/0558/bibs/0765/07650248.htm

Patuit, E. (2011). Voting methods. In *Stanford Encyclopedia of Philosophy*. Retrieved from https://plato.stanford.edu/entries/voting-methods/

Rivest, R. L. (2008). On the notion of 'software independence' in voting systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366* (1881), 3759-3767.

Russell Michaels, S. A. (2006). *Hacking democracy*. Retrieved from https://www.youtube.com/watch?v=iZLWPleeCHE

Schneier, B. (2018). Securing elections. Retrieved from https://www.schneier.com/blog/archives/2018/04/securing_electi_1.html

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, *22* (11), 612-613. https://doi.org/10.1145/359168.359176

Simons, B., & Jones, D. W. (2012). Internet voting in the U.S. *Communications of the ACM, 55* (10), 68-77. https://doi.org/10.1145/2347736.2347754

Wolchok, S., Wustrow, E., Isabel, D., & Halderman, J. A. (2012). Attacking the Washington, D.C. internet voting system. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-32946-3_10

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. https://doi.org/10.1017/CBO9781107415324.004

Yu, B., Liu, J. K., Sakzad, A., Nepal, S., Steinfeld, R., Rimba, P., & Au, M. H. (2018). Platform-independent secure blockchain-based voting system (pp. 369-386). https://doi.org/10.1007/978-3-319-99136-8_20

Zacharakis, A., Grontas, P., & Pagourtzis, A. (2017). Conditional blind signatures. *Short Version Presented in 7th International Conference on Algebraic Informatics - CAI 2017*, *2017*, 682. Retrieved from http://eprint.iacr.org/2017/682

Notes on contributors

Panagiotis Grontas is a PhD candidate in the field of Cryptography and Electronic Voting in the School of Electrical and Computer Engineering of the National Technical University of Athens. He holds an MSc in Logic, Algorithms and Computation from the University of Athens (2014), an MSc in Information Systems from Athens University of Economics and Business (2001) and a BSc in Informatics from the University of Piraeus (2000). His research, published in international conferences and workshops, focuses on cryptographically secure and verifiable protocols for electronic elections. He has coauthored chapters in the electronic book "Computational Cryptography" and since 2013 has been a teaching assistant to the cryptography undergraduate course in NTUA. Moreover, Mr. Grontas has experience as a software engineer both in the private and in the public sector. He is also currently employed as a computer science teacher in public education.

Aris Pagourtzis received his Diploma in Electrical Engineering and his Ph.D. in Electrical and Computer Engineering, both from the National Technical University of Athens (in 1989 and 1999 respectively). He is currently an Associate Professor at the School of Electrical and Computer Engineering of NTUA and head of the Computation and Reasoning Lab. He has held academic posts at the University of Ioannina, the University of Liverpool, the ETH Zurich, the University of Athens, and the Athens University of Economics and Business. He has co-authored four books on foundations of computer science, distributed computing, and cryptography and more than seventy scientific publications. His main contributions lie in the areas of network algorithms, distributed computing, approximation algorithms, and computational complexity and cryptography. He has served in the program and organizing committee of various theoretical computer science and cryptography conferences, and as a reviewer for several computer science journal and conferences. His research has often been funded by grants from UK, France, EU, and national resources.