D4: DISTRIBUTED DIRECT DIGITAL DEMOCRACY - A REMOTE ELECTRONIC VOTING PROTOCOL

RASMUS VÄLLING

to obtain a Master of Science in Computer Science
Software Technology
with a Cyber Security specialization
to be defended publicly on October 4, 2018

Delft University of Technology
Faculty of Electrical Engineering, Mathematics & Computer Science
Intelligent Systems, Cyber Security Group

STUDENT NUMBER:
4561058

THESIS COMMITTEE:
Prof. Dr. P. Hartel
Assist. Prof. Dr. Z. Erkin
Assist. Prof. Dr. M. Aniche
MSc. O. Ersoy

SUPERVISORS:
Assist. Prof. Dr. Z. Erkin
MSc. O. Ersoy

ABSTRACT

Recently we have been witnesses to a string of controversial elections: the 2016 US Presidential election, the 2016 Brexit referendum, the 2018 Russian presidential election, the 2018 Zimbabwe elections. The controversies surrounding elections are seemingly endless.

Why are these examples relevant? In the 21st century, we are still conducting most of our voting with paper and pencil. Electronic voting methods could resolve much of the controversies and difficulties regarding conventional elections. It would eliminate the uncertainty of counting and be directly verifiable by anyone, not just by a limited number of officials. Electronic voting also has the benefit of being easier to set up and run periodically, which could be implemented to increase the participation of the population further.

This work focuses on how to utilise blockchain in an electronic voting scheme. The blockchain is perfect for voting application for two crucial aspects. First and foremost, due to its inherent public nature, everyone can verify the facts on the blockchain. Verifiability should be a critical requirement for any modern voting system. Secondly, the consensus mechanism allows anyone to participate in the validation and recording of the election process. Furthermore, being distributed, no single entity has full control over the infrastructure.

In the context of voting, verifiability is as valuable as privacy. If the body of literature indicates that verifiability is an inherent property of the blockchain, this is not true for privacy. We propose a scheme that satisfies the strong privacy requirement of receipt-freeness. Validation has shown that the scheme is suitable for large-scale elections and that it performs well on consumer grade hardware.

Driving is the only thing I love about F1.

— Kimi-Matias Räikkönen

ACKNOWLEDGMENTS

Anyone who knows me knows I am not a man of many words. Therefore this will be brief. First, I would like to thank my supervisors Zeki and Oğuzhan for pushing me. Second, I would like to thank Feddy and Hari for all the support. Finally, I would like to thank anyone else who made this thesis possible.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

DISTRIBUTED DIRECT DIGITAL DEMOCRACY

1 INTRODUCTION

Voting is a method to make a collective decision. It is an essential tool to democracy. The origins of democracy can be traced back to the 5th century BCE ancient Greece. Democracy originates from Greek and means "the rule of the people". Currently representative democracy is prevalent in the free countries [36, 37]. In the representative democracy, constituents elect representatives through periodic elections. It is the primary way for the majority to participate in the democratic process. Therefore it forms the basis of a modern democracy.

Communities, small and large, need to make decisions on a daily basis to function. It is impractical to organise and conduct ballots on numerous decisions. Therefore representative democracy delegates this obligation to the elected representatives. The representatives hopefully reflect the will of the majority in their actions.

Since the beginning of democracy, society has looked for ways to enhance voting in several aspects: efficiency, freeness and fairness. It was first realised that secrecy was necessary for preventing coercion. Then it was found that individuals should be able to verify their vote was counted correctly. Followed by that everyone should be convinced of the correctness of the whole process. Machines, mechanical first and electronic after, have long been used to automate the process of counting and make elections more efficient.

In order to consider elections to be democratic, they need to be free and fair. Numerous international agreements, such as United Nations, 1984 *Universal declaration of human rights* [3] or Inter-Parliamentary Union's, 1994 *Declaration on criteria for free and fair elections* [75], dictate that elections must be free and fair.

Definitions of freeness and fairness differ per agreement or author. However, we summarise them as follows:

- **Free**. The right for any eligible citizen to vote and participate as a candidate in the election.
- **Fair**. All equals will be treated equally during the election process.

Bishop et al. [9] further clarify the freeness to be judged based on the rules of the election and the events leading up to the election. Freeness thus means the right to vote and participate in the election. The fairness is judged based on the process during the election. Fairness, thus means that equals need to be treated equally. *Declaration on Criteria for Free and Fair Elections* [75] additionally states that demo-

cratic elections need to be secret. In order to achieve fair elections, the secret ballot is necessary by definition.

1.1 ISSUES WITH PAPER-BASED VOTING

The secret ballot is a method of voting where the voter's choice remains anonymous. Anonymity or in the context political privacy is necessary to counter voter coercion and discrimination. Coercion in the voting sense can include any activity to change the way a voter is voting whether they are cooperative or forced.

Typically secret ballot is achieved through the voting booth. For example, voter arrives at the voting station. They provide identification to the authority and receive a blank ballot and an envelope. Voting takes place in the privacy of the voting booth. The voter fills in the ballot, puts it in the envelope, and seals it. They exit the voting booth and enter the envelope with the ballot into a box with other ballots. It is expected that the order of the votes in the box cannot be determined thus completing the anonymisation of voting.

This system of voting provides reasonable privacy. However, the prevalence of technology nowadays is eroding the security in the traditional model. The traditional method can be attacked in any step of the process by any actor in the system. For example, blank ballots can be watermarked by the authority giving them out, leaving no visible indicators to the voter. Furthermore, the voter may watermark the ballots themselves. Later the adversary can correlate voters and ballots breaking the security. A voter can take a smartphone into the voting booth to create a receipt for coercer. Asking the voters to enter voting booth without anything can perhaps prevent malicious activities shortly. However, bionic technology augmenting human capabilities will surpass this measure. To summarise, the voting procedure is already exposed to technology in every step of the process, whether or not it is done on paper.

Furthermore, some types of coercion are inherent to traditional voting. Specifically, coercion taking place at the voting stations. Such as violence by national police in the voting stations may intimidate people from voting [13]. On the other hand, the presence of military may intimidate people to vote in a certain way [78]. These forms of coercion can effectively be mitigated by remote voting. However, then the question of verifiability and correctness is raised.

Critics believe that secret ballot elections enable fraud. It is argued that secret ballots can easily be replaced, altered or destroyed [55]. The secret ballot as a measure against coercion has been further weakened by the presence of smartphones. With a smartphone, a voter can record their vote and provide proof to coercer. These examples show that the traditional secret ballot voting does not inherently guarantee the freeness or fairness of the elections.

Modern elections are often characterised by low voter turnout. Hooghe [76] posits that trust stimulates voter turnout and that low trust is associated with populist voting. The study also found that electoral dealignment has led to lower levels of voter turnout and popularity of populist parties. As an example, close results have been demonstrated in recent prominent elections such as the United States presidential elections of 2016 or the referendum of prospective withdrawal of the United Kingdom from the European Union of 2016. Relatively low participation rates also characterised these elections.

According to "*A Theory of the Calculus of Voting*" [68] the voter will vote if the motivation is higher than the cost of voting. Raising the motivation is beyond our scope. Therefore, we focus on lowering the time, effort, and financial cost associated with voting.

1.2 CRYPTOGRAPHY TO THE RESCUE

As explained, the current form of paper-based secret ballot election suffers from several issues regarding privacy, verifiability, and trust. These issues can be remedied with the implementation of cryptographic methods and electronic means. Cryptographic methods can provide mathematical guarantees on the privacy and verifiability. For sufficient security, it is infeasible to do the calculations manually. Thus, electronic means are necessary. Also, it is often pointed out that electronic voting can involve a significant chunk of voters who otherwise can or will not vote, such as people with physical disability or people overseas [46, 48, 50].

WHY ELECTRONIC VOTING? Electronic voting offers higher transparency and verifiability than traditional voting. Traditional voting handles a wide array of exceptions, which results in extensive rules for it. This, in turn, decreases the transparency of the process as a whole [33]. Electronic voting is pragmatic. It costs less to run and to participate; it has a lower carbon footprint, higher efficiency and security, as well as improved accuracy of the tally [50]. There is a significant erosion of security in paper-based voting due to the omnipresence of digital devices in everyday lives [67]. Electronic voting is widely expected to increase the turnout see for example [50, 67]. The reasoning is that due to the convenience people are statistically more likely to participate in voting [68]. Furthermore, it mobilises those groups of people who otherwise will not or cannot vote [50]. Higher turnout reflects the will of the people more accurately thereby increasing the legitimacy of the outcome.

Cryptographic methods for voting can be traced back to mix nets by Chaum 1981 [16] and blind signatures by Chaum 1983 [18]. Numerous electronic voting schemes have been proposed from then on. Electronic and cryptographic means offer several advantages over tra-

ditional paper-based elections such as increased accessibility, cryptographic guarantees, immediate feedback on the casted vote, transparency, coercion evidence [28], and reduced effort necessary to run the elections. However, in nearly four decades since they have not seen wider adoption. Lack of adoption can be attributed to distrust and security concerns.

Cryptographic voting thus needs to be completely open while preserving the privacy of the participants. Similar issues were faced in the financial sector to come up with a currency system that would eliminate the need for a trusted authority. Nearly a decade ago a distributed ledger technology [61] based on blockchain was introduced. The use of blockchain technology completely opens the system making it possible for anyone to verify transactions. As anyone can participate in the running of the system, there is no need for a trusted third party. This technology counters the problems faced by the electronic voting.

WHY BLOCKCHAIN FOR ELECTRONIC VOTING? The blockchain is a promising component for election integrity [33, 45, 50]. Nobody will be in complete control over the entire infrastructure lowering the trust necessary of individual actors [50]. End-to-end verifiable voting schemes require public bulletin boards by definition. Voting schemes use them but often do not detail them. Furthermore, in actual implementations, they are neither append-only, distributed, nor broadcast [62]. Existing bulletin board solutions have expensive assumptions to fulfil making them impractical in reality [33]. As blockchains essentially function as immutable append-only logs, they can be used as public bulletin boards. The radical openness throughout the entire validation process increases the transparency of the system [50]. Furthermore, by actively involving stakeholders in the process the trust in the authorities is significantly lowered increasing the adversarial tolerance [33]. Due to the high distribution, the system is inherently robust as there is no weakest link and damages in case of breach are localised [50, 67]. Thus, the risk is minimised compared to traditional electronic voting schemes. Some surveys have also pointed out the similarity between social processes including elections and blockchain regarding distribution [45, 67]. The similarity can further be used to entangle the system with other official functions such as taxes, smart contracts, property ownership, execution of law and so forth. [45, 62].

The devised electronic voting scheme ensures voter privacy through cryptographic means, provides conclusive verifiability through blockchain and enhance trust by allowing distributing the authority of the election to the concerned voters. Besides, remote voting counters coercion, as seen in recent referendums at the same time not being any worse than absentee voting by mail, and increases the ac-

cessibility. Through these means, it could be possible to increase voter turnout, trust in the system and as a result, have elections with outcomes more accurately reflecting voters preferences. The freeness and fairness of elections are thus increased as well.

1.3 PRIVACY AND VERIFIABILITY CONCERNS

In order to gain trust, the system needs to be inherently verifiable. On the other hand, to be free and fair it needs to preserve the privacy of voters. These are two naturally conflicting requirements. Achieving the secrecy of sensitive information, while irrefutably proving to anyone that everything has been done correctly is the central problem of designing a suitable electronic voting scheme. A tradeoff between privacy and verifiability is necessary, and only one can be satisfied unconditionally [21].

Privacy notion, or as it is known in information security confidentiality, in voting has evolved over time with stronger notions devised in each iteration as follows ballot privacy [18], receipt-freeness [6], coercion-resistance [44], and coercion-evident [28]. Democratic elections need to have the quality of integrity. Integrity makes sure the results are not tampered with, and there is irrefutable proof that everything is correct. Verifiability, or as it is known in the information security integrity, notion has evolved over time as well with the following requirements individual verifiability [18], universal verifiability [69], end-to-end verifiability [14], and accountability [54].

1.4 DIRECT DEMOCRACY IN GLOBAL COMMUNITIES

There is a consensus that the current maturity of electronic voting schemes does not provide strong enough security for high stakes elections such as national elections [50]. However, to eventually get to that maturity both theoretical and practical work needs to be done. We can first validate the implementations in settings where the stakes are low. Thus, we describe a potential scenario where a large scale voting can be conducted periodically. This could serve as a testbed for finding the weaknesses of the system in the real-world example. Furthermore, it is a step towards eventual acceptance and adoption of electronic voting.

Several assumptions about the setting need to be made. First, we assume a setting where the decisions are made on low stake issues. This has a two-fold effect, it significantly lowers the reward and thereby motivation for the adversary, and on the other hand, it lowers the impact of risks. Second, we assume such a global or local community where meeting in person to vote would be impractical due to distance or frequency of voting. This requires that voting be remote and as a

result somewhat weakens the potential coercer. Finally, we assume the risk of coercion is lower in this setting.

Based on these assumptions we have an adversary who can still act according to the established security requirements model but is not able to physically threaten the voter. It is thus a model of an adversary in which remote voting can still be designed to be secure. We assume for different levels of security requirements that the adversary either cannot interact with the voter, interact with the voter after the voting process has ended, or interact with the voter remotely during the voting process.

These assumptions and adversaries match with global non-profit non-governmental organisations such as professional organisations IEEE[1] or IACR[2], open source organisations FSF[3] or Linux[4], or recreational associations such as PADI[5]. Thus the voting in these cases could be done to increase the directness of management by holding more frequent votes on decisions by members, such as approval of budgets or election of organisation president.

1.5 RESEARCH QUESTION

The trust and security concerns are hindering public acceptance and adoption of electronic voting systems. The goal of this work is to address these issues in remote voting for a low stake and coercion risk environments and thus make a step towards the acceptance and adoption. This is achieved by minimising the trust required by utilising the distribution of record keeping with blockchain technology while preserving strong privacy for the voters through cryptographic means.

The main research question of this thesis is:

> *How can we achieve strong notions of privacy for voters in remote electronic voting setting while providing transparency to*

1 Institute of Electrical and Electronics Engineers (IEEE) is the world's largest technical professional organisation dedicated to advancing technology for the benefit of humanity. IEEE has over 420'000 members worldwide. More information at https://www.ieee.org/.

2 The International Association for Cryptologic Research (IACR) is a non-profit scientific organisation whose purpose is to further research in cryptology and related fields. IACR has 1702 members (in 2013) worldwide. More information at https://www.iacr.org/.

3 The Free Software Foundation (FSF) is a nonprofit with a worldwide mission to promote computer user freedom. FSF has over 4'000 active members in 76 countries. More information at https://www.fsf.org/.

4 Linux is a family of free and open-source software operating systems developed by a community of over 1'000 active developers. More information at https://www.kernel.org/.

5 Professional Association of Diving Instructors (PADI) is a recreational diving membership and diver training organisation with an average of over 900'000 certifications each year globally. More information at https://www.padi.com/.

voters through an opportunity to participate in the process of election recording?

The main research question is composed of the following sub-questions:

1. *How to achieve strong privacy in remote voting supported by blockchain?*
2. *How to increase the transparency by employing volunteers in the election validation and record keeping in a distributed manner?*
3. *How to make the process irrefutably verifiable?*
4. *How to make the process simple for voters who only want to participate by voting?*

1.6 DISTRIBUTED ELECTRONIC VOTING

To address the research questions set we systematised the knowledge of blockchain based electronic voting protocols. From there we identified the issues in the existing solutions. First of all, due to the inherent public accessibility of blockchain, how to ensure the strong privacy of the voters. Second, in conventional elections, the active members of the community have a chance to participate by volunteering at polling stations. With remote electronic voting, it is not trivial to offer voters a similar opportunity for transparency. Third, complex cryptographic schemes need to be adequately followed by voters to ensure their privacy. The challenge is to ensure that unintentional mistakes will not strip voter of their privacy. Finally, there should be an option to volunteer in the elections, but for the uninterested, the process should be seamless and straightforward.

Based on these open questions and the set research questions we have designed a remote electronic voting protocol, which uses a community run blockchain as a public election message log. It offers stronger privacy than the existing schemes by being receipt-free. This means that a voter cannot prove to someone else how they have voted, preventing for example vote buying. Meanwhile, the scheme is entirely verifiable satisfying the state of the art verifiability properties. We have implemented a naive proof-of-concept of the design. Validation has shown that it is suitable for large-scale elections using consumer grade hardware, regarding run-time, storage, and bandwidth requirements.

1.7 THESIS OUTLINE

The rest of the work is structured as follows. In Chapter 2 we provide a context for voting. We discuss electronic voting, the security requirements including privacy and verifiability, public bulletin boards in

voting and why blockchain is a suitable candidate, and cryptographic primitives necessary to understand the following content. Chapter 3 dives into the existing blockchain based electronic voting schemes and states the open research questions in the field. Chapter 4 sets the scene for our protocol by describing the modelling, the assumptions, and the properties we aim to satisfy. Chapter 5 describes the protocol we have designed. In Chapter 6 we analyse the design theoretically going through the security, computational and communication complexity. Then in Chapter 7 we validate the protocol experimentally by describing the implementation, conducted experiments, and results. Finally in Chapter 8 we conclude and offer future directions of work.

2

PRELIMINARIES

In this chapter, we provide the background knowledge necessary to understand the rest of the thesis. We begin by detailing what we mean by electronic voting. In order to understand the security context, we provide an overview and development of security requirements in electronic voting. Then we describe the cryptographic primitives necessary to understand prior art in the field and our proposed scheme. Finally, as transparency and verifiability are a significant part of the work we dedicate a subchapter for describing the developments in public bulletin boards in electronic voting.

2.1 WHAT IS ELECTRONIC VOTING?

An electronic voting scheme is any scheme that makes use of digital means. Electronic voting schemes achieve security through the use of **cryptography**. Usually, an electronic voting scheme has a main primitive that it uses for breaking the link between voter and vote or making the audit trail verifiable.

As in traditional voting systems, the electronic voting schemes can be either **remote** or **supervised**. It describes the environment the voting takes place. In traditional voting, the voting stations are publicly observable and conducted by appointed authorities making them an example of supervised voting. On the other hand, in some cases, absentee voters can mail in their votes such as in the example of Switzerland. Sending votes by mail is an example of remote voting. Remote electronic voting usually means that a voter can vote from the convenience of their home using their own devices. In case of supervised electronic voting, the voting stations contain digital means to cast and record votes. In this thesis, we are concerned with remote voting.

Depending on the voting scenario the **performance** regarding necessary computations or privacy achieved can vary. For example, in smaller scale elections such as boardroom voting or local community voting the number of voters can be low. On the other hand, in large-scale elections such as national elections, the system must be able to support a large number of voters. These environmental details set the requirements for the complexity and performance of the protocol. Small-scale elections can be made perfectly secret as the low number of voters are all able to interact with each other and communication is thus not a problem. Whereas, this does not scale to a larger population. In order to be practical, the voting scheme must correspond with the scale of the intended setting.

Voting schemes have some form of **authority** in them. Authority can be used to distinguish eligible voters, conduct the election, count the votes, publish results or do something else. Schemes employ various authorities with different powers. The more the power is divided between authorities, the more risks can be lowered. Thus the **trust** required by the system is diminished.

Due to the lack of control over the voting booth environment in remote scenarios, it is not possible to guarantee absolute security. We make some **assumptions** of the model to achieve security in practice and not attempt to solve all the problems. For example, some schemes assume the existence of a public append-only bulletin board whereas others might assume the existence of secure anonymous communication channel. The fewer assumptions are made about the model, the more practical it is to implement a scheme.

Security comes with a trade-off between usability. **Vulnerabilities** might be known during the design and implementation of a scheme. The **risks** then explained and ways to mitigate them offered. On the other hand, vulnerabilities may be discovered after the proposal is scrutinised. In any case, it is essential to understand the possible **attacks** on a scheme, how to counter them or improve the systems.

Modern verifiable voting schemes require a medium to publish and record data for public scrutiny. It is achieved through **public bulletin boards**. Blockchain, a distributed ledger technology, can be viewed as a form of a public bulletin board. In blockchain voting systems the straightforward way to implement voting is by providing each eligible voter with a token that can then be transferred to preferred candidate wallet. It is an example where the transactions on the blockchain are used straight for voting. In another case, more like traditional electronic voting schemes, the blockchain can be used for recording messages, which are then in a later phase used to compute the final tally. It is an example of a voting scheme using blockchain for recording messages. There can also be hybrid systems, where for example voter first exchanges messages with authorities to prove their eligibility and then in later stage obtain tokens to their ephemeral wallet which they then use for voting.

2.2 SECURITY REQUIREMENTS

In cryptographic voting schemes, the security aspects are roughly divided into two conflicting fields - privacy and verifiability. On the one hand, everyone should be able to follow from the published data that the procedure is correct, but on the other hand, nobody should be able to detect how any individual voted. These are inherently conflicting requirements.

Traces of cryptographic voting schemes go back to the seminal paper on mix nets Chaum, 1981 [16] coining secret ballot and individual

verifiability requirements. Since then the requirements have been divided into two categories and further refined in iterations to provide ever stronger notions of security. In the following, we first introduce the evolution of the privacy aspects followed by the verifiability aspects.

For examples, we define the following actors. Alice and Bob are eligible non-malicious participants of the election. Craig is a malicious actor who may or may not be eligible to vote depending on the scenario and has the intent of coercion or finding out individual votes. Erin is the election authority. Victor is an honest independent verifier of the process. Carol and Dave are eligible honest candidates of the election between whom the voters need to choose. For clarity and where useful, we use red colour to indicate private values and blue colour to indicate public values.

2.2.1 *Privacy*

The privacy in its essence deals with breaking the link between a voter Alice and her choice.

Mix net introduced by Chaum, 1981 [16] can be used to construct elections where individual votes remain private. This is called **ballot-privacy** (BP). Jonker, Mauw, and Pang, 2013 [43] define the secret ballot as *'no outside observer can determine for whom a voter voted'*. Suppose Alice wants to vote for Carol and casts her ballot as such. Then a passive adversary Craig, or anyone else for that matter, should not be able to determine whom Alice voted for just by looking at the ballot.

Privacy was further refined by Benaloh and Tuinstra, 1994 [6] with the requirement of **receipt-freeness** (RF). Jonker et al., 2013 [43] defines receipt-freeness as *'a voter cannot prove after the election how she voted'*. This requirement assumes that Alice as a voter and Craig as her coercer cannot interact during the voting process. Intuitively it then means that after Alice has voted, she has not gained any information to later prove to Craig how she has voted.

Another yet stronger notion of privacy is called **coercion-resistance** (CR) proposed by Juels, Catalano, and Jakobsson, 2005 [44]. Jonker et al., 2013 [43] define coercion-resistance as *'a voter cannot interact with a coercer during the election to prove how she is voting'*. This requirement assumes that Alice as a voter can interact with Craig the coercer during the voting process. Intuitively then Alice should be unable to provide information to Craig so that he will learn with certainty whom she voted for while she is voting.

Intuitively if there is coercion-resistance, then receipt-freeness follows, and if there is receipt-freeness, there is inherently ballot privacy as well. Otherwise, the ballot serves as a proof to coercer Craig.

In recent work, **coercion-evidence** (CE) has been proposed by Grewal, Ryan, & Bursuc, 2013 [28] for remote electronic voting to mitigate

and counter the impact of coercion. Grewal et al., 2013 [28] argue that due to an inherent risk of coercion in remote settings where coercion cannot be prevented via public supervision, there should at least be evidence of how much coercion has taken place during the process. It is achieved by two means. There is a test to determine the amount of coercion taken place and there is coercer independence meaning the ability for a voter to follow a protocol without the coercer detecting the execution of it. Intuitively if Alice votes as coercer Craig has dictated, then there should be a method for Victor the verifier to statistically find it from the results without Craig knowing whether Alice belongs in the set of coerced voters.

2.2.2 *Verifiability*

The verifiability in its essence deals with providing the link for anyone to convince themselves the voting procedure has been performed correctly.

The simplest form of verifiability the **individual verifiability** (IV) was first coined in the seminal work by Chaum, 1981 [16]. Jonker et al., 2013 [43] define it as *'a voter can verify that the ballot containing her vote is in the published set of 'all' (as claimed by the system) votes'*. Intuitively it means that Alice after the election can look at what Erin has published publicly and concluded that her ballot is included in the set of published votes.

This notion was strengthened with **universal verifiability** (UV) by Sako & Kilian, 1995 [69]. Jonker et al., 2013 [43] define it as *'anyone can verify that the result corresponds with the published set of all votes'*. It means that Victor can look at the results and votes Erin published and conclude without doubt that results follow from votes.

The current state-of-the-art is requirement noted as **end-to-end verifiability** (E2E) coined by Chaum, 2004 [14]. It consists of 3 sub-steps that all need to be satisified. The steps are **cast-as-intended** (CAI), **recorded-as-cast** (RAC), and **tallied-as-recorded** (TAR). Exact wording of the steps differs in literature but the overall semantics remain the same.

Jonker et al., 2013 [43] define cast-as-intended as that voters choice is correctly denoted on the ballot by the system. It means that Alice after creating her ballot is convinced without a doubt that it contains her choice of candidate. This can be achieved via methods such as Benaloh, 2007 [5] cast-or-audit protocol where the voter is given a choice to audit a ballot few times until they are convinced it is correctly formed and then cast it. The next requirement in the procedure is recorded-as-cast, which is defined as the ballot being received the way it was cast. Intuitively Erin will convince Alice that the same ballot was received that Alice cast in the previous step. The final step is tallied-as-recorded which means that the ballot was counted as it was

received. Intuitively that means the final results account for all the legitimate ballots published and received by Erin. It has been shown that universal verifiability.

Less studied and employed notion of **accountability** (ACC) was proposed by Küsters, Truderung, and Vogt, 2010 [54]. This attribute requires that after the election process misbehaving parties can be held accountable. They argue that it is a less limited form of verifiability, thus providing stronger notions of security. It differs from the similar notion of coercion-evidence on the privacy side in that it seeks to blame a specific person, whereas coercion-evidence aims to quantify the type of misbehaviour in the process.

2.3 CRYPTOGRAPHIC PRIMITIVES

Cryptographic primitives are essential in electronic voting schemes to achieve security. In the following, we explain the necessary primitives to understand the prior art and the design proposed by this work. We outline what problem they are solving, how they work, and briefly explain how they can be employed in electronic voting through examples. We discuss mix network, blind signature, masking, commitment scheme, secret sharing, and zero-knowledge proof.

2.3.1 *Mix network*

Metadata of network traffic is an invaluable source of intelligence for an adversary. It can be used to strip the privacy of individuals. In order to counter the network metadata analysis, mix network was proposed by Chaum, 1981 [16]. In essence, the mix shuffles the network traffic around with proxy server such that it is difficult to trace the metadata such as who is communicating with who and when.

In Figure 1, a mix network with three servers is depicted. The original sender will encrypt the message with the public keys of the mixes sequentially beginning from the last mix. Each encryption also includes the next destination.

We will explain a "Chaumian mix network" based on layered encryption. Suppose M_n is a particular mix, associated with public and private key pair pk_{M_n} and sk_{M_n}. Assuming A wants to send a message m to B they will proceed by sending first mix the constructed message as follows

$$A \xrightarrow{E_{pk_{M_1}}(M_2, E_{pk_{M_2}}(M_3, E_{pk_{M_3}}(B, E_{pk_B}(m))))} M_1$$

M_1 will then decrypt the received message with their private key sk_{M_1} and obtain the destination M_2 and message to be delivered. Mix then proceeds with

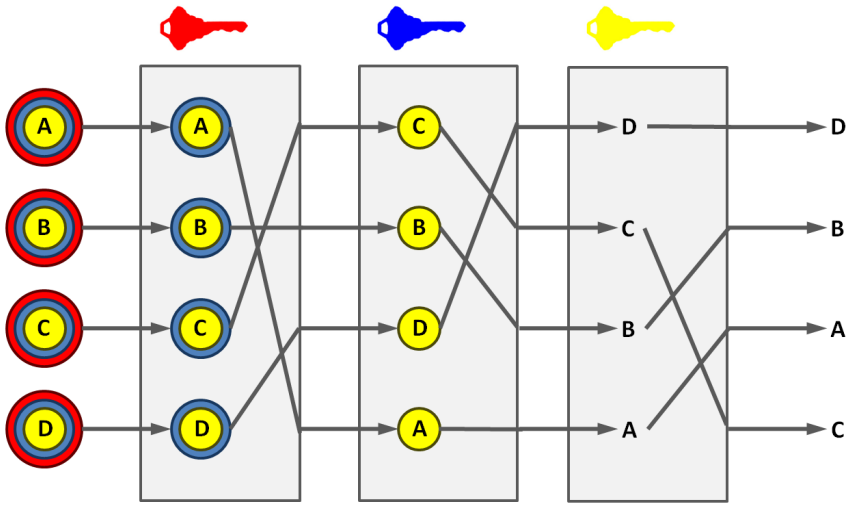

Figure 1: Mixnet diagram (https://en.wikipedia.org/wiki/Mix_network#/media/File:Red_de_mezcla.png)

$$M_1 \xrightarrow{E_{pk_{M_2}}(M_3, E_{pk_{M_3}}(B, E_{pk_B}(m)))} M_2$$

M_2 will then decrypt the received message with their private key sk_{M_2} and obtain the destination M_3 and message to be delivered. Mix then proceeds with

$$M_2 \xrightarrow{E_{pk_{M_3}}(B, E_{pk_B}(m))} M_3$$

M_3 will then decrypt the received message with their private key sk_{M_3} and obtain the destination B and message to be delivered. Mix then proceeds with

$$M_3 \xrightarrow{E_{pk_B}(m)} B$$

B will then decrypt the received message with their private key sk_B and obtain the intended message m. None of the intermediate steps beside the last mix knows where the messages were delivered. Only the first mix will no the origin of the message.

Mixes can be flushed generally in two ways based on time or threshold of some messages. Each has their advantages and disadvantages. Time-based flushing guarantees that message is delivered in specified time. Whereas, threshold-based mixes will be more resilient to time-based network traffic analysis attacks.

In electronic voting, a mix network is used to break the link between the voter and the vote. For example, each mix in the network will collect a threshold of votes, shuffle them and send it on to the

next mix. By the last mix, the list of votes is sufficiently permuted that it would be difficult to determine the original order.

2.3.2 *Onion Routing*

Onion routing [65] is a technique for anonymous communication. Similar to the "Chaumian mix network" it uses layered encryption. Instead of fixed mix servers, it uses nodes from a dynamic network to hinder traffic analysis. In order to establish a communication channel, random nodes are picked from the network and messages sent are encrypted in a layered manner for each of the nodes.

Numerous electronic voting schemes assume the existence of an anonymous untappable channel. Onion routing is a close approximation to such a channel in practice. However, there exist methods to break the anonymity of onion routing, for example [60].

2.3.3 *Blind Signatures*

Blind signatures are a form of signatures which allow the recipient to receive a signee's signature without revealing the content they are signing introduced by Chaum, 1983 [17]. In voting, they are generally used for the voter to create a ballot and obtain a signature on the ballot from authority without revealing the chosen candidate.

In a physical world analogy, they can be explained with an envelope and a carbon paper. Suppose Alice wants to get Bob's signature on a particular contract but does not want to reveal the details of the contract. Alice will then draft the contract and enclose it in the envelope. In cryptographic terms, Alice blinds the contract along with a carbon paper and hands it over to Bob. Bob will verify the envelope indeed came from Alice and will sign the contract through the envelope. The carbon paper will transfer the signature on the contract encapsulated by the envelope. Given that the envelope is still closed on return, the content has not been leaked or tampered. Later Alice can reopen the envelope or in cryptographic terms unblind the contract. Alice now has a signature on a contract without Bob knowing the contents.

A simple blind signature scheme implemented with RSA public key cryptography is depicted in Figure 2. Assume Alice wants to get a signature on message m from Bob with public key (e, N) and private key (d, N). The blinding factor is r. In line 1, Alice blinds message m with the blinding factor r obtaining blinded message m'. Alice sends this to Bob. Bob signs the message as per normal in RSA on line 3 obtaining a signature on the blinded message s'. Bob sends it back to Alice. Alice then unblinds the message by cancelling out the blinding factor with its inverse r^{-1}. Alice has thus the signed message $s = m^d \pmod N$.

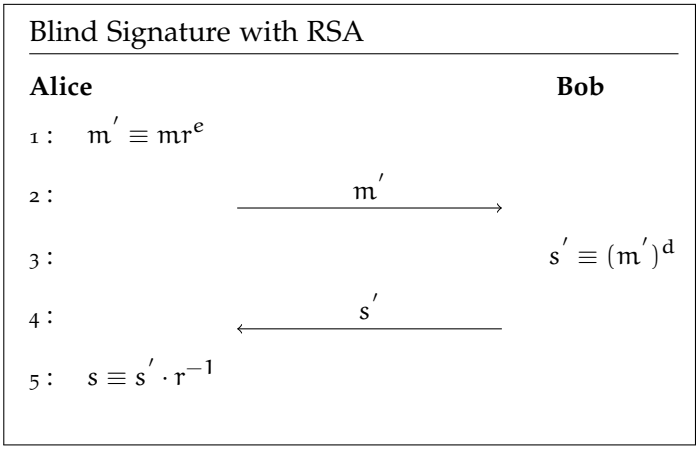

Figure 2: Blind signature with RSA

2.3.4 *Masking*

Masking is the hiding of sensitive data with random data. It is used in cases where sensitive data needs to be protected, but meaningful operations still need to be carried out on it.

In voting, say we have a mix network which needs to shuffle votes to break the link between vote and voter. Assume there are three mixes in total. They can then generate for each mix a mask in a way that the total sum of masks is 0 modulo some N larger than the possible choices in the election. They can then in each step shuffle the list of votes according to a particular permutation and then apply each vote the generated mask. Then they send the next mix the permutation and masked vote. The last mix will also shuffle the list and apply the last mask thereby unmasking the vote. Then the vote can be published for public verification.

In a concrete example assume we have mixes Alice, Bob, and Charlie and a voter Victor who is also the verifier. There are 10 possible options. Victor will cast the ballot with his vote 1 and a randomly generated number 5 as $v = (1||5)$. Alice, Bob, and Charlie will then generate their masks M_A, M_B, and M_C such that $M_A + M_B + M_C \equiv 0 \pmod{100}$. It can be done by generating the first two uniformly randomly and then calculating the last one. For example, $M_A = 34$, $M_B = 19$, and $M_C = 47$. Victor will then submit his vote to Alice. Alice applies the mask $v_A = 15 + 34 = 49 \pmod{100}$ and sends it to Bob. Bob applies the mask $v_B = 49 + 19 = 68 \pmod{100}$ and sends it to Charlie. Charlie applies the mask $v_C = 68 + 47 = 115 = 15 = v \pmod{100}$ and publishes it for verification. Victor then looks up the published votes. Sees that a vote for 1 with his random number 5 is published and is convinced the mix has been done correctly. In this example, masking has hidden Victor's vote from the middle mix Bob. If permutations are also used among a list of votes, then Charlie

would also not realise the link between the vote and Victor. Still, the first mix needs to be trusted.

2.3.5 *Commitment Scheme*

Commitment scheme in cryptography is a primitive which allows for an entity to bind to a chosen value with the option of revealing it in the future. Two phases are making the commitment and revealing. It is useful in cryptographic protocols for multiple entities to choose particular values without revealing them at the moment, which they can not change in the future. Assuming the knowledge of said numbers before revealing own number can provide an advantage to an entity. Thus they can be used to make the execution of the protocol fair for everyone.

In a real example, a box with a lock can be used. Alice first commits to a value by placing it inside the box, locking it with the key and handing it over to Bob. Alice is then said to be bound to the value in the box. In the future, Alice can also provide the key to Bob, thereby revealing the value.

A common commitment scheme used is Pedersen Commitment scheme as seen in Figure 3. Suppose Alice wants to commit to a value m for Bob. Bob selects generators g, h and sends them to Alice. Alice commits to a value as seen on line 3 using a randomisation factor r and sends it back to Bob. In the future, when Alice wants to open the commitment, she reveals the message m and the randomisation factor r. Bob verifies on line 8 that the revealed value was indeed committed to in step 5.

From the security perspective, there are two aspects to commitment schemes the hiding and binding. In the example, Alice hides the value by including a randomisation factor. Hiding refers to the ability of the adversary to find out the value committed. Binding refers to the inability to change the value committed to later. Only one of these aspects can be information theoretically secure, while the other one is computationally secure. For example, the described scheme is information theoretically hiding because there are infinitely many pairs m, r evaluating to commitment c. On the other hand, it is only computationally binding because if Alice can solve the discrete logarithm, they can find other pairs to reveal in the future instead of the initially committed value.

2.3.6 *Secret Sharing*

In cryptography, the security is usually guaranteed by knowledge of a secret. If the secret is guarding something of an exceptionally high

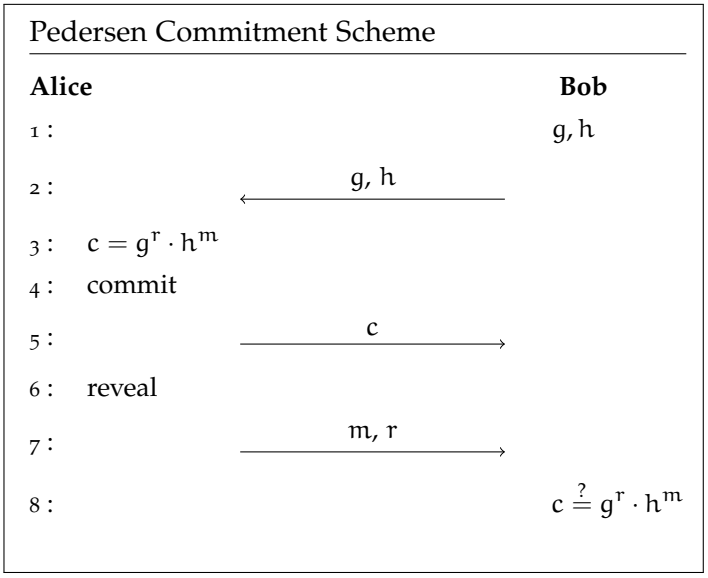

Figure 3: Pedersen Commitment Scheme

impact such as nuclear football [1], it makes sense also to protect the secret. Secret sharing is a method to distribute the secret among a group in shares. Then a subset of the whole group can reconstruct the secret to gain access to the guarded information.

Say there is a nuclear weapon which requires the knowledge of a secret to be launched. There is a considerable impact on activating such a program. Therefore if the power is given to a single person, there are enormous risks associated. Firstly, the single actor can be malicious. Secondly, should the single actor become unavailable it is no longer possible to be activated due to a single point of failure. Thus, it makes sense to distribute it to multiple actors with a certain threshold of actors necessary to reconstruct the secret. Then if someone becomes unavailable, the procedure can still be continued. Furthermore, some conspiring malicious actors are tolerated by the system given a certain number of honest actors remain.

Assume Alice, Bob, Charlie, and Dave share a secret amongst themselves with the requirement that at least three are necessary to reconstruct the secret. Then any subset of the four can be present to restore the secret, for example, Alice, Bob, and Charlie or Bob, Charlie, and Dave. Furthermore, such a secret would tolerate at least two conspiring actors. Because with just two actors they are unable to reconstruct the secret.

Shamir, 1979 [71] proposed a polynomial based secret sharing scheme. It works on the fact that $k+1$ points are necessary to reconstruct a polynomial of degree k. For example, a line requires two points or parabola requires three points and so forth. The idea is that to con-

1 A briefcase which contents can be used to authorise a nuclear attack. More information at https://en.wikipedia.org/wiki/Nuclear_football.

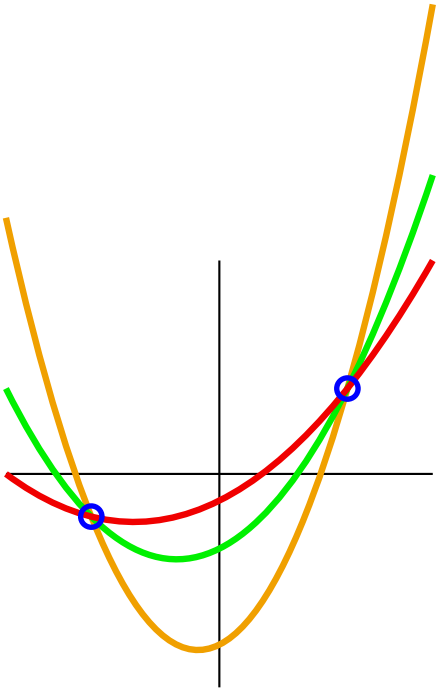

Figure 4: Secret Sharing: 2 points and 3 sample polynomials they are part of (https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing#/media/File:3_polynomials_of_degree_2_through_2_points.svg)

struct a (t, n) secret, where n is the total number of shares, and t is the threshold of required shares, a polynomial of degree $t-1$ is chosen, and n uniformly random points on the polynomial are chosen and distributed. Once the secret needs to be reconstructed, t shares are collected, and interpolation is used to determine the polynomial used revealing the secret.

In Figure 4 we observe two points and three sample polynomials of which they are part. Given a third point, a particular parabola could be determined. In the secret sharing, at least three shares would be needed to reconstruct the secret. Thus it is an example of a $(3, n)$ secret sharing scheme, with the number of shares $n \geqslant 3$.

In electronic voting, it is useful for diminishing the power of single authorities as well as making the system more robust against the attacks against secret holders. For example, it could be the case that voters encrypt their votes with the public key of the authority, with the secret key being shared with a t, n secret sharing scheme. Then for tallying at least t authorities are required to decrypt the votes. Thus no $t-1$ conspiring malicious authorities can decrypt the results before the tally phase.

2.3.7 *Zero-Knowledge Proofs*

Specific scenarios require the proof of knowledge of something without leaking the secret itself. In cryptography, such protocols are called zero-knowledge proofs. They are constructed to sufficiently convince the verifier that a prover knows a secret. For example, assume Peggy wants to go to a nightclub, while Victor is the guard on the door wanting to know Peggy is sufficiently old. A simple solution would be to provide the driving licence. However, showing driving licence will reveal besides the age many other sensitive attributes such as the full name, birth date, licence number, and so forth. Thus besides the proof of age, it leaks further information.

A primitive example of zero-knowledge proofs can be explained regarding a colourblind person and coloured balls. Assume Peggy, the prover, wants to prove to Victor, the colourblind verifier, that two balls are of different colour. Victor cannot distinguish the balls thus no secret is leaked. How do we convince Victor the balls are of different colour? Victor can hold them in his hands, behind his back and take two actions. Either leave them in the same hand or switch them. Then Victor will bring them forward to the view of Peggy. Peggy can now obviously tell whether they have been switched or not. Of course, Peggy could have been lucky since there is $1/2$ chance of guessing correctly. They can then repeat the protocol until Victor is sufficiently convinced that they are indeed of a different colour.

A zero-knowledge proof must satisfy three properties:

- **Completeness** - *if the actors are honest the verifier will be convinced by the prover.*
- **Soundness** - *if the verifier is honest, a dishonest prover cannot prove a false statement.*
- **Zero-knowledge** - *if the prover is honestly following the protocol no secret is leaked to any verifier.*

In practice, zero-knowledge proofs can be used to prove a range of statements. For example, assume there is an election between two possibilities 0 and 1. The votes are aggregated together by homomorphic encryption. To convince others that voter has cast a vote according to the protocol, they may need to submit a proof that their vote is either a 0 or 1. Otherwise, the voter may be incentivised to cheat and give their choice more votes for example by submitting a higher number than 1, thereby giving more votes than allowed. Furthermore, not only does it ensure the following of the protocol, nothing can be learned about how the voter voted.

2.3.8 *ElGamal Cryptosystem*

ElGamal is a probabilistic asymmetric key encryption algorithm for public-key cryptography [25]. Its security relies on the discrete logarithm problem, and it is *IND-CPA* secure. It is homomorphic under multiplication, which is useful for several functions in electronic voting.

It can be defined over a cyclic group (G, q, g). Formally we define the cryptosystem as a tuple of functions $\mathcal{E} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{ReEnc})$.

KEYGEN(λ) The key generation function KeyGen takes as an input security parameter λ and outputs the secret key x and the public key y as seen in Equation 1.

$$(x, y) \leftarrow \text{KeyGen}(\lambda) \tag{1}$$

ENC(m, y) The encryption function Enc takes as an input the message m and the public key y and yields a ciphertext c as seen in Equation 2. A random value $r \in \mathbb{Z}_q$ is chosen, which functions as an ephemeral secret key. The public ephemeral key (also the first part of the ciphertext) is calculated from it $c_1 = g^r$. Shared secret y^r is multiplied with the message to obtain second part of the ciphertext $c_2 = m \cdot y^r$. These two parts form the whole ciphertext.

$$c = (c_1, c_2) = (g^r, m \cdot y^r) \leftarrow \text{Enc}(m, y) \tag{2}$$

DEC(x, c) The decryption function Dec takes as an input the private key x and the ciphertext c and yields a message m as seen in Equation 3. The message is calculated as seen in Equation 4.

$$m \leftarrow \text{Dec}(x, c) \tag{3}$$

$$c_2 \cdot {c_1}^{-x} = m \cdot y^r \cdot (g^r)^{-x} = m \cdot g^{xr} \cdot g^{-rx} = m \tag{4}$$

REENC(c, γ) The re-encryption function adds fresh randomness to a ciphertext to change it without changing the underlying message as seen in Equation 5. It is based on the multiplicative homomorphic property of ElGamal encryption. The idea is to encrypt identity $\text{Enc}(1, y)$ value and then multiply it with the ciphertext as seen in Equation 6.

$$c' \leftarrow \text{ReEnc}(c, y) \tag{5}$$

$$\begin{aligned} c' = \text{Enc}(m, y) \cdot \text{Enc}(1, y) = \\ (g^{r_1}, m \cdot y^{r_1}) \cdot (g^{r_2}, 1 \cdot y^{r_2}) = \\ (g^{r_1+r_2}, m \cdot y^{r_1+r_2}) = \\ (c'_1, c'_2) \end{aligned} \tag{6}$$

THRESHOLD Threshold encryption allows sharing the secret between multiple parties in a way that less than a threshold of parties cannot decrypt the ciphertexts. Here we describe a (n, n) scheme where all the parties need to work together to decrypt a ciphertext. We model the scheme as a tuple of five functions described below.

$$\mathcal{E}_{(n,n)} = \{\text{KeyGen}, \text{PubKeyGen}, \text{Enc}, \text{DecShr}, \text{Dec}\}$$

KEYGEN The key generation function is run n times for the number of parties where each participant $1 < i \leqslant n$ obtains a key pair (x_i, y_i) as seen in Equation 7.

$$(x_i, y_i) \leftarrow \text{KeyGen}(\lambda) \tag{7}$$

PUBKEYGEN The public encryption key y is computed from the shares of each participants public keys $y_1, ..., y_n$ as seen in equation Equation 8. It is calculated as the product of the individual keys $y = \prod_{i=1}^{n}(y_i)$.

$$y \leftarrow \text{PubKeyGen}([y_1, ..., y_n]) \tag{8}$$

ENC The encryption function works just as in the usual case.

DEC The decryption function takes as an input the ciphertext c, individual decryption shares $d_1, ..., d_n$ and computes the plaintext m as seen in Equation 9. Plaintext is computed as follows $m = \frac{c_2}{\prod_{i=1}^{n}(d_i)}$

$$m \leftarrow \text{Dec}(c, [d_1, ..., d_n]) \tag{9}$$

DECSHR The function decryption share for party i takes as an input the ciphertext c and private key x_i and outputs a decryption share d_i as seen in Equation 10. Decryption share is computed as $d_i = {c_1}^{x_i}$

$$d_i \leftarrow \text{DecShr}(c, x_i) \tag{10}$$

2.3.9 *Chaum-Pedersen Discrete Logarithm Equality Proof*

Chaum-Pedersen proof [19] is used to prove given two tuples the discrete logarithm is equal without revealing the discrete logarithm. Given the cyclic group parameters (G, q, g, h) it proves the equality of discrete logarithm of two tuples (g, y) and (h, z) as seen in equation Equation 11.

$$\log_g(y) = \log_h(z) \tag{11}$$

The protocol can be seen Figure 5. Alice knows the discrete logarithm x of y with regard to generator g and the discrete logarithm x of z with regard to generator h. The proof follows three steps. First Alice creates a commitment (a, b) to a random number r. Bob chooses a random challenge c. Alice computes response to the challenge s. Bob finally verifies the challenge by verifying $g^s \stackrel{?}{=} a \cdot y^c$ and $h^s \stackrel{?}{=} b \cdot z^c$.

By applying the Fiat-Shamir heuristic [26] the protocol can be made non-interactive. The challenge is computed by Alice as $c = H(h, z, a, b)$, where H denotes cryptographic hash function.

Formally, we model this protocol as a tuple of two functions $\Sigma_{\mathrm{DLogEq}} = (\mathtt{GenProof}, \mathtt{VerifyProof})$. Proof generation function takes as an input the discrete logarithm x, the value y, the value g, the value z, the value h, the cryptographic hash function H, and outputs the proof $\Pi_{\mathrm{DLogEq}} = (a, b, s)$ as seen in Equation 12. Proof verification function takes as an input the proof and returns true $\top$ if it accepts the proof or false $\bot$ if it rejects the proof as seen in Equation 13.

$$\Pi_{\mathrm{DLogEq}} = (a, b, s) \leftarrow \mathrm{GenProof}(x, y, g, z, h, H) \tag{12}$$

$$\top \vee \bot \leftarrow \mathrm{Verify}(\Pi_{\mathrm{DLogEq}}) = \mathrm{Verify}((a, b, s)) \tag{13}$$

2.3.10 *Andrew Neff's Shuffles of ElGamal Pairs*

One of the most efficient shuffle protocol is due to Andrew Neff[63, 64]. It is a proof protocol for shuffles of ElGamal pairs. It can be applied to electronic voting to create a mix network.

Given an input to mix server $[(X_1, Y_1), ..., (X_k, Y_k)]$ and a shuffled output $[(\bar{X}_1, \bar{Y}_1), ..., (\bar{X}_k, \bar{Y}_k)]$ the protocol is used to generate a proof that the output list is a shuffle and re-encryption of the original list. The proof itself however efficient is rather lenghty thus we refer readers to the full paper for further details in [64].

In electronic voting, we can apply this on a list of encrypted votes to break the link between original permutation and the shuffled list that

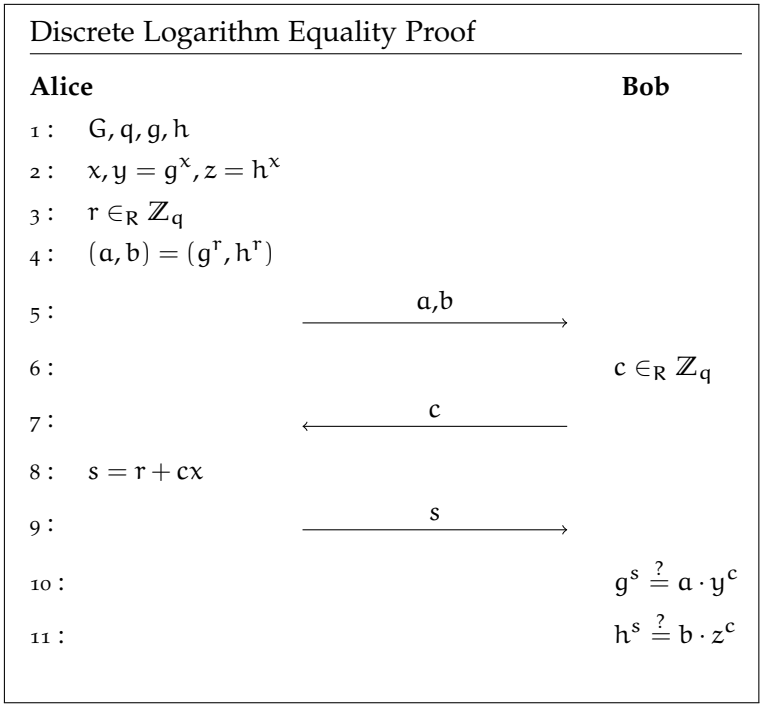

Figure 5: Chaum-Pedersen Discrete Logarithm Equality Proof

will be decrypted. Few mixes can be applied in sequence to ensure that no single mix can recover the link between a voter and decrypted vote.

We denote the shuffled list as seen in Equation 14, where (X_i, Y_i) denotes an ElGamal encryption of a vote i, β is the randomness applied to the list of votes, $\pi(i)$ is the permutation of ith vote, and the number in brackets j denotes the mix that applied the shuffle.

$$(X_i(j), Y_i(j)) = (g^{\beta_{\pi(i)}} X_{\pi(i)}, h^{\beta_{\pi(i)}} Y_{\pi(i)}) \tag{14}$$

2.3.11 *Digital Signature Algorithm*

Digital Signature Algorithm (DSA) [51] is a standard for digital signatures. It is used for authenticating messages. The signer signs a message. A verifier can check the message came from the claimed source. Formally we model it as a tuple of three functions $\mathcal{E}_{DSA} = \{KeyGen, Sign, Verify\}$

KEYGEN The key generation function KeyGen takes as an input security parameter λ and outputs the secret key x and the public key y as seen in Equation 15.

$$(x, y) \leftarrow \text{KeyGen}(\lambda) \tag{15}$$

SIGN The signing function Sign takes as an input the message m, hash function H and returns a signature $\sigma = (r, s)$ as seen in Equation 16.

$$(r, s) \leftarrow \text{Sign}(m, H) \tag{16}$$

In order to compute a signature the following steps are taken. Choose a random value $k \in_R \mathbb{Z}_q^*$. Compute commitment $r = (g^k \pmod p) \pmod q$. If $r = 0$, repeat with new k. Compute response $s = k^{-1}(H(m) + xr) \pmod q$. If $s = 0$, repeat with new k. Then signature $\sigma = (r, s)$.

VERIFY The function Verify takes as an input the signature σ and outputs true $\top$ if it accepts or false $\bot$ if it rejects it as seen in Equation 17.

$$(\top \vee \bot) \leftarrow \text{Verify}(\sigma) \tag{17}$$

In order to check a signature the following steps are taken. Verify $0 < r < q$ and $0 < s < q$ hold, reject otherwise. Compute $v = (g^{H(m)\cdot s^{-1} \pmod q} y^{r\cdot s^{-1} \pmod q} \pmod p) \pmod q$. Accept if $v = r$, reject otherwise.

2.3.12 *Designated Verifier Signature and Proofs*

DESIGNATED VERIFIER SIGNATURE [39] allows a prover Alice to create a signature which only a designated verifier Bob can verify. Intuitively it proves that the prover has either signed a message or is in possession of verifier's secret key. It is achieved by the verifier being able to simulate correct signatures for any message.

Formally, we model it as a tuple of four functions as described below.

$$\Sigma_{DVS} = \{\text{Setup}, \text{Sign}, \text{Verify}, \text{SimProof}\}$$

SETUP The setup function takes as an input security parameter λ and outputs a key pair for the prover (x_A, y_A) and a key pair for the verifier (x_B, y_B) as seen in equation .

$$((x_A, y_A), (x_B, y_B)) \leftarrow \text{Setup}(\lambda) \tag{18}$$

SIGN The signing function takes as an input the secret key of the prover x_A, the public key of the verifier y_B, the message m, and outputs a signature σ as seen in Equation 19.

$$\sigma \leftarrow \text{Sign}(x_A, y_B, m) \tag{19}$$

The prover generates the signature as follows. Prover chooses random values $w, r, t \in_R \mathbb{Z}_q^*$. The prover then computes $c = g^w {y_B}^r \pmod p$, $G = g^t \pmod p$, $M = m^t \pmod p$, $h = H(c, G, M)$, $d = t + x_A(h + w) \pmod q$. The signature is then $\sigma = (w, r, G, M, d)$.

VERIFY The function verify takes as an input the signature σ and the public keys of the verifier y_B and prover y_A and accepts $\top$ or rejects $\bot$ as seen in Equation 20.

$$(\top \vee \bot) \leftarrow \text{Verify}(y_A, y_B, \sigma) \tag{20}$$

In order to verify the proof, the verifier first computes $c = g^w {y_b}^r \pmod p$, $h = H(c, G, M)$. Then accepts if $G {y_A}^{h+w} \stackrel{?}{=} g^d \pmod p$ and $M s^{h+w} \stackrel{?}{=} m^d \pmod p$, otherwise rejects.

SIMPROOF The proof simulation function takes as an input the private key of verifier x_B, the public key of the prover y_A, the message m and outputs a simulated signature σ' as seen in Equation 21.

$$\sigma' \leftarrow \text{SimProof}(y_A, x_B, m) \tag{21}$$

Simulated signature is created by verifier as follows. Verifier chooses $d, \alpha, \beta \in_R \mathbb{Z}_q^*$. Verifier then computes $c = g^\alpha \pmod p$, $G = g^d {y_A}^{-\beta} \pmod p$, $M = m^d s^{-\beta} \pmod p$, $h = H(c, G, M)$, $w = \beta - h \pmod q$, $r = (\alpha - w){x_B}^{-1} \pmod q$. The simulated signature is then $\sigma' = (w, r, G, M, d)$.

Analysis [57] found that the scheme as described is not disavowable. There are two modifications to the scheme they suggest to fix the problem. First, the prover adds proof of $\log_m M = \log_g G$ to the signature. Second, calculate the hash as follows $h = H(c, G, M, s, y_A, y_B)$.

DESIGNATED VERIFIER RE-ENCRYPTION PROOF (DVRP) DVRP [56] are used to prove that a ciphertext has been re-encrypted with the underlying plaintext staying the same. Given ciphertexts $c = (a, b)$ and $c' = (ag^w, bh^w)$ it proves that c' is a re-encryption of c. Intuitively this is achieved by proving g^w and h^w have equal discrete logarithms.

We model DVRP formally as a tuple of functions as described below.

$$\Sigma_{DVRP} = \{\text{Setup}, \text{Sign}, \text{Verify}, \text{SimProof}\}$$

SETUP The setup function is the same as in designated verifier signature.

SIGN The sign function takes as an input the re-encrypted ciphertext c', the randomness w, the verifier public key y_B, and outputs signature σ as seen in Equation 22.

$$\sigma \leftarrow \text{Sign}(c', w, y_B) \tag{22}$$

To construct the signature σ the prover does the following steps. First, chooses random values $k, r, t \in_R \mathbb{Z}_q^*$. Then computes g^k, h^k, $d = g^r {y_B}^t$, $c = H(g^k, h^k, d, c')$, $u = k - w(c + r)$. The signature is then $\sigma = (c, r, t, u)$.

VERIFY The function verify takes as an input the signature σ and the public keys of the verifier y_B and prover y_A and accepts $\top$ or rejects $\bot$ as seen in Equation 23.

$$(\top \vee \bot) \leftarrow \text{Verify}(y_A, y_B, \sigma) \tag{23}$$

In order to verify the proof, the verifier checks the equality in Equation 24 holds, otherwise rejects.

$$c \stackrel{?}{=} H(g^u (g^w)^{c+r}, h^u (h^w)^{c+r}, g^r {h_B}^t, c') \tag{24}$$

SIMPROOF The proof simulation function takes as an input the private key of verifier x_B, the public key of the prover y_A, the re-encrypted ciphertext c', a different ciphertext $\hat{c}$ and outputs a simulated signature $\bar{\sigma}$ that c' is a re-encryption of $\bar{c}$ as seen in Equation 25.

$$\bar{\sigma} \leftarrow \text{SimProof}(y_A, x_B, c', \bar{c}) \tag{25}$$

Simulated signature is created by verifier as follows. Verifier chooses $\alpha, \beta, \bar{u} \in_R \mathbb{Z}_q^*$. Verifier then computes $\bar{c} = H(g^{\bar{u}} (g^{\bar{w}})^{\alpha}, h^{\bar{u}} (h^{\bar{w}})^{\alpha}, g^{\beta}, c')$, $\bar{r} = \alpha - \bar{c}$, $\bar{t} = (\beta - \bar{r})/x_B$. The simulated signature is then $\bar{\sigma} = (\bar{c}, \bar{r}, \bar{t}, \bar{u})$.

2.3.13 *Fiat-Shamir Heuristic*

Fiat-Shamir heuristic [26] is a technique in cryptography to turn interactive proofs into non-interactive proofs. The requirement is that the original interactive proof is a public coin protocol. The heuristic then is to replace the verifier challenge with the output of a random oracle. In practice, cryptographic hash functions are used with the input of public parameters. Example of its application can be seen by Schnorr proof in Section 2.3.14. It is a useful technique to make the verification process independent of the proof generation.

2.3.14 *Schnorr Proof of Knowledge of a Discrete Logarithm*

Schnorr's proof is used to prove the knowledge of a discrete logarithm of a value [70]. Given the cyclic group parameters (G, q, g) it proves the knowledge of discrete logarithm x of y as seen in equation Equation 26.

$$x = \log_g y \tag{26}$$

The protocol can be seen Figure 6. Alice knows the discrete logarithm x of y. The proof follows three steps. First Alice creates a commitment a to a random number r. Bob chooses a random challenge c. Alice computes response to the challenge s. Bob finally verifies the challenge $g^s \stackrel{?}{=} a \cdot y^c$.

By applying the Fiat-Shamir heuristic [26] the protocol can be made non-interactive. The challenge is computed by Alice as $c = H(g, y, a)$, where H denotes cryptographic hash function.

Formally, we model this protocol as a tuple of two functions $\Sigma_{DLog} = (\mathtt{GenProof}, \mathtt{VerifyProof})$. Proof generation function takes as an input the discrete logarithm x, the value y, the cryptographic hash function H, and outputs the proof $\Pi_{DLog} = (a, s)$ as seen in Equation 27. Proof verification function takes as an input the proof and returns true $\top$ if it accepts the proof or false $\bot$ if it rejects the proof as seen in Equation 28.

$$\Pi_{DLog} = (a, s) \leftarrow \text{GenProof}(x, y, H) \tag{27}$$

$$\top \vee \bot \leftarrow \text{Verify}(\Pi_{DLog}) = \text{Verify}((a, s)) \tag{28}$$

2.3.15 *Stealth Address*

Stealth address is a method to hide the recipient of the transaction, introduced in CryptoNote whitepaper in 2013 [77]. It is based on Diffie-Hellman key exchange [24]. Effectively the sender executes half of the key exchange. If a message is intended for a recipient, they can complete the key exchange and decrypt the message. In the following, we describe the elliptic curve based stealth address scheme. In electronic voting, this method can be used to send messages in a way that they are unlinkable to the recipient by a spectator.

Formally we model stealth address system as a tuple of functions to generate a one-time public key and one-time secret key. Encryption and decryption work as usual.

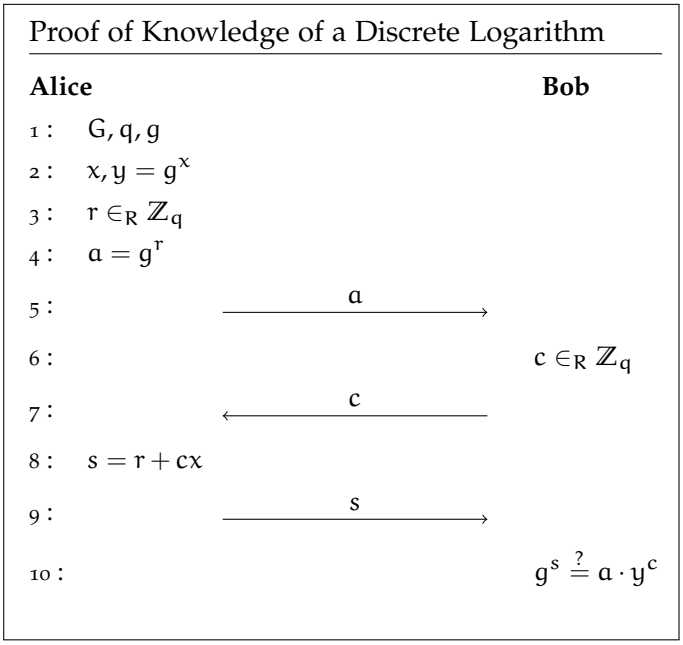

Figure 6: Schnorr Proof of Knowledge of a Discrete Logarithm

$$\mathcal{E}_{ST} = \{\text{OTPubKey}, \text{OTCheck}, \text{OTSecKey}\}$$

Assume Alice wants to send Bob a message. Bob has public keys (A, B) and secret keys (a, b).

OTPUBKEY One-time public key generation function takes as an input the public keys (A, B) of Bob and returns a one-time public key P and a commitment R as seen in Equation 29. Alice picks a random number $1 \leqslant r \leqslant l-1$, computes a one-time public key $P = H(rA)G + B$, and a commitment $R = rG$.

$$(P, R) \leftarrow \text{OTPubKey}(A, B) \tag{29}$$

OTCHECK One-time public key checking function takes as an input Bob's keys (a, B), one-time public key P, and returns true $\top$ if it is for Bob and false $\bot$ if its not for Bob as seen in Equation 30. Bob checks if messages are for him by calculating $P' = H(aR)G + B$, and checking $P' \stackrel{?}{=} P$.

$$(\top \vee \bot) \leftarrow \text{OTCheck}(a, B, P) \tag{30}$$

OTSECKEY One-time secret key generation algorithm takes as an input Bob's secret keys (a, b), public commitment R, and yields a

one-time secret key x. If the message is for Bob, he can calculate the one-time secret key $x = H(aR) + b$ and decrypt the message.

$$x \leftarrow \text{OTSecKey}(a, b, R) \tag{31}$$

2.4 PUBLIC BULLETIN BOARD

In verifiable e-voting, it is essential to publish data for public scrutiny. For without public evidence there is no way to convince oneself of the correctness of the process. Modern verifiable e-voting schemes make use of a building block called public bulletin board, sometimes also referred to as web bulletin board. However, the details such as how it works or what are the exact requirements are not elaborated.

Earliest attempts to construct an append-only web bulletin board by Heather and Lundin, 2009 [32] narrowly predate the release of original Bitcoin proposal by Nakamoto, 2008 [61]. They propose the first robust append-only web bulletin board to the knowledge of the author.

Heather and Lunding, 2009 [32] elicit three agents for the system the web bulletin board, readers, and writers. The bulletin board itself works primarily as a hash list. A message to be posted on the board is hashed with the previous state hash, timestamp, message, writers and bulletin boards signature, and contains some further information. Then the hash of the new message becomes the state hash to be used in the next message.

The proposed bulletin board needs to be append-only, immutable and have accountability on the messages. The accountability is achieved by certified publishing, which works by signatures of writers and bulletin board itself. To make it robust Heather and Lunding, 2009 [32] propose a threshold signature scheme over the peers, requiring that each message is to be signed by some threshold of peers, thus allowing for some failures in the system.

Jonker and Pang, 2011 [42] provided further work on public bulletin boards and a proposal for a verifiable election scheme. They model and measure the privacy in a formal framework, furthermore formalising the notion of coercion-resistance.

Formal analysis of web bulletin board with Event-B [2] modeling framework were done by Culnane and Schneider, 2014 [22]. They propose a distributed public bulletin board with a threshold of honest nodes $t > 2n/3$. They further refined the requirements of public bulletin board by proposing:

1. only items that have been posted to the bulletin board may appear on it;

2 Event-B is a formal method for system-level modelling and analysis. More information at http://www.event-b.org/.

2. any item that has a receipt issued must appear on the published bulletin board;

3. two contradictory items must not both appear on the bulletin board;

4. items cannot be removed from the bulletin board once they are published.

Culnane and Schneider, 2014 public bulletin board is updated in periodic rounds, for example, every hour. Distribution is achieved with a fully connected distributed network of peers. The received messages to be posted are broadcasted to peers, and a threshold signature is created.

An extensive survey on public bulletin boards and further refinement to the Culnane and Schneider, 2014 scheme was done by Kuldmaa, 2017 [52]. They propose a formal model for analysis of security and functionality of public bulletin boards. Furthermore, the extension achieves persistence on $< n/3$ and liveliness on $< n/2$ malicious nodes.

In summary, there has been an effort in research to build a robust distributed append-only immutable public bulletin board to publish messages from the process of e-voting. It needs to be robust because it is critical to the success of fair voting. It needs to be distributed as if a single entity holds power over history they can change it based on who is looking. It needs to be append-only and immutable so that any evidence of tampering is detectable. Lastly, it needs to be public so that anyone can convince themselves of the correctness of the process of voting.

2.4.1 *Blockchain*

With blind signatures, Chaum, 1983 [18] explained, how to achieve untraceable electronic cash. Since then research has been conducted to find a solution which would enable the system to function without a central authority. Satoshi Nakamoto proposed the first practical solution in 2008 [61].

Nakamoto, 2008 describes a distributed ledger of transactions called Bitcoin. It eliminates double spending by eventually reaching a global consensus. The underlying technology, the blockchain, is effectively a public append-only immutable log. As such, it is attractive to various fields beyond finance. In the following, we give a technical overview of the blockchain, the properties relevant to electronic voting, and explain the difference of blockchains based on the access and permissions.

Blockchain system consists of the data structure and the network of nodes mining it. In essence, blockchain is a hash linked list of blocks

of transactions. Each block in the chain contains a nonce, reference to previous block hash, timestamp, and a list of transactions.

The blockchain network runs the system and extends the data structure by mining new blocks. In Bitcoin, mining is done by the Proof-of-Work mechanism. It works by searching for a nonce to the block that satisfies certain criteria. A valid nonce is found once the hash of the block begins with some zeros specified by the current difficulty of the network. The difficulty is adjusted every two weeks to keep the average block mining time around 10 minutes.

A global consensus is achieved in the system by miners mining always on the longest valid blockchain. It may happen that at the same time two valid but different blocks are mined. In this case, miners continue working on the chain they observed first. Eventually, one chain will get longer, and the shorter one will be abandoned. Thus a global consensus is reached in time. It is generally accepted that if a transaction is six blocks deep from the newest block, then it has been confirmed with sufficient guarantee by the network that it is statistically negligible chance of being reverted.

A transaction can contain one or more inputs and one or more outputs. In order to be valid, a transaction input needs to have a reference to a previously unspent transaction and a solution to the problem specified by that output. The outputs specify the amount, public key address, and a puzzle to unlock the output in future transactions. These puzzles generally consist of providing a proof of knowledge of specific private key corresponding to a public key. To make a transaction the user first creates a valid transaction and then submit it to the blockchain network. The miners will verify the transaction and include legitimate ones in future blocks.

The 10 minutes block generation time was chosen arbitrarily by the author(s), but it serves a purpose. If the time would be too short, then the propagation of transactions in the network would take too long time concerning block generation. It would create a situation where lots of forks begin to happen on the blockchain. Thus, the overall performance of the system would suffer.

The wallets in blockchain sense are a pair of the public and private key. Public keys are used as pseudonymous identities. These can be used to reference someone's wallet and to send tokens. Private keys are a secret which unlocks the funds to that particular wallet. Anyone can generate a wallet by generating public and private key pair.

Proof-of-Work is a computationally heavy mechanism for mining consuming lots of energy. Thus several others have been proposed. One of the most vetted is Proof-of-Stake, which replaces search of nonce with the stake of the miners. The stake determines the next miner by chance according to the proportion of tokens owned. It is computationally not heavy. Another approach is to replace the consensus mechanism with Byzantine Fault Tolerance algorithms. The

Byzantine Fault Tolerance (BFT) algorithms solve the Byzantine Generals' Problem. In that problem, generals are attacking a city and need to communicate and make a joint decision otherwise failure is imminent. The downside of BFT algorithms is the scalability. They require nodes to be interconnected and authenticated.

Blockchains can be classified based on the visibility and access. Blockchains which anyone can obtain and view are public, whereas ones which are kept out of public are private. Furthermore, it can be regulated whether anyone can mine new blocks or just a set of specified nodes. Permissionless blockchains allow for anyone to mine. In contrast, permissioned blockchains allow for only specific actors to mine. Bitcoin is an example of a public permissionless blockchain.

Thus given the properties blockchain can offer it makes it attractive to electronic voting research. For e-voting, it can be viewed as a public bulletin board, which is append-only and immutable. Several modern e-voting schemes use this kind of building block (see for example [15, 20, 81]) and is generally considered to be necessary for achieving E2E verifiability. However, the privacy of the voters needs to be guaranteed. Due to its inherently public nature, it poses a significant challenge. Double-spending prevention is essential if votes are transferred by sending tokens. However, if the tally is computed based on cryptographic messages, this becomes irrelevant. It is a promising building block to build a public bulletin board which is not governed by authority.

3

PRIOR ART

The scene of blockchain based voting is vibrant and has seen considerable expansion in the past few years. Some surveys have already studied the field extensively [33, 45, 50, 62, 67]. In this chapter, we want to aggregate and summarise the findings they had, as well as detail our findings. The academic world, governments and companies have shown considerable interest in blockchain based voting. We briefly discuss the real world applications to date in Section 3.1. Then focus on the overview of academic efforts in blockchain based voting in Section 3.2. Finally, we outline issues in blockchain based voting in Section 3.3.

3.1 BLOCKCHAIN BASED VOTING IN THE REAL WORLD

Companies, governments, and other organisations have shown interest in blockchain based voting. Here we provide an overview of the government interest and usage of it, the private sector efforts, and outline use cases already applied in practice.

PUBLIC SECTOR Several government bodies around the world have announced researching or trialling blockchain for voting. We have gathered such instances in Table 1 with the location either city or country, the intended purpose, and when it was announced or used. The voting has been proposed broadly for three different categories - representative elections such as general or local council elections, the direct democratic approach of voter initiative for local government, and for the functionality of conducting votes in elected bodies. Further, Bitcoin blockchain was used to make commitments pre-election in the municipal election of Takoma Park, MD [62].

PRIVATE SECTOR The interest and applicability in the real world have been shown by a large number of companies and start-ups working in this field. The complete list of companies the author is aware of that work on blockchain voting solutions can be seen in Table 2. Many of them have whitepapers, and some have even been used in real-world elections such as Agora in Sierra Leone general election, TIVI in Utah GOP caucus, and Voatz in West Virginia absentee voting. However, due to the competitive nature, the information revealed about the exact mechanics of the protocols are sparse as well as lacking in scientific detail.

Location	Purpose	Date
Moscow (Russia) [1]	Voter initiative	December 2017
Sierra Leone [2]	General elections	March 2018
South Korea [3]	General elections	March 2018
West Virginia (USA) [4]	General election.	May 2018
Zug (Switzerland) [5]	City council	June 2018
Kenya [6]	General elections	August 2018
Ukraine [7]	General elections	August 2018
Spain [8]	Congress voting	August 2018

Table 1: Government Interest in Blockchain Based Voting

FURTHER USE CASES Besides, two distinct areas have emerged where blockchain has been used for voting - shareholder voting and political party internal voting. Shareholder voting has been announced or used by Nasdaq Stock Market[9], and Abu Dhabi Securities Exchange[10]. Political parties (Utah (USA) GOP[11], Denmark Liberal Al-

1 Moscow city government is running pilot project for voter initiative on city management. More information at https://ag.mos.ru/.

2 More information on Sierra Leone use of blockchain for general election at https://techcrunch.com/2018/03/14/sierra-leone-just-ran-the-first-blockchain-based-election/.

3 Korea National Election Commission consulting with private sector to use blockchain for general election, more information at https://techcrunch.com/2018/03/14/sierra-leone-just-ran-the-first-blockchain-based-election/.

4 West Virginia running trials to use blockchain for absentee voters, more information at https://bitcoinnews.com/us-overseas-military-personnel-to-vote-via-blockchain-mobile-app/.

5 More information on Zug city council use of blockchain for voting in http://fortune.com/2018/07/03/blockchain-voting-trial-zug/.

6 It has been reported Kenya will adopt blockchain based voting system. See more at https://www.bloomberg.com/news/articles/2018-08-20/kenya-elections-agency-to-adopt-blockchain-for-vote-transparency.

7 Ukraine is reportedly testing blockchain for elections. See more at https://www.ccn.com/ukraine-election-body-trials-voting-on-an-nem-blockchain/.

8 Political party Podemos is lobbying for the congress to research the use of blockchain for voting in Congress. More information at https://bitcoinexchangeguide.com/spains-political-party-urges-congress-to-study-blockchain-technology-and-its-applications/.

9 Nasdaq reportedly uses blockchain for shareholder voting as outlined by *Nasdaq Blockchain Strategy*, accessible from https://business.nasdaq.com/media/Blockchain%20Mutual%20Fund%20Strategy%20SEB%20and%20Nasdaq%202018_tcm5044-61791.pdf.

10 Abu Dhabi Securities Exchange has partnered with Equifax to explore blockchain voting for annual general meetings, more information at https://www.ccn.com/abu-dhabi-securities-exchange-partners-uk-fintech-for-blockchain-applications/.

11 Republican Party in Utah used TIVI voting internally already in March 2016, more information at https://www.wired.com/2016/03/security-experts-arent-going-like-utahs-online-primary/.

liance[12], Texas (USA) Libertarian Party[13]) around the world have used blockchain for internal voting.

Company	Website
Agora	https://www.agora.vote
Boule	https://www.boule.one
Coalichain	https://www.coalichain.io
CryptoVoter	http://cryptovoter.com
Democracy Earth	https://www.democracy.earth
DemocracyOS	http://democracyos.org
e-Vox	http://e-vox.org
FollowMyVote	https://followmyvote.com
Horizon State	https://horizonstate.com
Milvum	https://milvum.com
Polyz	https://polys.me
SecureVote	https://secure.vote
TIVI	https://tivi.io
Voatz	https://voatz.com
Votem	https://votem.com
Votewatcher	http://votewatcher.com
VotoSocial	http://votosocial.github.io

Table 2: Complete List of Companies Offering Blockchain Based Voting Solutions

3.2 BLOCKCHAIN BASED ELECTRONIC VOTING SCHEMES

The earliest case of using Bitcoin blockchain to commit election values was in 2011 Takoma Park, MD [62] municipal elections. The commitments were opened in a later stage. Effectively the Bitcoin blockchain is used as a timestamping service. These methods are not considered here as they do not use blockchain as an underlying broadcast channel. The focus is on schemes that use blockchain as a bulletin board. It can be realised either using existing cryptocurrencies in unmodified ways to transfer votes or sending messages on the blockchain to directly record the voting process.

12 Danish party Liberal Alliance announced its use of blockchain for internal voting in April 2014, more information in https://www.ccn.com/blockchain-voting-used-by-danish-political-party/.

13 Libertarian Party of Texas announced the use of blockchain for voting internally, more information at https://www.coindesk.com/libertarian-party-texas-logs-votes-presidential-electors-blockchain/.

We briefly describe how each scheme works, followed by a short analysis of which requirements it satisfies or does not and the advantages and disadvantages of the schemes. First, self-tallying protocols are described. We follow by schemes applying blind signatures to votes. Then schemes using existing privacy-preserving cryptocurrencies are discussed. Schemes using ring signatures and homomorphic encryption are then discussed. We conclude by stating the open questions and unsolved problems in the prior art. The overview of the schemes can be seen in Table 3.

3.2.1 *Masking Based*

First, considerable attempt to use blockchain as the primary tool for a voting protocol is due Zhao and Chan, 2015 [83]. It is a lottery inspired voting scheme using techniques of secure multiparty computation to achieve security requirements. It allows for an arbitrary number of voters to choose between two options. The winner is determined by a simple majority and will receive the total funds. Voters jointly execute a protocol to generate random masks for each voter. The total of the masks will eventually cancel out leaving the tally of the election. For a similar method of masking in electronic voting see [30, 31, 47, 59]. Zero-knowledge proofs are used to prove at every step the protocol is being followed. Vote casting is built based on the bitcoin scripting language. Thus upon correct setup, the protocol will execute regardless of any actors further actions. They provide two methods for vote casting by claim-or-refund introduced by Bentov and Kumaresan, 2014 [7] and a joint transaction protocol proposed by Andrychowicz, Dziembowski, Malinowski, and Mazurek, 2014 [2] or Kumaresan and Bentov, 2014 [53]. In essence, the participants will first lock a certain amount of funds in a transaction that they can later refund if they correctly follow the protocol.

This protocol satisfies strong verifiability notions. The voter can verify their vote is cast-as-intended as they will construct it themselves and recorded-as-cast on the blockchain. Furthermore, convinced by the proofs of other voters and the final tally they can be satisfied their vote has been tallied-as-recorded. It satisfies the universal verifiability as well. This protocol achieves perfect ballot privacy if at least one participant generates uniformly random masks.

The downside of this protocol is that for a successful election every single participant needs to follow the protocol correctly. It leaves the protocol open for denial of service attacks, where the attacker will each time lose funds but succeeds in preventing the execution of the protocol. Honest participants will theoretically not lose anything. In practice, they will have to also pay for transaction fees to the miners. Another disadvantage is that the protocol only satisfies the weakest form of privacy the ballot privacy. The mask and the vote will serve

as a receipt to a coercer. Thus it does not achieve receipt-freeness nor coercion resistance. Coercion evidence has not been considered by the authors either.

Smart contracts on Ethereum and self-tallying protocols have been explored for the possibility of voting by McCorry, Shahandashti, and Hao, 2017 [59]. They adapt and implement the protocol proposed by Hao, Ryan, and Zielinski, 2010 [31] on the Ethereum scripting language. In essence, the protocol follows similar principles to the Zhao and Chan, 2015 [83] protocol. The participants jointly generate masks for each voter and in the second round cast and tally the results. Instead of additive masks on the votes themselves, this protocol relies on the fact that the addition of masks in the power of the generator adds up to zero cancelling out the masking factor. It is more complex in a sense that the final tally still needs to be brute-forced from $g^{\sum_i v_i}$. However, given the possible number of voters, the time to calculate final tally is negligible.

The advantage of this protocol is that it provides perfect ballot secrecy. Furthermore, by using the Ethereum scripting language template for voting can be made which significantly eases the administration of elections. It is suitable for small-scale elections such as boardroom environment, due to the scaling of the protocol. It satisfies universal, individual and end-to-end verifiability.

The downside of the protocol is that every participant needs to follow through the whole duration of the election to be successful. Furthermore, the last voter has the advantage of finding out the result before the other participants. It is because, as the last voter they complete the protocol and can compute the final tally before officially announcing their vote. Thereby being able to choose their vote based on previous votes.

3.2.2 *Blind Signature Based*

Blind signatures were first employed for voting in the seminal paper by Fujioka, Okamoto, and Ohta, 1983 [27]. Similarly, it has been explored for blockchain based voting see for example [10, 40, 58].

Blind signatures were first described in the blockchain based voting system by Jason and Yuichi, 2017 [40]. The scheme uses bitcoin blockchain for publishing messages rather than sending votes as transactions. First, the voter creates a commitment to a vote and then blinds it. In a face to face meeting with the administrator, the voter is authenticated and provided with signature on the blinded commitment. Then the voter unblinds and obtains the commitment of the vote signed by the administrator. In the next step, the voter anonymously registers their bitcoin pseudonym by creating a transaction to administrators public key and including the commitment and administrator signed commitment in the OP_RETURN statement. In the opening

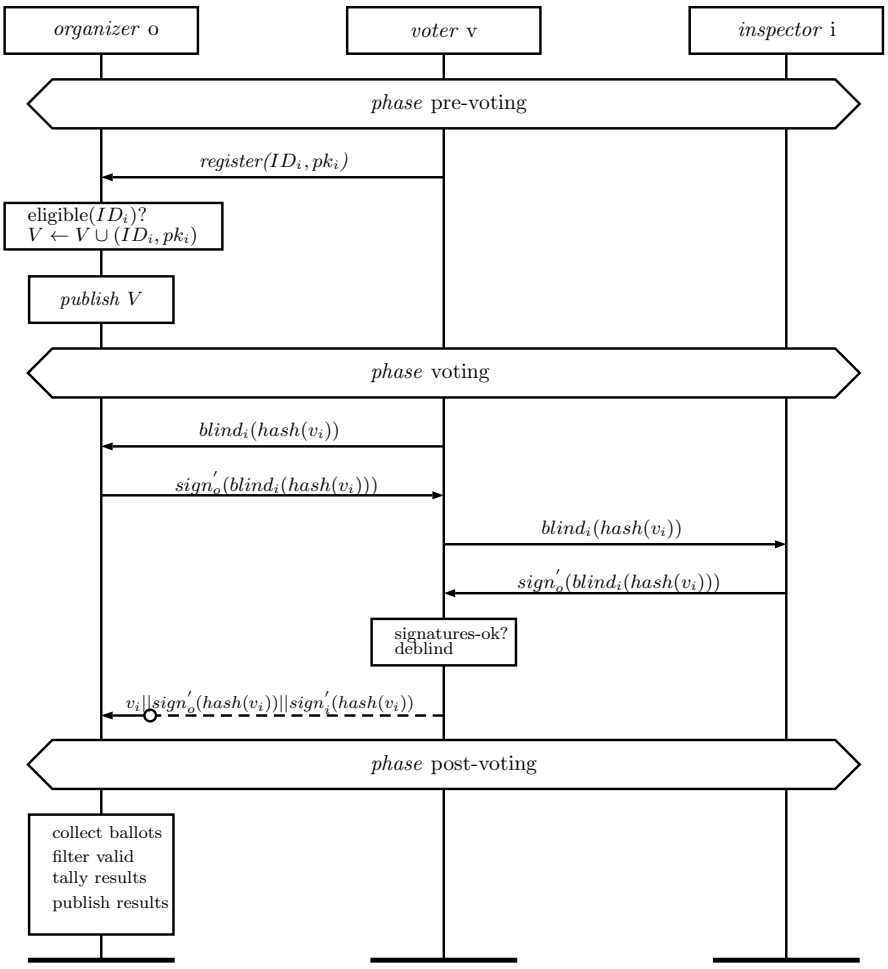

Figure 7: Protocol of Liu and Wang, 2017 [58]

phase, the voter creates a transaction to the counter with the key of the commitment. The counter then counts and publishes all authorised public key addresses and the votes.

This protocol is fair in that it does not keep live tally nor lets any voter find out the tally before the others. Given the commitment scheme used in the implementation, this protocol provides computational or theoretic ballot privacy. The protocol is individual, universal, and end-to-end verifiable.

Due to the usage of Bitcoin blockchain, there is an opportunity to reuse wallets. Thereby unintentionally losing the privacy. The Bitcoin wallet and records on blockchain serve as a receipt to coercer. Thus it does not achieve stronger notions of privacy. Another downside is that the voters need to pay fees to miners to create transactions to authorities. The fees can fluctuate significantly resulting in unfairness and being difficult to predict.

A similar protocol to the previously described one is proposed by Liu and Wang, 2017 [58] depicted on Figure 7. The difference is that no counters are included in the system. Instead, they have an organ-

iser and an arbitrary number of inspectors. The voter first obtains a blind signature using a previously authenticated wallet by creating a commitment to a vote with *sha256* hash function sending it to the organiser. Organiser responds with a signature which the voter then unblinds. Furthermore, the voter will obtain blind signatures in the same fashion from each of the election inspectors. In the next step, the link between the voter and the vote is broken by creating a new pseudonym to transfer the vote and all collected signatures on the commitment to the vote to the organiser.

This scheme is an improvement over Jason and Yuichi, 2017 [40] in that it includes the blind signature directly on the blockchain thus simplifying the process. It supports a nearly arbitrary number of possible voting choices. Inspectors are introduced to diminish the power of the organiser. Diminished power is achieved as all the inspectors need to agree on the eligibility of a particular voter. However, a malicious authority can disagree and thus prevent an eligible voter from participating.

The voter is expected to keep good security hygiene and follow the protocol correctly. Otherwise, the link between voter and vote is unintentionally lost and therefore privacy is lost. Privacy loss can happen in several steps. For example, the voter can create a ballot with insufficient randomness allowing an adversary to brute-force the vote or by using the same wallet for obtaining blind signatures and casting votes.

Similar to blind signatures are anonymous tokens. These have been explored for blockchain based voting by Bistarelli, Mantilacci, Santancini, and Santini, 2017 [10]. The system achieves security procedurally through the division of power and Anonymous Kerberos protocol. It employs two authorities the authentication server (AS) and the token distribution server (TDS). The voter obtains an anonymous token by following the Anonymous Kerberos protocol. First, voter authenticates to the AS and obtains anonymous ID. Then by using the anonymous ID voter receives an anonymous token from the TDS to an ephemeral wallet. It is then used in voting phase to cast a vote by sending the token to preferred candidate.

This scheme is lightweight on the knowledge and participation required from the voter. It diminishes the power of single authority by dividing tasks. It strongly achieves the individual, universal, and end-to-end verifiability notions. Furthermore, it provides ballot privacy, as long as the authorities do not conspire.

However, the protocol is unfair due to the live tally. Any participant can follow the election results as the ballots are cast. The ephemeral wallet and transactions on blockchain serve as a receipt to a coercer, thus stronger notions of privacy are not satisfied. The cost-effectiveness and performance can fluctuate according to the Bitcoin network and are difficult to forecast.

3.2.3 *Privacy-Preserving Cryptocurrency Based*

Some voting schemes have been proposed that base their security on privacy-preserving cryptocurrencies such as Zerocoin and ZCash (see for example [73, 74]). The underlying idea is to use a cryptocurrency which preserves the user's privacy and adapt it to the use case of electronic voting.

Takabatake, Kotani, and Okabe, 2016 [73] use ZeroCoin to break the link between the voter and the vote. In essence, voters receive bitcoin to their wallet. Then they mint ZeroCoins and use them to transfer the funds to eligible candidates. The weakness of the systems is that small ZeroCoin pool can be correlated with network traffic to deanonymize the voters. Furthermore, the authority can use unspent coins to cast votes.

Tarasov and Tewari, 2017 [74] use ZCash in unmodified way. Thus the security follows from what ZCash offers. Individual, universal, and end-to-end verifiability are achieved in certain cases. This scheme also satisfies ballot secrecy. However, the user private keys and records on blockchain serve as a receipt for coercer. Thus stronger notions of privacy are not achieved.

The issue of applying privacy preserving cryptocurrency in an unmodified way to the problem of electronic voting is that it does not take into account the different security model. Furthermore, it can create additional problems, such as the ability of the authority to vote instead of abstaining voters. In some instances, the verifiability can be lowered due to hiding too many details. Further analysis needs to be carried out on the comparison of the two problems and the suitability of these cryptocurrencies applicability on electronic voting.

3.2.4 *Ring Signature Based*

Ring signatures have been explored to design blockchain based voting by Wu, 2017 [79]. The protocol works as follows. Authorities will generate a Bitcoin address pool with funds for voting. Each voter will receive all the addresses from the pool and their public key. They further request the private key corresponding to their public key. Users generate a ring signature among all the public keys, on the candidate and their secret key. A voter uses provided private key to transfer funds back to authority including in the OP_RETURN the commitment to the signature, candidate and election ID. In the tallying phase, signatures are verified and eligible votes counted. Authorities publish the set of all public keys, the ring signatures and votes. Voters can then verify the presence of their vote.

This scheme achieves plausible deniability against coercion. In a sense, the voter is unable to provide proof of receipt to a coercer after the election process. It satisfies individual, universal, and end-to-end

verifiability through blockchain. Through the commitment and ring signatures, ballot privacy is achieved.

However, generating ring signatures on the set of all voters public keys do not scale gracefully. Thus it is suitable for small-scale elections. They claim coercion-resistance but not receipt-freeness. Coercion-resistance is claimed to achieved by ring signatures. The voter can pick arbitrary suitable vote already on the chain to present it to the coercer as their own. Coercer cannot verify the falseness of the statement due to the way ring signatures work. However, this satisfies receipt-freeness rather than coercion-resistance. They further explain that receipt-freeness is not satisfied as the ring signature, hash of the signature and transaction ID form the receipt, and thus receipt-freeness is not satisfied. Intuitively the coercer still cannot verify the link between the signature and the voter. One or both statements they make is false because they are mutually exclusive and vague. Further analysis needs to be conducted to prove or disprove these statements. Thus, while the scheme provides ballot privacy, it is questionable whether it satisfies stronger notions of privacy.

Most recently, short linkable ring signatures and homomorphic encryption were explored for blockchain based voting by Yu et al. 2018 [80]. They propose a smart contract based scheme, which is platform independent. Platform independent means that it can theoretically be implemented on Hyperledger, Cardano, Ethereum, or any other platform. The voting starts by authority uploading the election parameters and triggering the smart contract. Voters will then register their public keys and other identifying information. The keys are then accumulated together in batches, with the bottom half accumulated by the smart contract and the top half accumulated offline by the voter. In the casting phase, the voter will encode the choice, create a correctness proof, and send it to the smart contract. The smart contract verifies the ballot, encrypts it with zero from the pool. If the voter accepts the re-encrypted ballot, they will sign it with short linkable ring signature, and the ballot is finalised and stored on the blockchain. During the tallying phase, the ballots are homomorphically aggregated together, decrypted by the election authority and results decoded from the plaintext.

Reportedly, the scheme achieves privacy, anonymity, double-voting prevention, slanderability avoidance, receipt-freeness, public verifiability, correctness, vote-and-go, and coercion-resistance. Time, computational, and storage complexity wise it is suitable for large-scale elections. The scheme has some disadvantages though. First of all, using short linkable ring signatures has two downsides. List of voters needs to be locked at some point to do the accumulation of public keys. In practice, the eligibility requirement is dynamic because voter eligibility can change, for example, due to birthdays, deaths, or criminal convictions. Furthermore, it provides a single attempt per voter.

A single vote should count per voter, but the voters should have the ability to change their mind and cast differently, in case they were coerced or made a mistake. It is a precondition to stronger privacy requirement coercion evidence and is used in practice in national elections. Second, voting requires participation in multiple steps, first in the registration phase, then in the casting phase twice.

According to our analysis, the scheme does not achieve individual verifiability. In the casting phase, when the smart contract takes zero encryption to add it to the ballot and sends it back with a signature to the voter, the voter has no way to verify zero was added. The uncertainty also affects the recorded-as-cast property. Perhaps this could be fixed by including a zero-knowledge proof that zero encryption was used, but this is made difficult by the fact that smart contract does not know the randomness used in the zero encryption. Otherwise, if the voter can verify the vote in this step, the voter can also construct a receipt. Such receipt would consist of the choice of candidate, randomness used in the Paillier encryption, and the signature from the smart contract. If this is enough to convince the voter, it is enough to convince the coercer. The same phase could also be used to attack coercion-resistance. An adversary can follow through the phase until step five getting back re-encrypted ballot and signature from the smart contract, then force the voter to sign on that and as a result have its choice of the vote go on record.

3.2.5 *Homomorphic Encryption Based*

Homomorphic encryption has been explored for blockchain based voting by Hsiao, Tso, Chen, and Wu, 2017 [38]. Similar to McCorry, et al., 2017 [59] they propose smart contracts for the execution and implementation of the public bulletin board. The security of the protocol is built on Paillier encryption, Shamir's secret sharing, and Rabin's oblivious transfer.

The election progresses in four phases. In the initial phase, the necessary encryption schemes are initialised with key pairs. The registration phase contains two steps and is done offline. A user generates its unique user code and sends it for verification to the Registration Server (RS). If eligible, the RS will certify the unique code and publish the unique user codes on the blockchain. Voter provides the certificate for authentication service (AS) and receives its Paillier keys. Through oblivious transfer, the voter will obtain the AS's signature on the filled ballot. The signature will be distributed via $(3, 5)$ threshold secret sharing scheme to the Distributed Data Servers. In the last billing phase, the eligible cast ballots are decrypted, and the tally is calculated.

The advantages of the scheme are self-executing smart contracts, high distribution of elections tasks to Registration Server, Authenti-

cation Server, Voting Website, Recording Center, and 5 Distributed Data Servers. The scheme achieves individual, universal, and end-to-end verifiability. It also provides ballot privacy. It is transparent in a sense that the voters can participate both in recording and verification of ballots. However, the downside is that this scheme has a high communication complexity and for each ballot significant amount of computations need to be made.

Scheme	Privacy	Verifiability	Based on	Complexity	Votes
Riemann, 2017 [66]	BP	IV, UV, E2E	Aggregation	$O(n \log n)$	M
Zhao and Chan, 2015 [83]	BP	IV, UV, E2E	Masking	$O(n^2)$ or $O(n)$ [14]	M&T
McCorry et al., 2017 [59]	BP	IV, UV, E2E	Masking	$O(n)$	M
Hsiao et al., 2017 [38]	BP	IV, UV, E2E	Homo enc	$O(n)$	M
Bistarelli et al., 2017 [10]	BP	IV, UV, E2E	Anon tokens	$O(n+c)$	T
Wu, 2017 [79]	BP, CR[15]	IV, UV, E2E	Ring sig	$O(n+c)$	M
Yu, et al., 2018 [80]	BP, RF[16], CR[17]	IV[18], UV, E2E[19]	SL Ring sig	$O(n)$	M
Cruz and Kaji, 2016 [40]	BP	IV, UV, E2E	Blind sig	$O(n)$	M
Liu and Wang, 2017 [58]	BP	IV, UV, E2E	Blind sig	$O(n(o+i))$	M
Tarasov, 2017 [74]	BP	IV, UV, E2E[20]	zk-SNARKs	$O(n)$	T
Takabatake et al., 2016 [73]	BP	IV, UV, E2E	ZKP	$O(n)$	M

Table 3: Blockchain Based Electronic Voting Schemes

3.3 OPEN RESEARCH QUESTIONS

In this section, we enlist the open research questions identified in the prior art as well as by the surveys [33, 45, 50, 62, 67].

Blockchain distributed ledger bears a strong resemblance to the immutable public append-only log used in modern electronic voting schemes. It readily satisfies the verifiability requirements of electronic voting schemes. By putting the ballots on the blockchain, anyone can verify them. The chain itself ensures the immutability of the audit trail. As we can see from the existing solutions in Table 3, they easily achieve the standard requirements of Individual Verifiability (IV), Universal Verifiability (UV), and End-to-End Verifiability (E2E).

14 Complexities for two different protocols proposed.
15 Claimed coercion-resistance, but our analysis shows it is not achieved in Section 3.2.4.
16 Receipt-freeness claimed but our analysis shows that it is not achieved in Section 3.2.4.
17 Coercion-resistance claimed but our analysis shows that it is not achieved in Section 3.2.4.
18 Individual verifiability claimed but our analysis shows that it is not achieved in Section 3.2.4.
19 Recorded-as-cast property not achieved as shown by our analysis in Section 3.2.4.
20 Certain variations of the proposed protocol can weaken verifiability.

3.3.1 *Privacy on Blockchain*

Due to the public nature of the blockchain, it is challenging to achieve voter privacy [45, 50]. There are a lot of risks and pitfalls. From network layer, by correlating voter activity times and votes appearing on the chain to breaking the cryptographic primitives used. Thus we have identified a significant gap in the privacy of the blockchain voting schemes, which is advanced and paramount in conventional cryptographic voting schemes.

Thus we state the first open question:

> *1) How to achieve strong privacy in an electronic voting scheme based on blockchain?*

In its essence, public bulletin boards are used in a variety of modern electronic voting schemes, which do satisfy receipt-freeness, coercion-resistance. Thus there must be a way to incorporate them into the blockchain voting schemes.

3.3.2 *Public Supervision and Active Participation*

Another question identified from the literature is the utilisation of the participants. According to Kovic, 2017 [50] blockchain based electronic voting can only lower the risk and work when different societal stakeholders are involved in the supervision and maintenance of the project. There are many similarities between traditional paper-based voting and how blockchain works [67]. Conventional methods for voting utilise volunteers for public supervision of the election process. Furthermore, by involving voters in the validation of the process, we weaken the trust in authorities as at least some nodes have no affiliation to the authorities [62] and increase the adversarial tolerance [33]. Thus far, this has been incorporated into electronic voting schemes as a passive form of participation. We argue that it is possible to involve voters actively to the process in a secure manner and show that given sufficient participation the security of the system can be increased.

Thus we state the second open question:

> *2) How to enable active participation of voters in the process of voting?*

3.3.3 *Honest Unintentional Mistakes Without Loss of Privacy*

State-of-the-art blockchain based voting schemes require significant cryptographic knowledge and correct behaviour from the user. There is a chance of breaking the privacy by the voters through unintentional deviation from the execution of the protocol. These errors can be bypassed by implementing software that takes care of this for the

users and follows protocol appropriately. However, the protocol could be designed in a way that the wrong behaviour does not break user privacy, or interrupt the process, and thus guarantee it through cryptographic measures. According to Heiberg et al., 2018 [33] none of the existing blockchain voting schemes has a full list of checks needed to establish internal consistency.

Thus the third open question is:

> *3) How to ensure cryptographically that honest but unintentional deviation from the protocol does not break voters privacy?*

3.3.4 *Minimizing User Effort*

State-of-the-art blockchain based voting schemes require user interaction and knowledge of cryptography in numerous steps such as registration, casting a ballot, verifying tally, and so forth. This level of involvement raises the risk of user fatigue of the system and poor security hygiene.

Thus the fourth open question is:

> *4) How to design a blockchain based electronic scheme limiting the knowledge and participation the user has to go through to vote?*

This final question concludes our overview of the prior art. We now proceed to explain, how we have attempted to answer those open questions and the posed research question in the following two chapters.

4

SETTING THE SCENE

This chapter sets the scene necessary to understand the context of our protocol description. We begin by laying out the modelling, introducing the actors of the system, and the voting system functionality in Section 4.1. Modelling is followed by a description of the threat model and the assumptions we make in Section 4.2. Then we enlist the properties we aim to satisfy with the design in Section 4.3. Finally, we describe the requirements we have of the public bulletin board Section 4.4.

4.1 MODELLING

Here we describe the voting system model. We first explain the actors involved in the system the authorities, the voters, and the blockchain nodes. Then we formally describe the voting system functions.

4.1.1 *Actors*

A voting system $\mathcal{VS}$ consists of some actors. The word *authority* indicates a person or an organisation, who has a specific administrative power and control over the listed tasks.

- **Election Authority** ($\mathcal{EA}$) - authority in charge of setting up the election, publishing question and options, setting the deadlines of phases, and calculating the final tally of the election. There is one $\mathcal{EA}$.
- **Authentication Authority** ($\mathcal{AA}$) - authority in charge of authenticating the voters in the election. There is one $\mathcal{AA}$.
- **Mix Authorities** - denoted by $\mathcal{M} = \{M_1, M_2, ..., M_{n_M}\}$, this is a set of n_M actors responsible for mixing the votes before they are decrypted. Andrew Neff's shuffle supports arbitrary number of mixes. If at least one of them is honest, the privacy of the votes is guaranteed. Note that, we assume the $\mathcal{M}$ are semi-honest and thus follow the protocol correctly. As a result, $\mathcal{M}$ always decrypt the votes correctly.
- **Voter** - a person who votes and has a right to vote in the election run by $\mathcal{EA}$. The set of voters $\mathcal{V} = \{V_1, V_2, ..., V_{n_V}\}$ consists of n_V voters denoted by V_i for $1 \leqslant i \leqslant n_V$, where i denotes the public identifier of a given voter.

- **Blockchain Nodes** - Denoted by $\mathcal{N} = \{N_1, N_2, ..., N_{n_N}\}$ is dynamic set of entities running the election blockchain network, this can be any authority or voter and thus is a subset of them $\mathcal{N} \subseteq \mathcal{EA} \cup \mathcal{AA} \cup \mathcal{M} \cup \mathcal{V}$.

We make use of a *blockchain*, denoted by $\mathcal{BC}$. $\mathcal{BC}$ is a universally accessible broadcast channel, which all the actors have appendive-writing and reading access. Thus, once something has been written to $\mathcal{BC}$ it can no longer be removed from it.

4.1.2 *Algorithms*

The voting system algorithms are formally defined based on the Bernhard et al. 2015 [8] formalizations. We define a voting system $\mathcal{VS} = \{\text{Setup}, \text{Vote}, \text{Valid}, \text{Mix}, \text{SimProof}, \text{Tally}, \text{Verify}\}$ as the collection of a number of algorithms as described below.

SETUP(λ) The algorithm setup takes as input a security parameter λ and outputs an election public key pk and secret key shares $[sk_1, \ldots, sk_{n_M}]$ for each mixnet as seen in Equation 32.

$$(pk_M, [sk_1, \ldots, sk_{n_M}]) \leftarrow \text{Setup}(\lambda) \tag{32}$$

VOTE(id, v, r) The algorithm Vote takes as input the id of the voter, the choice v, fresh random value r and yields a ballot b, which has been encrypted with the public key of the election, as seen in Equation 33.

$$b \leftarrow \text{Vote}(id, v, r) \tag{33}$$

VALID(b) The algorithm Valid takes as input the ballot b and yields a re-encrypted ballot b', designated verifier re-encryption proof of the ballot δ, and $\mathcal{AA}$ signature on the re-encrypted valid ballot $\sigma(b')$, as seen in Equation 34.

$$(b', \delta, \sigma(b')) \leftarrow \text{Valid}(b) \tag{34}$$

SIMPROOF($b', b, \delta, \bar{v}, \bar{r}$) Simulated designated verifier re-encryption proof can be created with SimProof algorithm. It takes as an input re-ecnrypted ballot b', original ballot b, new vote $\bar{v}$, new randomness $\bar{r}$ and produces a simulated proof δ' as seen in Equation 35.

$$\delta' \leftarrow \text{SimProof}(b', b, \delta, \bar{v}, \bar{r}) \tag{35}$$

MIX($\mathcal{B}$) The algorithm Mix takes as input an ordered list of valid ballots ($\mathcal{B}$). Mix algorithm yields a shuffled and re-encrypted list of ballots $\mathcal{B}'$ and a proof Π of shuffle and re-encryption, as seen in Equation 36.

$$(\mathcal{B}', \Pi) \leftarrow \text{Mix}(\mathcal{B}) \tag{36}$$

TALLY($\mathcal{B}', \{sk_1, \ldots, sk_{n_M}\}$) The algorithm Tally takes as input shuffled and re-encrypted list of ballots $\mathcal{B}'$, the secret key shares of mixes and yields a tally vector $\mathcal{O}$, as seen in Equation 37.

$$(\mathcal{O}) \leftarrow \text{Tally}(\mathcal{B}', \{sk_1, \ldots, sk_{n_M}\}) \tag{37}$$

VERIFY($\mathcal{BC}, \mathcal{O}, \Pi$) The algorithm Verify takes as input the blockchain $\mathcal{BC}$, tally vector $\mathcal{O}$, and proof Π. It checks the correctness of the proof and returns true $\top$ or false $\bot$, as seen in Equation 38.

$$(\top \text{or } \bot) \leftarrow \text{Verify}(\mathcal{BC}, \mathcal{O}, \Pi) \tag{38}$$

4.2 THREAT MODEL AND ASSUMPTIONS

The proposed system assumes a particular threat model. In this section, we outline the assumptions about the adversary that we have made and motivate why such choices were made.

4.2.1 *Adversary*

First, we make some assumptions about the capabilities of the adversary. We assume the adversary is computationally bounded, can only coerce the voter remotely, and can interact with an honest voter after the election has ended.

Assumption 1. *Adversary is computationally bounded.*

The system does not provide perfect secrecy. This assumption is necessary to design a practical system. Perfect secrecy would require the secure generation and distribution of data at least equal to the amount of data communicated between parties as shown by Shannon [72]. Perfect secrecy is feasible for scenarios with a low number of participants but does not scale to a more significant number of participants. Thus, we focus on an efficient probabilistic polynomial time (PPT) adversary $\mathcal{A}$.

Furthermore, we assume the Decisional Diffie-Hellman (DDH) assumption holds for the security. DDH is a pretty standard assumption, and we will not discuss it here any further. Readers can refer to Dan Boneh 1998 [12] for further information.

Assumption 2. *Adversary is semi-honest.*

Our adversary can tap into the communication, record the ciphertexts, and in the future try to decrypt them.

Assumption 3. *Adversary can only coerce the voter remotely.*

As we design a remote voting system, we assume that any coercion takes places remotely and adversary does not have physical access to voters. This requirement reflects the reality, where coercing a significant amount of voters physically in short period would be unscalable for an adversary.

Assumption 4. *Adversary cannot interact with the voter during ballot preparation and recording.*

We effectively prevent the adversary from simulating the voter. Naturally, there must be a phase in which voter can act independently. Otherwise, the adversary could simulate the voter and the system would be unable to distinguish between the voter and the adversary. Therefore, the adversary would have complete control over the voter. As such, we assume the voter has access to a **voting booth** environment during ballot preparation and recording.

4.2.2 *Authorities*

As a significant point of failure, certain assumptions have been made about the authorities.

Assumption 5. *Authorities are semi-honest.*

The authorities have a significant impact because they handle sensitive data. Therefore, it makes sense to assume they are somewhat of a malicious nature. Therefore, as with the adversary, we assume they can listen, record, and in the future try to crack the ciphertexts.

Assumption 6. *Authorities do not collude.*

Authorities have been separated by different tasks to limit the powers of individual actors. The idea is to give the authority to entities, which are mutually distrusting and therefore discouraged to work together in order to break the security of the system.

4.2.3 *Voters*

Generally, in any elections, there may be some malicious voters. If the voter willingly acts together with an adversary, there is nothing a scheme designer can do to prevent privacy violations. Thus, the assumptions about voters focus on honest voters.

Assumption 7. *Voters will not share their private keys with the adversary.*

We assume an honest voter will effectively not give their identity to the adversary. This assumption closely reflects the reality for example in the cases where the voter keys are tied to the voter in the legal sense. For example, in Estonia, the cryptographic keys used for voting are the same that are used for all other legal functions. Therefore, by sharing them with an adversary, the individual would effectively be volunteering their legal identity to a voter. Thus, given the legal and financial incentives, it is reasonable to assume an honest voter would not do this.

Assumption 8. *Voters are malicious with respect to coercer.*

An honest voter will not co-operate with the adversary. We provide in our scheme mechanisms to simulate to a coercer certain proofs and assume an honest voter makes full effort to take advantage of these functionalities.

Assumption 9. *Voter hardware and software is trusted.*

In literature, there are numerous methods to combat vulnerabilities regarding these issues such as out-of-band code sheets, trusted hardware tokens, or trusted random number generators (see for example [11, 14, 29, 34, 82]). Furthermore, a lot of these methods are not cryptographically interesting, but rather due to implementation decisions. Therefore, we leave this out of scope and assume the environment is safe.

Assumption 10. *There is an existing public key infrastructure for all the actors.*

We assume there exists a public key infrastructure for all the actors. Also, the mixes will jointly generate the public key for the ballot encryption.

4.3 PROPERTIES

Any voting scheme should satisfy some properties to be practically applicable. In this section, we examine the privacy, correctness, verifiability, and some auxiliary properties that our proposed scheme aims to satisfy.

4.3.1 *Privacy*

Privacy is elementary for any election with an impact. The design aims to address ballot privacy and receipt-freeness properties. Furthermore, the design combats replay attacks thereby offering ballot independence and does not provide live tally, therefore, satisfying fairness property. We defer formal definitions to the security analysis in Chapter 6.

BALLOT PRIVACY Informally, ballot privacy requires that any observer should not be able to distinguish for whom a particular voter voted.

RECEIPT FREENESS This property states that a voter should not be able to construct a proof after the election to prove to a coercer how the voter voted. Hugo Jonker 2005 [41] has defined the receipt r to have the following properties concerning a voter v and candidate c:

1. r can only have been generated by v,
2. r proves that v chose candidate c,
3. r proves that v cast her vote.

BALLOT INDEPENDENCE The voting system should prevent replay attacks. In a replay attack, an adversary records the messages a voter sends to the blockchain. After the fact, the voter may change their choice and send a new ballot. If the attacker now sends the first recorded ballot again, it should not affect the choice of the voter. In another scenario, the adversary should not be able to create their ballot based on a previously recorded ballot they have tapped.

FAIRNESS The voting system should not have a live tally. Providing a live tally may affect the independence of the voters' choices who have not yet voted, which is a form of coercion in itself.

4.3.2 *Correctness and Verifiability*

Following properties are related to the integrity of the voting system and how specific actors can verify the results.

CORRECTNESS If all the actors behave correctly, then the tally should be computed corresponding to the submitted votes. Formally, we have defined correctness as seen in Definition 3.

VERIFIABILITY The final tally is verifiably correct. Verifiability is achieved through sub-properties. First, all the voters should be able to verify their ballot has been included. Second, anyone should be able to verify the tally corresponds to the set of all eligible votes. Third, the eligibility of the tallied votes should be verified. Finally, the votes should be end-to-end verifiable.

INDIVIDUAL VERIFIABILITY A voter should be able to verify their vote is in the set of all eligible votes as published by the authorities.

UNIVERSAL VERIFIABILITY Anyone should be able to verify that the set of eligible votes as published by the authorities corresponds to the final tally.

END-TO-END VERIFIABILITY End-to-end verifiability property ensures that a voter is convinced of the correct handling of their vote through its lifecycle. First, the vote should be cast the way the voter intended. Second, the vote should be recorded the way voter cast it. Finally, the vote should be tallied as it was recorded.

ELIGIBILITY VERIFIABILITY Only eligible voter can cast a vote and only eligible votes count towards the final tally, with the last vote per voter counting.

4.3.3 *Functional properties*

Beyond the integrity and confidentiality properties, we have some administrative and auxiliary properties the design aims to satisfy. As stated before, we designed the scheme to be **remote**. The vote can be conducted on elections with two or more options. As with any critical system, **availability** is an essential security property. The design mitigates the risk related to availability with separation of tasks, high distribution, and offering participation in a scalable way. However, we do not stop any further on this property as it is cryptographically less interesting.

We designed the scheme to be **publicly observable**. Thus the blockchain it utilises is publicly accessible. However, in an attempt to mitigate risks of availability, appending to the blockchain is **permissioned**. Therefore, only certified actors can write to the blockchain. Certified actors constitute of the set of all actors as described previously.

4.4 PUBLIC BULLETIN BOARD

The protocol is designed in the way that the entire communication of the voting is persisted in a public log. Therefore, any messages sent are stored and mediated through a public bulletin board. For the public bulletin board, we propose to use blockchain.

The design and implementation of an aspect specific blockchain for electronic voting entail in itself research orthogonal to a cryptographic voting scheme. Therefore it should be an independent building block. Thus, we leave it out of the scope of this work and only describe the requirements we have of the blockchain.

The main idea of the blockchain is to create an immutable log of the voting process. Ideally, this log is created by the voters. All of the information on the blockchain is public for anyone to view and

scrutinise. However, the mining process is permissioned to a set of stakeholders. The participation is incentivised.

4.4.1 *Assumption*

We assume that all the actors have certified public keys necessary for the running of the blockchain. Anyone who has such a certified key can participate in the mining process. Any miner is also running as a node in the onion router used to create an anonymous communication line to send transactions on the blockchain.

4.4.2 *Actors*

First of all, there are some actors involved in the system. For electronic voting, the interesting ones are the developers of the software, the miners, and the users of the system. The developers of the system should be chosen by the election authority, and the security of the software verified independently of the developers. The miners should be anyone with a stake in the election process, therefore, for example, the voters, the authorities, parties and so forth. Thus, the blockchain should be permissioned to the election stakeholders only. The voters are a set of actors who have a stake in the outcome of the election and are identified by the election authority before the elections begin. Any actor can submit transactions to the blockchain provided they have identified key pair by the election authority.

4.4.3 *Transaction and Block*

Transaction header	From: *source*	To: *destination*
	Hash of message	Timestamp
	Signature of message (required in some messages)	
Message	Cryptographic Message Data	

Figure 8: Public Bulletin Board Transaction Structure

The election voting protocol consists of steps of communication between the described actors. Transaction structure can be seen in Figure 8. A transaction has three attributes *from*, *to*, and the *message* itself. Of the three, only *message* is a necessary part of the transaction. *To* and *from* attributes are used to identify where the message is from or who is the intended recipient. In the case *from* attribute is used, a signature authenticates the message. The *to* attribute helps intended recipients collect messages from the blockchain. If the *to* attribute is missing, this means the message is broadcast to anyone and is usually

used for publicising commitments and facts for election verifiability. Each transaction is timestamped.

The blocks contain transactions collected during a period between block creation. It further links to the previous block by hash, has a timestamp, number of transactions in the block, the signature of the block miner for future reference and any further information required by the consensus mechanism.

4.4.4 *Consensus*

Bitcoin blockchain reaches consensus by Proof-of-Work [61]. This mechanism attempts to prevent double spending by incentivising honest behaviour and making the malicious behaviour more expensive. In our voting scenario, double spending is not a problem, because transactions do not send tokens. Furthermore, as seen in existing systems this consensus mechanism seems to centralise into parties with the most processing power. Once the centralisation reaches a critical point, it can choose which transactions are going to be included in the blocks and which not.

As double spending is not an issue and we aim for high distribution the blockchain should have Proof-of-Stake consensus mechanism. In the voting scenario, the stake can be distributed equally among all the actors or all the actors who show interest in running the network. Then anyone who wants can participate in running the network and participants are not decided by the processing power they own. The intention is thus to let voters run the election process. However, to make sure the network continues to run the authorities are always required to be running.

For the Proof-of-Stake, Ouroboros scheme [23, 49] could be used for example. It has been proven to be secure and satisfies the requirements for the consensus mechanism we have outlined previously.

4.4.5 *Incentives*

To incentivise honest behaviour, and limit malicious behaviour we have the following three proposals.

Tokens are distributed by the election authority upon request and used mainly to incentivise mining and prevent denial of service. If a participant wishes to broadcast a message they pay a token as a fee. Miner gains the tokens. Later these can be used to claim payment from the Election Authority for work done. They can only be transferred by authority or as a fee for sending a message.

Participant interacts with the election authority out-of-band to get token necessary to broadcast their message. The tokens should prevent denial of service as participants have a limited number of messages they can broadcast. However, if the voter changes their mind

and wishes to revote, the voter needs to authenticate with authority again to receive the new token. This authentication will be time limited to ensure someone is not flooding the blockchain.

Transaction spamming could be limited with proof-of-work [4]. In effect, this makes the creation of transaction expensive in comparison to the verification of the transaction. The difficulty should be high enough to deter any malicious entity, but still be usable for honest participants.

5

D4: DISTRIBUTED DIRECT DIGITAL DEMOCRACY

In this chapter, we propose a novel electronic voting scheme employing blockchain technology for full transparency. Blockchain has the benefit of increasing system transparency being inherently public and enabling the participation of voters on a larger scale. This scheme, to the knowledge of the author, is among the first which provides receipt-freeness property among the existing blockchain voting protocols.

Briefly, we achieve our goals by largely three different cryptographic constructs designated verifier re-encryption proofs [35, 39] (DVRP), Andrew Neff's universally verifiable shuffle [63, 64], and onion routing [65]. Intuitively, the designated verifier re-encryption proof is used to prove to a voter that fresh randomness was added to a ballot in a way that voter cannot find out what was added. Fresh randomness makes the voter unable to create a receipt, and thus receipt-freeness property follows. Mixnet is used to hide the link between a voter and the decrypted vote from the authorities. Onion routing is used to create an anonymous channel between a voter and the public bulletin board to counter network analysis. The public bulletin board is provided by the custom blockchain we describe in Chapter 4.

We proceed by first giving an overview of the protocol phases and then describing each in detail.

5.1 OVERVIEW

The protocol follows established common steps as listed by Ben Adida in 2006 [1]:

1. Setup,
2. Ballot preparation,
3. Ballot recording,
4. Anonymization & Aggregation,
5. Results.

The schematic of the actors and data flow can be seen in Figure 19. It assumes an untappable channel between the voter and the authority and that the mix network and authority do not conspire. However, it has several problems. First, how does the voter know their vote is included in the published set of votes. Secondly, the mix knows the

order of the votes. Thirdly, the authority or the mix can add votes to the list and control what they publish, without the voters being able to detect malicious behaviour.

We now proceed by laying out the protocol utilising blockchain to solve these problems. To ensure an authority has a weak power we separate the tasks between a set of authorities. The untappable channel between voter and the authentication authority is achieved by onion routing. The privacy of the votes is guaranteed as long as one of the three mixes is honest.

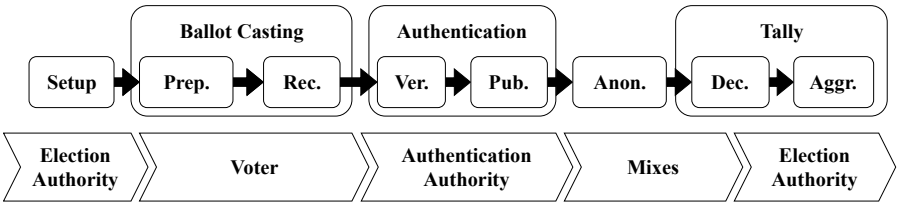

Figure 9: D4: Protocol Phases and Involved Actors

We have further refined the standard model of verifiable elections with the following phases as seen on Figure 9:

1. Election Setup,
2. Ballot Preparation & Recording,
3. Vote Verification & Publishing,
4. Anonymizing by Mixing,
5. Decryption of Votes, and Aggregation.

Ballot preparation and ballot recording have been merged into one phase. Furthermore, a phase of vote verification and weeding has been added before anonymisation. Anonymization and aggregation phases have been separated. Aggregation is included in the final step along with decryption of votes and announcement of results. The data that is recorded on blockchain is shown in Figure 20.

5.2 ELECTION SETUP

Each actor in the system is assumed to have a certified Elliptic Curve ElGamal public and private key. The public keys and the associated identities are thus known to any observer. Assume conventional cyclic group ElGamal is used for ballot encryption and everything else is done with an elliptic curve. Let pk_{ID} denote the public key of an actor identified by ID and sk_{ID} denote the secret key of an actor identified by ID.

GENERAL PARAMETERS Election Authority publishes the question of the election Q represented by a string using some standard encoding such as ASCII, UTF-8 or UTF-16 depending on the Election Authority's preference, the n options encoded as positive decimal integers $\{0, 1, ..., n-1\}$, the deadlines of the protocol phases 2.-4. (t_2, t_3, t_4) indicating the year, month, day, hour, minute, and second encoded as a string *YYYY-MM-DD hh:mm:ss* consisting of decimal digits, where Y indicates year and is 4 digits long, M indicates month and is 2 digits long, D indicates day and is 2 digits long, h indicates hour and is 2 digits long, m indicates minute and is 2 digits long, s indicates seconds and is 2 digits long. (e.g. 2018-04-25 00:00:00), and the time zone used in Coordinated Universal Time (e.g. UTC+0). The deadline here indicates the latest time and date by which a phase should be completed and further actions from that phase considered invalid.

MIXNET PARAMETERS Election Authority ($\mathcal{EA}$) sets up the ElGamal [25] parameters for the mixnet. It generates a description of a cyclic group G of order q with generator g. $\mathcal{EA}$ then publishes the description (G, q, g) on the blockchain. Furthermore, $\mathcal{EA}$ chooses a hash function H and publishes it on the blockchain.

Each Mix Authority M_i, where $1 \leqslant i \leqslant n_M$ executes the threshold ElGamal key generation algorithm to obtain a key pair $(x_{M_i}, y_{M_i}) \leftarrow \text{KeyGen}(\lambda)$. M_i then generates a Schnorr proof [70] for x_{M_i}, $\Pi_{DLog}(M_i) \leftarrow \text{GenProof}(x_{M_i}, y_{M_i}, H)$. Finally, M_i publishes the tuple $(\Pi_{DLog}(M_i), y_{M_i})$ on the blockchain as seen in Figure 10.

From: M_i	To: all
Hash of message	Timestamp
Signature of message	
commitment t	
challenge c	
response s	
public key y_{M_i}	

Figure 10: D4: Mixnet Setup Message

After all the mixes have published their public key shares, the $\mathcal{EA}$ verifies each Schnorr proof. If for all i, $\top \leftarrow \text{Verify}(\Pi_{DLog}(M_i))$, $\mathcal{EA}$ calculates the election public key $h_M \leftarrow \text{PubKeyGen}(y_{M_1}, ..., y_{M_{n_M}})$. Finally, $\mathcal{EA}$ publishes the election encryption key on the blockchain as (G, q, g, h_M) as seen in Figure 11.

Election encryption key transaction concludes the election setup. $\mathcal{EA}$ has published the question and options, the schedule of the election, and initialised the encryption parameters for the mix network

From: $\mathcal{EA}$	To: all
Hash of message	Timestamp
Signature of message	
Group ID	
prime p	
prime q	
generator g	
election encryption key h_M	

Figure 11: D4: Election Encryption Key Message

ElGamal cryptosystem. The Mix Authorities $\mathcal{M}$ have generated their shares of the key and proven knowledge of the secret key shares. $\mathcal{EA}$ then verified the proofs and combined the shares into one single election encryption key h_M.

5.3 BALLOT PREPARATION & RECORDING

In this phase, the voter chooses their options, creates a ballot, and publishes it on the blockchain.

5.3.1 *Ballot Preparation*

As the first step in the ballot preparation and recording phase, the voter prepares a ballot as seen in Algorithm 1. The process goes as follows:

1. Voter V_i chooses an option $v_i \in \{0, 1, ..., n-1\}$ for the question Q of the election they wish to participate in.
2. Voter V_i encrypts the vote with the mixnet public key as seen in Equation 39.

$$c_{v_i} = \text{Enc}(v_i, h_M) = (X_i, Y_i) \tag{39}$$

3. Voter V_i records the hash $t_{c_{v_i}}$ of the latest block on the blockchain, which is sufficiently buried.
4. Voter V_i constructs the ballot as a bit string concatenation of the hash and the encrypted vote as seen in Equation 40.

$$b_i = t_{c_{v_i}} \| c_{v_i} \tag{40}$$

5. Voter V_i signs the ballot with Digital Signature Algorithm [51] for eligibility verification as seen in Equation 41.

$$\sigma_{V_i} = \text{Sign}(b_i, H) = (r, s) \tag{41}$$

6. Voter V_i encrypts the tuple $(b_i, \sigma_{V_i}, ID_{V_i})$, which is encoded as a concatenation of the bitstrings, with Authentication Authority's public key for secrecy as seen in Equation 42.

$$c_{V_i} = \text{Enc}((b_i, \sigma_{V_i}, ID_{V_i}), pk_{AA}) \tag{42}$$

Algorithm 1 Preparing a ballot

```
1: function PrepareBallot(O, BC)
2:     v_i ← Choose(O)
3:     c_{v_i} ← Enc(v_i, h_M)
4:     t_{c_{v_i}} ← latest-hash(BC)
5:     b_i ← t_{c_{v_i}} || c_{v_i}
6:     σ_{V_i} ← Sign_{sk_{V_i}}(b_i, H)
7:     c_{V_i} ← Enc((b_i, σ_{V_i}, ID_{V_i}), pk_{AA})
8:     return c_{V_i}
9: end function
```

5.3.2 *Recording*

As the second step in the ballot preparation and recording phase, the voter publishes the ballot in a transaction as seen in Figure 12.

From: -	To: $\mathcal{EA}$
Hash of message	Timestamp
Nonce	Proof-of-Work
timestamp ts_i	
X_i	Y_i
r	s
ID_{V_i}	

Figure 12: D4: Voter Ballot Casting Transaction

1. Voter V_i creates an onion route and chooses j uniformly random nodes from the blockchain network N_1, ..., N_j.

2. Voter V_i encrypts in layered manner the ballot created in previous steps for the onion route.

3. Voter V_i sends the encrypted message through the onion route.

4. Each router will process messages as seen in Algorithm 2. If they are the last router, they will broadcast it to the blockchain network. Otherwise, they will send the message to the next router in the chain.

Algorithm 2 Process messages

```
1:  function ProcessMessage(m)
2:      (is − last, addr, m′) ← decrypt(m)
3:      if is − last then
4:          broadcast(m′)
5:          return True
6:      else
7:          send(addr, m′)
8:          return True
9:      end if
10:     return False
11: end function
```

5. After sending the ciphertext the voter V_i waits until predetermined t_w time, while scanning the network for c_{V_i}. Time t_w is chosen to be long enough so that message could pass through the onion route and be propagated to a sufficient degree so that voters can see it. Furthermore, if the vote has been observed propagating on the blockchain network, the voter needs to verify it is sufficiently buried (for example $n = 6$ blocks) to be satisfied that it has been recorded. The procedure for verifying the vote has been recorded is outlined in Algorithm 3.

Algorithm 3 Verify vote recorded-as-cast

```
1:  function VerifyRecorded(c_{v_i}, t_w, n)
2:      t ← time()
3:      while t < t_w do
4:          if scan-for(c_{v_i}) then
5:              wait(n)          ▷ Observe n-blocks to be mined after c_{v_i}.
6:              return True
7:          end if
8:          t ← time()
9:      end while
10:     return False
11: end function
```

5.4 VOTE VERIFICATION & PUBLISHING OF ELIGIBLE VOTES

Assuming the Authentication Authority is honest, this phase guarantees the eligibility of votes in the final tally. It is started when the deadline t_2 for vote casting is passed. It consists of two sub-phases

1. the collecting, verification and weeding of votes
2. and randomising, signing, and publishing of eligible votes.

5.4.1 *Collection, Verification & Weeding*

Authentication Authority ($\mathcal{AA}$) verifies votes and weeds out duplicates or non-eligible votes.

1. $\mathcal{AA}$ decrypts message c_{V_i} (as seen in Equation 43) on the blockchain in chronological sequence to obtain a tuple (ballot, signature, identifier).

$$\begin{aligned} m &= \text{Dec}(sk_{AA}, c_{V_i}) \\ &= (b_i, \sigma_{V_i}, ID_{V_i}) \end{aligned} \tag{43}$$

2. $\mathcal{AA}$ verifies the signature checks out σ_{V_i} (as seen in Equation 44) and ID_{V_i} is an eligible voter.

$$\top \stackrel{?}{=} \text{Verify}(\sigma_{V_i}) \tag{44}$$

3. $\mathcal{AA}$ records the vote c_{b_i} as seen in Algorithm 4.

Algorithm 4 Record vote

```
1: function RecordVote(V_i, c_{b_i}, votes)
2:     t ← time()
3:     if V_i ∈ votes then
4:         votes[V_i] ← votes[V_i] ∪ (t, c_{b_i})
5:     else
6:         votes[V_i] ← (t, votes[V_i])
7:     end if
8:     return votes
9: end function
```

5.4.1.1 *Publishing of Eligible Votes*

$\mathcal{AA}$ publishes re-encrypted votes and provides signatures and proofs for the voter and mix network.

1. $\mathcal{AA}$ randomizes vote by re-encrypting them as seen in Equation 45.

$$c'_{V_i} = \text{ReEnc}(c_{V_i}, h_M) \tag{45}$$

2. $\mathcal{AA}$ creates a designated verifier re-encryption proof [39] for voter V_i that the vote was re-encrypted as seen in Equation 46.

$$\sigma_{DVRP(V_i)} = \text{Sign}(c'_{V_i}, w_{V_i}, pk_{V_i}) \tag{46}$$

3. $\mathcal{AA}$ signs the re-encrypted vote with Digital Signature Algorithm [51] as seen in Equation 47.

$$\sigma_{DSA(c'_{V_i})} = \text{Sign}(c'_{V_i}, H) \tag{47}$$

4. $\mathcal{AA}$ generates a one-time public key [77] (P, R) for V_i as seen in Equation 48.

$$(P, R) = \text{OTPubKey}(pk_{V_i}) \tag{48}$$

5. $\mathcal{AA}$ encrypts the tuple $(c'_{V_i}, \sigma_{DSA(c'_{V_i})}, \sigma_{DVRP(V_i)})$ with voter V_i one-time public key (P, R) (as seen in Equation 49) and sends it to the stealth address of voter V_i as seen in Figure 13.

$$c_{\sigma_{AA}} = \text{Enc}((c'_{V_i}, \sigma_{DSA(c'_{V_i})}, \sigma_{DVRP(V_i)}), (P, R)) \tag{49}$$

6. $\mathcal{AA}$ sends publicly to the mixnet the tuple $(c'_{V_i}, \sigma_{DSA(c'_{V_i})})$ in a transaction as seen on Figure 14.

From: $\mathcal{AA}$	To: (P, R)
Hash of message	Timestamp
Re-encrypted Vote c'_{V_i}	
Signature of the Vote $\sigma_{DSA(c'_{V_i})}$	
Designated Verifier Re-encryption Proof σ_{DVRPV_i}	

Figure 13: D4: Sending Proof of Re-encryption to Voter Through Unlinkable Channel

From: $\mathcal{AA}$	To: $\mathcal{M}$
Hash of message	Timestamp
Re-encrypted Vote c'_{V_i}	
Signature of the Vote σ_{V_i}	

Figure 14: D4: Re-encrypted Vote Publishing Transaction

Algorithm 5 PublishEligibleBallot

function PUBLISHELIGIBLEBALLOT($c_{V_i}, pk_{V_i}, pk_M, sk_{AA}$)
  $c'_{V_i} \leftarrow \text{ReEnc}(c_{V_i}, pk_M)$
  $\sigma_{DVRP(V_i)} \leftarrow \text{Sign}(c'_{V_i}, w_{V_i}, pk_{V_i})$
  $\sigma_{DSA(c'_{V_i})} \leftarrow \text{Sign}(c'_{V_i}, H)$
  $(P, R) \leftarrow \text{OTPubKey}(pk_{V_i})$
  $c \leftarrow \text{Enc}((c'_{V_i}, \sigma_{DVRP(V_i)}, \sigma_{DSA(c'_{V_i})}), (P, R))$
  Send$((P, R), c)$
  Publish$(c'_{V_i}, \sigma_{DSA(c'_{V_i})})$
  return True
end function

5.5 ANONYMIZING BY MIXING

After deadline t_3 for publishing of authenticated votes has passed, the protocol proceeds to anonymisation phase. Mixing is done by employing Andrew Neff's Shuffles of ElGamal Pairs protocol [63, 64]. In principle, the protocol can support an arbitrary number of mixes. The mix count is a detail that is up to the implementation of the scheme. However, at least two are necessary to achieve the property that no single authority will learn the voter's votes.

1. Mix Authority M_1 collects the messages on the blockchain.
2. Mix Authority M_1 verifies the signatures on the messages as seen in Equation 50 and adds the valid votes $c'_{V_i} = (X_i, Y_i) =$ to a list of votes to be shuffled.

$$\text{Verify}(c'_{V_i}, \sigma_{DSA(c'_{V_i})}) \stackrel{?}{=} \top \tag{50}$$

3. Mix Authority M_1 generates fresh randomness β_i for all i and a new permutation $\pi \in \Sigma_i$ for the whole list of verified votes.
4. Mix Authority M_1 randomizes and shuffles the encrypted votes as seen in Equation 51.

$$(X_i(1), Y_i(1)) = (g^{\beta_{\pi(i)}} X_{\pi(i)}, h^{\beta_{\pi(i)}} Y_{\pi(i)}) \tag{51}$$

5. Mix Authority M_1 generates a proof of correct shuffle $\Pi_{Shuffle(1)}$.
6. Mix Authority M_1 publishes the randomized and shuffled votes $[(X_1(1), Y_1(1)), ..., (X_{n_{V_i}}(1), Y_{n_{V_i}}(1))]$ on the blockchain along with the proof $\Pi_{Shuffle(1)}$ as seen in Figure 15.
7. Mix Authority M_j generates fresh randomness β_j for all i and a new permutation $\pi \in \Sigma_i$ for the whole list of verified votes.
8. Mix Authority M_j randomizes and shuffles the encrypted votes as seen in Equation 52.

$$(X_i(j), Y_i(j)) = (g^{\beta_{\pi(i)}} X_{\pi(i)}(j-1), h^{\beta_{\pi(i)}} Y_{\pi(i)}(j-1)) \quad (52)$$

9. Mix Authority M_j generates a proof of correct shuffle $\Pi_{Shuffle(j)}$.
10. Mix Authority M_j publishes the randomized and shuffled votes $[(X_1(j), Y_1(j)), ..., (X_{n_{V_i}}(j), Y_{n_{V_i}}(j))]$ on the blockchain along with the proof $\Pi_{Shuffle(j)}$.
11. Steps 7.-10. are run until all the mixes have mixed the votes.

From: $\mathcal{M}_1$	To: $\mathcal{M}_2$
Hash of message	Timestamp
$X_1(1)$	$Y_1(1)$
⋮	
$X_{n_{V_i}}(1)$	$Y_{n_{V_i}}(1)$
$\Pi_{Shuffle(1)}$	

Figure 15: D4: Publishing Shuffled Votes and Proof

5.6 DECRYPTION OF VOTES, AGGREGATION AND RESULTS

In the final phase of the protocol, the votes $(X_i(n_M), Y_i(n_M))$ are decrypted, tallied, and results announced.

1. Each mix M_j computes for each vote i their decryption share as seen in Equation 53, proof that decryption share is correct as seen in Equation 54, and publishes them in transaction as seen in Figure 16.

$$d_{i,j} = \text{DecShr}(X_i(n_M), x_{M_j}) \quad (53)$$

$$\Pi_{\text{DLogEq}_{i,j}} = \text{GenProof}(x_{M_j}, y_{M_j}, g, d_{i,j}, X_i(n_M), H) \quad (54)$$

From: $\mathcal{M}_i$	To: $\mathcal{EA}$
Hash of message	Timestamp
Ballot #j	
Decryption share $d_{i,j}$	
Proof of decryption $\Pi_{\text{DLogEq}_{i,j}}$	

Figure 16: D4: Decryption Share Publishing

2. $\mathcal{EA}$ verifies discrete logarithm equality proofs, and if it accepts then combines decryption shares $[d_{i,1}, ..., d_{i,n_M}]$ for vote i and decrypts the vote as seen in Equation 55 publishing it in a transaction as seen in Figure 17.

$$v_i = \text{Dec}((X_i(n_M), Y_i(n_M)), [d_{i,1}, ..., d_{i,n_M}]) \tag{55}$$

From: $\mathcal{EA}$	To: -
Hash of message	Timestamp
Hash of $d_{i,1}$	
⋮	
Hash of d_{i,n_M}	
Decrypted Vote v_i	

Figure 17: D4: Decrypted Vote Transaction

3. $\mathcal{EA}$ tallies and publishes the election results on the blockchain in a transaction as seen in Figure 18.

From: $\mathcal{EA}$	To: -
Hash of message	Timestamp
Option 1	Count(1)
⋮	
Option n	Count(n)

Figure 18: D4: Results Transaction

The last phase concludes the description of the D4 protocol. We proceed by analysing the scheme theoretically in Chapter 6 and validating it practically in Chapter 7.

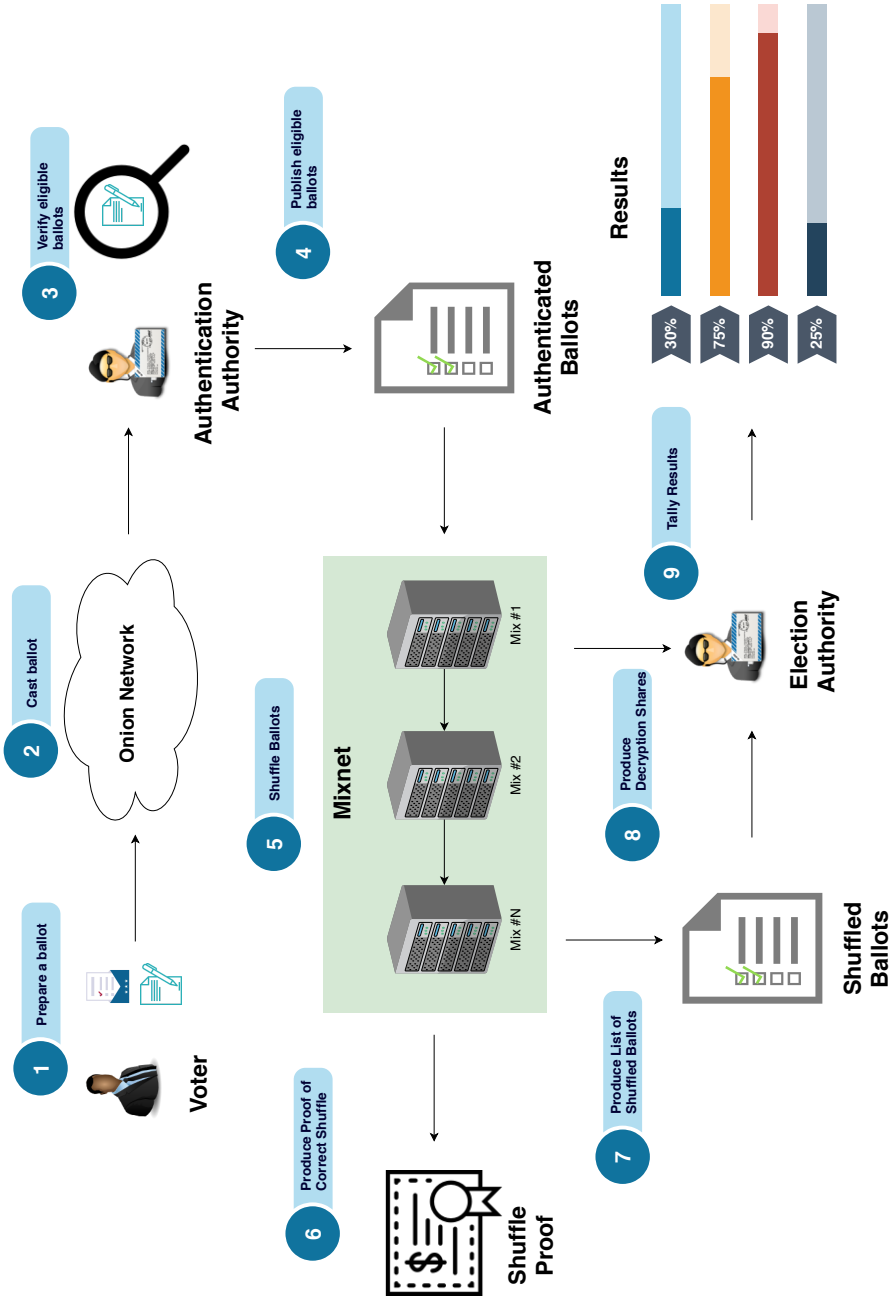

Figure 19: D4: System Schematic

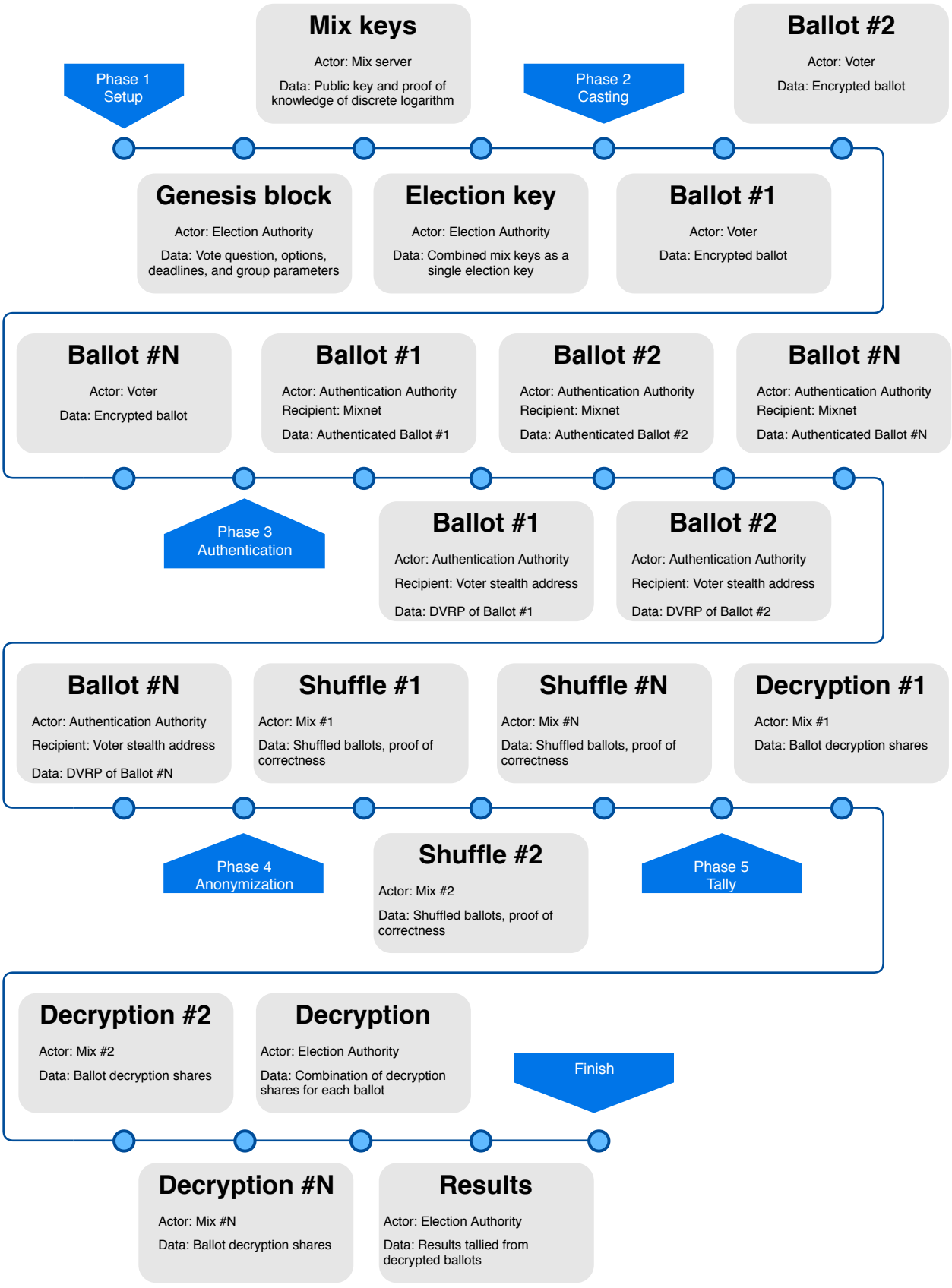

Figure 20: D4: Data Recorded on the Blockchain

6 ANALYSIS

We proposed a new scheme in Chapter 5, and in this chapter, we analyse the scheme from a theoretical perspective in two dimensions. We begin by going through the security aspects of privacy and verifiability in Section 6.1 and finish by analysing the complexity of the scheme from communication and computational aspects in Section 6.2.

6.1 SECURITY

The proposed electronic voting protocol is secure if it satisfies the security properties related to privacy and verifiability, and the auxiliary properties enlisted in Section 4.3. In this section, we analyse the protocol's security concerning these properties one by one focusing on privacy and verifiability.

6.1.1 *Privacy*

A voting protocol ensures privacy if the voter's choice remains anonymous. Thus the attempts to influence or buy votes is hindered. To this end, we have attempted to satisfy *ballot privacy* and *receipt freeness*.

BALLOT PRIVACY The proof that the property of *ballot privacy* as defined earlier is satisfied hinges on the security of the underlying encryption scheme used to encrypt the vote in ballot casting phase. Since after that step until mixing, only independent fresh randomness is added to the ballot on the blockchain, it is trivial to see that it does not leak any information about the underlying vote. The mixing procedure breaks the link between the input and output votes. The correctness and privacy of mixing are proven by the respective papers in [63, 64]. Hence nothing can be learned of the ballot with greater than negligible probability after it has been decrypted.

Definition 1. *A voting scheme has ballot privacy if no adversary can do better in the ballot distinguishing Experiment* $B - IND$ *than guess at random.*

Experiment $B - IND$

1 : $(pk_M, [sk_1, sk_2, ..., sk_{n_M}]) \leftarrow Setup(\lambda)$
2 : $(v_0, v_1) \leftarrow \mathcal{A}(\lambda, pk_M)$
3 : $\beta \leftarrow \{0, 1\}; b \leftarrow Vote(pk_M, v_\beta)$
4 : $\beta' \leftarrow \mathcal{A}(pk_M, b)$
5 : $\beta = \beta'$

In the Experiment $B - IND$ the challenger flips a coin, the adversary provides two votes, challenger finally casts a ballot for one of the votes. The adversary wins if they can distinguish which of the two votes was cast.

Lemma 1 (Ballot privacy)**.** *The protocol as described in Chapter 5 provides ballot privacy given that a PPT adversary $\mathcal{A}$ is unable to distinguish between two ballots encapsulating different votes in a chosen-plaintext attack (CPA) model as described in the Experiment* $B - IND$.

Proof. The adversary has access to an oracle for creating a ballot for a left or right vote. Also, an oracle for creating any ballot can be called at any time before or after creating a left or right ballot. Ballot creation is essentially ElGamal encryption of a vote. Therefore, it is trivial to see that the security of ElGamal encryption *IND-CPA* follows. *IND-CPA* is sufficient for ballot privacy in the scheme. □

Remark 1. *For ballot privacy it is not necessary to have IND-CCA security as the protocol does not provide a decryption oracle for any data that can be linked to an individual. After the mixing phase, the ciphertexts will be decrypted. These will be unlinkable to the input ciphertexts as proven in [63, 64]. Therefore, the decrypted votes are unlinkable to the ballots input for mixing.*

RECEIPT FREENESS After the voting ends, the voter cannot compile a receipt to prove to a coercer, how they voted. Recall that a receipt has to satisfy three conditions: the voter generates it, it shows how the voter voted, and it proves the ballot was cast [41].

Definition 2. *A voting scheme is receipt free if no adversary can do better in the receipt distinguishing Experiment* $R - IND$ *than guess at random.*

Experiment $R - IND$

1 : $(pk_M, [sk_1, sk_2, ..., sk_{n_M}]) \leftarrow Setup(\lambda)$
2 : $(v, r) \leftarrow \mathcal{A}(\lambda, pk_M)$
3 : $b \leftarrow Vote(pk_M, v, r)$
4 : $(b', \delta_0, \sigma(b')) \leftarrow Valid(b)$
5 : $\delta_1 \leftarrow SimProof(b', b, \delta_0, \bar{v}, \bar{r})$
6 : $\beta \leftarrow \{0, 1\};$
7 : $\beta' \leftarrow \mathcal{A}(pk_M, \delta_\beta)$
8 : $\beta = \beta'$

In $R-IND$ the adversary has the power to choose a vote for the challenger, effectively also dictating the randomness r used in the ElGamal encryption. The challenger then prepares a ballot and casts it, getting a re-encrypted ballot b', designated verifier re-encryption proof δ_0 and the authentication authorities signature on the re-encrypted ballot $\sigma(b')$. The challenger now simulates a proof for a different vote $(\bar{v}, \bar{r}) \neq (v, r)$ and obtains a proof δ_1. The challenger then flips a coin β and submits δ_β as a receipt to the adversary. The adversary must now determine whether the proof is simulated or not, and wins the game if they can do better than guessing at random.

Lemma 2 (Receipt freeness)**.** *The protocol as described in Chapter 5 provides receipt-freeness given that a PPT adversary $\mathcal{A}$ is unable to distinguish between two receipts proving the vote of different options as described in Experiment* $R-IND$.

Proof. This property is achieved in our scheme through designated verifier re-encryption proofs. Further, recall Assumption 7 the coercer does not own the voter identity in the sense of knowing their private keys. Otherwise, the coercer could participate in the voting as a voter in any phase of the protocol.

The voter publishes their encrypted vote under their ephemeral pseudonym on the blockchain through an anonymous channel provided by the onion routing. The voter can reveal the pseudonym and the plaintext ballot. In the Experiment $R-IND$ this is emulated by the adversary providing the vote and randomness for a ballot to the challenger. It reveals the intention of the voter to the coercer. An unwilling voter can thus submit a vote to the liking of the coercer and later submit their own vote, thus protecting against coercion in this stage.

The Authentication Authority verifies the vote and sends the voter back designated verifier re-encryption proof that it has been recorded. This proof in of itself cannot satisfy the coercer, as the voter can simulate the proof for any ballot. The eligible votes on the blockchain are randomised by the Authentication Authority, where the randomisation is kept secret from the voter. Furthermore, the use of voter stealth address can provide voter deniability of receiving any confirmation of recording. Thus we conclude, the receipt-freeness is satisfied, with the assumption that the coercer does not have ownership of the voter's secrets and the voter is not willingly cooperating with the coercer.

Therefore, the receipt-freeness property relies on the security of the designated verifier re-encryption proof [39, 56]. The security was analysed in [57], where a flaw was identified and a solution proposed. Thus we conclude the protocol achieves the receipt-freeness property. □

Lemma 3 (Private)**.** *The described protocol ensures privacy of the voters.*

Proof. The proof follows from the proofs of the privacy properties:

- ballot privacy,
- receipt freeness.

□

6.1.2 *Verifiability*

The voting protocol is verifiable if any observer can be convinced without a doubt that the procedure has been followed correctly. This property is ensured in the described protocol by satisfying the verifiability requirements enlisted above. We now proceed to argue they have been satisfied.

INDIVIDUAL VERIFIABILITY Voter is satisfied their vote is in the set of all the eligible votes.

Proof. The protocol provides overwhelming evidence for a voter. Should their vote ever be excluded they can dispute the results by showing it was recorded on the blockchain, but not in the results. The proof for a voter consists of:

- Observation of the ciphertext sent to Authentication Authority.
- Observation of the designated verifier re-encryption proof and the re-encrypted ballot on the blockchain.
- Confidence that $\mathcal{AA}$ is not in possession of the voter's secret key, nor that the key has been leaked to anyone else.

Thus we conclude the protocol has this property. □

UNIVERSAL VERIFIABILITY Final tally corresponds to the set of all eligible votes.

Proof. Assuming $\mathcal{AA}$ follows the protocol we are convinced that all eligible votes are published for mixing without discrimination. Given the voter's follow the protocol honestly and verify their votes have been recorded on the blockchain, we conclude any single vote not included in the records will be recognised. The shuffle protocol we use was proven secure in the original papers [63, 64]. Thus we conclude the universal verifiability is satisfied with overwhelming probability due to the public inputs, outputs and proof from mix network. □

END-TO-END VERIFIABLE The voting protocol is end-to-end verifiable given that a voter can verify their vote was cast-as-intended, recorded-as-cast, and tallied-as-recorded.

CAST-AS-INTENDED The ballot correctly encapsulates the vote of the voter.

Proof. In our scheme, the voter constructs the ballot without the help of anyone else. Assuming the user software and hardware is trusted, we conclude the ballots are cast-as-intended. □

RECORDED-AS-CAST The ballot is recorded as cast.

Proof. In our scheme, the voter verifies the ballot is published on the blockchain and is satisfied that it has been recorded as seen in Algorithm 3. Furthermore, the Authentication Authority provides proof that the vote was randomised and included in the set of votes to be mixed. Given the authority cannot forge a proof, we conclude the ballot was recorded-as-cast. □

TALLIED-AS-RECORDED The ballot recorded counts in the final tally.

Proof. In our scheme, the mix network provides overwhelming proof of correct shuffle. Thus when the ballots are finally decrypted and counted the voter is satisfied their vote was tallied. Thus we conclude the ballot was tallied-as-recorded. □

Lemma 4 (Verifiable)**.** *The described protocol is verifiable by anyone.*

Proof. The proof follows from the proofs of following verifiability properties:

- individual verifiability,
- universal verifiability,
- end-to-end verifiability.

□

Beyond the requirements explained we elicited auxiliary security requirements independent of the privacy and verifiability:

- fairness,
- eligibility,
- correctness,
- ballot independence,
- and accessibility and visibility of blockchain.

FAIRNESS The tally is unavailable during the casting phase.

Proof. In our scheme, fairness is achieved by the separation of phases. When the ballots are cast everything is encrypted, and there is no way to tally the results. Tallying is done in the final phase when the casting has already ended. Thus we conclude the fairness requirement is satisfied as long as PPT adversary $\mathcal{A}$ is unable to break the ElGamal encryption. □

ELIGIBILITY Only eligible voters can vote, and the votes must be for one of the published options.

Proof. The Authentication Authority establishes the eligibility of the voters. Assuming the authority follows the protocol honestly the eligibility of voters is satisfied. In the final phase, the ballots are decrypted, and correctly formed ballots are counted meaning that any non-eligible ballot is discarded. Anyone can observe the decryption shares for votes and tally the results independently. Thus we conclude the eligibility requirement is satisfied. □

CORRECTNESS Honest execution of the voting process yields results according to the cast ballots.

Definition 3. *The described protocol is correct if the final output after honest execution of the algorithms is true.*

Let $v_1, ..., v_i \in \mathbb{V}$ *be the set of valid votes and* $id_1, ..., id_i \in \mathbb{ID}$ *be the set of voter identifiers for* $1 \leqslant i \leqslant n_V$ *and the set of randomness* $r_1, r_2, ..., r_i \in \mathbb{Z}_q$ *used by the voters to create a ballot.*

The protocol is initialized by $(pk, [sk_1, ..., sk_{n_M}]) \leftarrow Setup(\lambda)$. *Honest votes are cast by* $b_i \leftarrow Vote(id_i, v_i, r_i)$ *for all* i. *Eligible ballots are then mined on the network* $(b'_i, \delta_i, \sigma(b'_i)) \leftarrow Valid(b_i)$ *for all* i. *Mixes anonymize the eligible ballots* $(\mathcal{B}', \Pi) \leftarrow Mix(\mathcal{B})$. *Tally is computed as* $(\mathbb{O}) \leftarrow Tally(\mathcal{B}', [sk_1, ..., sk_{n_M}])$. *Then on honest execution the Verify algorithm should return true as* $Verify(\mathcal{BC}, \mathbb{O}, \Pi) = \top$.

We are assuming the adversary is semi-honest, thus follows the protocol correctly. Therefore, it is sufficient to see that the protocol can be executed correctly.

Proof. The consensus mechanism of the blockchain ensures the integrity of the data on blockchain as an invalid transaction is not recorded. Furthermore, the cryptographic proofs are publicly verifiable meaning that anyone can audit them. As the adversary is assumed to be semi-honest, they follow the protocol correctly. Thus we conclude, the correctness is satisfied. □

BALLOT INDEPENDECE An adversary cannot record a ballot sent by a voter and later replay it to change the choice of the voter.

Proof. This property is achieved by including the latest hash of the block on the blockchain as observed by the voter in the ballot. The hash acts effectively as *carbon dating* that the adversary cannot forge. As such, the ballots are non-replayable. Furthermore, the adversary cannot ask the voter to create a ballot to be sent in the last minute before casting phase ends. Thus we conclude the ballot independence is satisfied. □

PUBLIC AND PERMISSIONED The blockchain is public for observation and permissioned to actors for appending.

Proof. Access to blockchain requires that anyone can view it and certified actors may be able to append to it. This property is achieved by making the blockchain public and permissioned. It ensures anyone can view it and only certified actors actions are considered in the consensus mechanism by other nodes in the blockchain. Thus we conclude the access and visibility of blockchain is as specified. □

6.2 COMPLEXITY

In this section, we will analyse the protocol complexity based on two dimensions communication and computation. The complexity depends on several variables listed in Table 4. The analysis is done gradually by phases and for each actor. Recall the actors in the system are Election Authority ($\mathcal{EA}$), Authentication Authority ($\mathcal{AA}$), voters ($\mathcal{V}$), mixes ($\mathcal{M}$), and blockchain nodes ($\mathcal{N}$). The complexity analysis assumes the correct execution of the protocol. The tables reflect the complexity for individual actor. Thus for the total cost, the complexities are linear to the number of actors of that type.

Symbol	Description
$\mathcal{V}$	Number of voters
$\mathcal{M}$	Number of mixes
$\mathcal{N}$	Number of routers in onion route

Table 4: D4: Complexity Analysis Symbols

6.2.1 *Computation*

For the computational complexity, we analyse how the number of actors affects the cost of computations asymptotically. Summary of the analysis can be seen in Table 5. The most costly phase is the decryption of the ballots after they have been anonymised, which is linear to the number of two actors the voters and mixes. Overall, the process is

linear in all phases and should scale well in practice. In the following, each phase is analysed in detail.

1. **Setup**: $\mathcal{EA}$ generates the election parameters and has to combine the public keys of $\mathcal{M}$ by multiplication, thus the complexity is $\mathcal{O}(1)$. Each mix has to generate a random secret key, compute the public key which is one exponentiation and Schnorr proof which involves one exponentiation. The setup computation complexity for a particular mix is thus constant $\mathcal{O}(1)$.

2. **Ballot Preparation & Recording**: To prepare a ballot the voter has to execute 2 ElGamal encryptions and generate a single DSA signature. Ballot preparation is thus constant for the voter. To record the ballot the voter needs to do $\mathcal{N}$ ElGamal encryptions. Thus ballot recording for the voter is linear to the number of onion routers used.

3. **Verification & Publishing**: To verify a ballot of a single voter the $\mathcal{AA}$ has to do an ElGamal decryption, DSA signature verification, and record the vote. Thus the verification process is linear to the number of voters. In order to publish a ballot the $\mathcal{AA}$ has to do a re-encryption of the ballot, generate a DVRP for the ballot, generate a DSA signature, generate a one-time public key for the voter, and execute an ElGamal encryption. Thus the computational complexity for publishing of the submitted ballots for the $\mathcal{AA}$ is linear to the voter.

4. **Anonymization**: To anonymize all the ballots each mix has to for each voter verify the DSA signature of the $\mathcal{AA}$ requiring 2 exponentiations, generate a proof of correct shuffle requiring a total of $8|\mathcal{V}|+4$ exponentiations, and for the shuffle itself execute a total of $2|\mathcal{V}|$ exponentiations. Thus the anonymisation is linear to the number of voters $\mathcal{O}(\mathcal{V})$

5. **Decryption & Results**: To decrypt each mix will execute a single ElGamal decryption share generation for each ballot requiring single exponentiation and Chaum-Pedersen discrete logarithm equality proof which requires further two exponentiations. Decryption is thus linear to the number of mixes and voters. Further then, the $\mathcal{EA}$ will multiply together all the ElGamal decryption shares of each voter, finally adding together $\mathcal{V}$ votes. This step is thus linear to the number of mixes and the number of voters $\mathcal{O}(\mathcal{MV})$.

6.2.2 *Communication*

All the communication in the protocol is done through the blockchain and the network. Thus we analyse the number of messages sent thro-

		Casting		Auth.				
Actor	Setup	Prep.	Rec.	Ver.	Pub.	Anon.	Dec.	Tally
$\mathcal{EA}$	$\mathcal{O}(1)$						$\mathcal{O}(\mathcal{VM})$	$\mathcal{O}(\mathcal{V})$
$\mathcal{AA}$				$\mathcal{O}(\mathcal{V})$	$\mathcal{O}(\mathcal{V})$			
$\mathcal{V}$		$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{N})$					
$\mathcal{M}$	$\mathcal{O}(1)$					$\mathcal{O}(\mathcal{V})$	$\mathcal{O}(\mathcal{V})$	

Table 5: D4: Computational complexity

ugh the network and on the blockchain. We assume each participant is also part of the blockchain network and once the messages are published everyone has them. Summary of the analysis can be found in Table 6.

1. **Setup**: $\mathcal{EA}$ generates the election parameters and publishes them once which takes constant time $\mathcal{O}(1)$. Each mix of $\mathcal{M}$ generates a key pair and Schnorr proof which can be sent in one message. Thus this step is linear to the number of mix servers $\mathcal{O}(\mathcal{M})$. Finally, $\mathcal{EA}$ collects the public key shares, computes the election public key and publishes it once which takes constant time $\mathcal{O}(1)$. In total, for *Setup* we have then $\mathcal{O}(\mathcal{M}+2)$.

2. **Ballot preparation & recording**: Ballot preparation does not require any communication on behalf of the voter. Once the voter has prepared a ballot, they send it through the onion route to be mined on the blockchain. For $\mathcal{N}$ nodes and $\mathcal{V}$ voters this phase then requires $\mathcal{O}(\mathcal{V}(\mathcal{N}+1))$ messages.

3. **Vote verification & publishing**: Verification is done non-interactively and thus requires no communication from the $\mathcal{AA}$. $\mathcal{AA}$ has to generate a proof of re-encryption for each ballot and a re-encrypted ciphertext for input to the mix network, which is broadcast to the network. In total this then requires $\mathcal{O}(2\mathcal{V})$ messages.

4. **Anonymization**: Each mix server will receive the ballots, shuffle them and publish a single proof over the entire shuffle. Thus the shuffled votes are linear to the number of votes and mix servers as in $\mathcal{O}(\mathcal{VM})$. Furthermore, each mix publishes a proof of correct shuffle $\mathcal{O}(\mathcal{M}(9\mathcal{V}+10))$. In total *Anonymization* then requires communication linear to the number of mixes and voters as in $\mathcal{O}(10\mathcal{M}(\mathcal{V}+1))$.

5. **Decryption & results**: Each mix publishes for each vote a decryption share $\mathcal{O}(\mathcal{MV})$. $\mathcal{EA}$ combines the decryption shares for

each vote and publishes them $\mathcal{V}$. Thus the total communication for decryption and results is $\mathcal{O}(\mathcal{V}(\mathcal{M}+1))$.

Considering all the step, the total communication complexity for the entire protocol is thus $\mathcal{O}(11\mathcal{V}\mathcal{M}+\mathcal{V}\mathcal{N}+3\mathcal{V}+\mathcal{M}+4)$.

		Casting		Auth.				
Actor	Setup	Prep.	Rec.	Ver.	Pub.	Anon.	Dec.	Tally
$\mathcal{EA}$	$\mathcal{O}(1)$							$\mathcal{O}(\mathcal{V})$
$\mathcal{AA}$					$\mathcal{O}(\mathcal{V})$			
$\mathcal{V}$			$\mathcal{O}(\mathcal{N})$					
$\mathcal{M}$	$\mathcal{O}(1)$					$\mathcal{O}(\mathcal{V})$	$\mathcal{O}(\mathcal{V})$	

Table 6: D4: Communication complexity

7 VALIDATION

In Chapter 5 we described the design and in Chapter 6 analysed the protocol theoretically from security and complexity dimensions. In this chapter we describe the practical work done to validate the designed protocol.

First, we describe the implementation of the protocol in Section 7.1. The description is followed by how the system was instantiated in Section 7.2. Then we outline the things we want to measure in experiments Section 7.3. The results are presented in Section 7.4 and finally scalability is discussed and future work outline in Section 7.5.

7.1 IMPLEMENTATION

In order to demonstrate the feasibility of the design, a proof of concept was implemented. For rapid development Python 3.6.5[1] was used. Python was chosen not because of efficiency but because of the ample availability of libraries which implement different concepts that were used.

The implementation makes use of some Python libraries amongst which the most relevant from cryptographic perspective are in no particular order:

- eth-keys[2],
- cryptography.io[3],
- cryptography-hazmat[4],
- pycryptodomex[5],
- ecies[6].

1 Python programming language https://www.python.org/.
2 Eth-keys library provides functionality to operate on Elliptic Curve secp256k1 keys, available from https://pypi.org/project/eth-keys/.
3 Cryptography.io library provides high-level functionality for symmetrical and asymmterical cryptography, available from https://pypi.org/project/cryptography/.
4 Cryptography-hazmat is part of the cryptography.io library providing low-level functionality referred to as *hazardous materials* due to the expertise required.
5 PyCryptodome is a package for low-level cryptographic primitives, available from https://pypi.org/project/pycryptodomex/.
6 Ecies is a package for Elliptic Curve Integrated Encryption Scheme for secp256k1 curve, available from https://pypi.org/project/eciespy/.

In the first phase of the implementation, all the primitives necessary were implemented in Jupyter Notebook[7]. Notable primitives that were implemented included:

- Designated Verifier Re-encryption Proof [39],
- ElGamal (n, n)-threshold cryptosystem [25],
- Schnorr Proof of Knowledge of Discrete Logarithm [70],
- Stealth Addresses [77],
- Shuffles of ElGamal Pairs [63, 64].

The primitives we borrowed from libraries and did not have to implement are listed below:

- ElGamal cryptosystem;
- Elliptic Curve Integrated Encryption System,
- SHA-256 Cryptographic Hash,
- Elliptic Curve Digital Signature Algorithm,
- Elliptic Curve Cryptography.

The second phase of implementation involved integrating the primitives into the five phases of the design:

1. Setup,
2. Casting,
3. Authentication,
4. Anonymization,
5. Tally.

Once the implementation was complete, we identified the processes we wanted to measure for storage, and time required.

7.2 INSTANTIATION

Several decisions were made regarding the configuration variables. Where possible, safe standards were used[8]. To show that the implementation works the key size is not essential. However, to demonstrate that the design is still feasible in realistic scenario safe parameters regarding length should be chosen.

7 Jupyter Notebook homepage http://jupyter.org/.
8 Cryptographic Key Length Recommendation found at https://www.keylength.com.

For the Diffie-Hellmann the guides recommend a group with 3072-bits and the key length of 256-bits. Thus, we chose Group 15[9] a 3072-bit MODP Group as seen in Table 7.

Parameter	Value
p	$2^{3072} - 2^{3008} - 1 + 2^{64} * 2^{2942}\pi + 1690314$
g	2

Table 7: Group 15 - 3072-bit MODP Group

For near future security, the guides suggest 256-bits for Elliptic Curve Cryptography. Thus curve secp256k1[10] as seen in Table 8 was chosen for Elliptic Curve Cryptography. The curve is not regarded as entirely safe by SafeCurves project[11]. SafeCurves has found that it has some weaknesses which lessen the security by marginal bits, there is still no attack identified which completely breaks the security. Furthermore, for performance, it is comparable to other curves with similar key sizes. Additionally, as it is used for numerous projects in cryptocurrencies and is thus widely implemented by libraries, it was chosen.

Parameter	Value
Curve	$y^2 = x^3 + 0x + 7$
p	$2^{256} - 2^{32} - 977$

Table 8: Elliptic Curve secp256k1

7.3 EXPERIMENTS

As we are interested in testing the feasibility of the design on consumer hardware, the experiments were run on a workstation. The experiment computer specification can be seen in Table 9.

To validate the feasibility we are interested in two characteristics the amount of data that needs to be handled by the network and stored locally, and the time processes take to execute by individual actors.

9 More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) found at https://www.ietf.org/rfc/rfc3526.txt.

10 SEC 2: Recommended Elliptic Curve Domain Parameters found at http://www.secg.org/SEC2-Ver-1.0.pdf.

11 SafeCurves: choosing safe curves for elliptic-curve cryptography (https://safecurves.cr.yp.to/).

Component	Specification
CPU	Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz
Memory	2x 4GB PC3-14900R DDR3-1866
Hard drive	500 GB SATA 3.0 7200 RPM

Table 9: Experiment hardware

SETUP In the setup phase we have two kinds of active actors namely the Election Authority and Mix. Here we want to measure the time it takes for the mix network to generate the Schnorr proof and how much data the proof requires. Secondly, we want to validate the amount of time it takes for Election Authority to verify the proofs and combine them as the election key.

CASTING In the casting phase voters are the active actors. Here we want to measure the time it takes to cast a ballot and the size of the data required for a ballot.

AUTHENTICATION In the authentication phase the Authentication Authority collects the submitted ballots, re-encrypts them, generates a designated verifier re-encryption proof, stealth address for the voter, and encrypts it for sending. Here we want to measure the time taken to execute these individual steps atomically and the size of the data required to store re-encrypted ballot and encryption of the designated verifier re-encryption proof.

ANONYMIZATION In the anonymisation phase the mix network collects ballots, shuffles them, generates a proof, and publishes the shuffled ballots and proofs. Here we want to measure how much time it takes for the mix network to shuffle the ballots and the size of the proof relative to the number of voters. The shuffled proofs are constant in size to the input.

TALLY In the tally phase, the mixes and Election Authority jointly decrypt the shuffled ballots by producing decryption shares for each ballot, and then combining the decryption shares. Here we want to measure, the size of data required to store all the decryption shares and the time taken to execute the decryptions and verify the shuffle proofs relative to the number of voters.

We can calculate the amount of data that needs to be handled by the network by estimating the amount of data that needs to be stored on the blockchain. Mostly, we are interested in how the operations scale relative to the number of voters.

7.4 RESULTS

Experiments were run using Python *timeit* functionality. Measurements were run for repetitions of batches of 10-100 in steps of 10, and 1000-10000 in steps of 1000 voters. As the assumption is that there is already an existing public key infrastructure no measurements were taken regarding key generation except for the election key generation. Therefore, we also generated the keys that were necessary and used static keys for the measurements. We used the mean values for analysis, as there can be significant discrepancies in timings, due to other activities the CPU is involved.

7.4.1 *Time Performance*

Phase	Actor	Procedure	Median Time (seconds)
Setup	$\mathcal{M}$	Key Generation and Schnorr Proof	0.36736
Setup	$\mathcal{EA}$	Public Key Combination	0.18614
Casting	$\mathcal{V}$	Casting a Vote	0.02011
Authentication	$\mathcal{AA}$	Authenticating and Publishing	0.18955
Anonymization	$\mathcal{M}$	Shuffle and Proof	0.67452
Anonymization	$\mathcal{EA}$	Verifying Proof	0.53820
Tally	$\mathcal{M}$	Creating a Decryption Share	0.00702
Tally	$\mathcal{M}$	Proof of Decryption	0.14474
Tally	$\mathcal{EA}$	Decryption	0.00153

Table 10: D4: Proof-of-Concept Implementation Run-time per Voter/Ballot

To measure time performance we cast in total 55550 ballots for measurement. The overall results of each operation per ballot/voter can be seen in Table 10. Comparison of the median operation times can be seen in Figure 21.

SETUP In this phase we measured the time it takes for a mix to generate a key and Schnorr proof of knowledge of the discrete logarithm of the public key. We used three mixes for each iteration, and the process was repeated 1000 times. The median time to generate

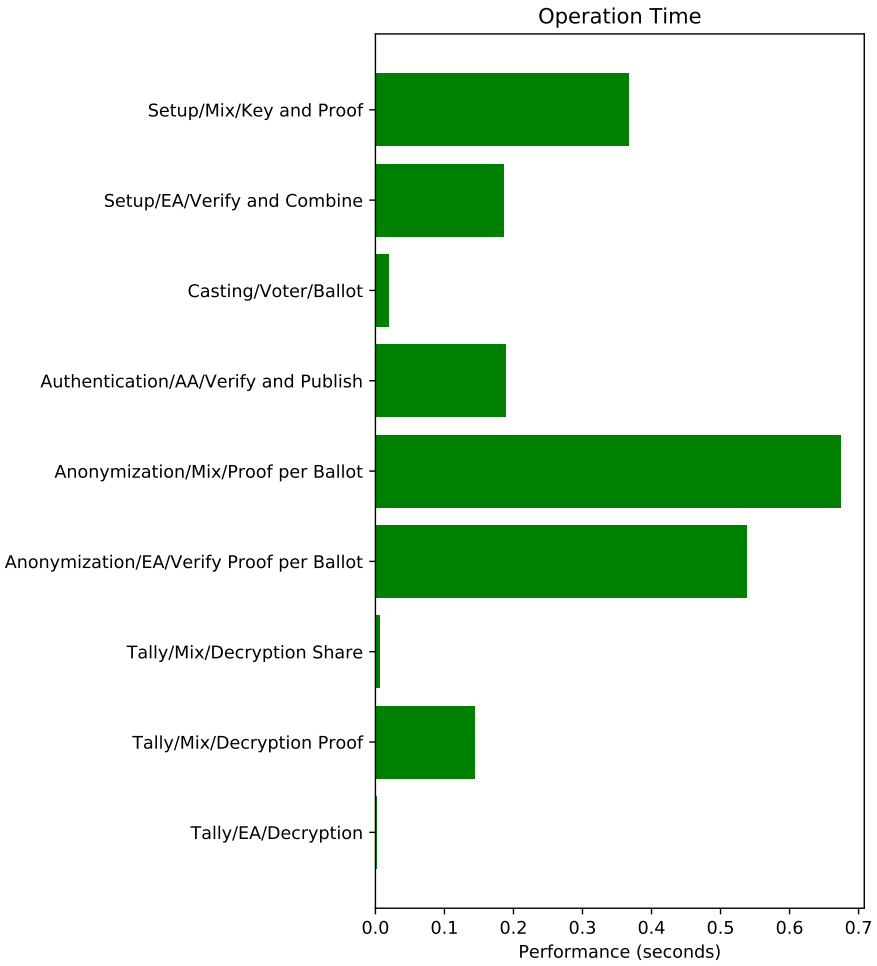

Figure 21: D4: Operation Times

a key and Schnorr proof for three mixes is 0.36736033099987253 seconds as seen in Figure 21. Then we measured the time it takes for the Election Authority to verify the proofs and combine the public key shares into a single election key. The median time to do that was 0.1861420570021437 seconds as seen in Figure 21. Given the security parameters, the setup computation heavy parts take in total 0.5535023880020162 seconds.

BALLOT CASTING In this phase we measured the time it takes for a voter to cast a ballot. We assumed the voter has their public infrastructure keys and has made up a choice. Thus we pre-generated these values and then measured the time it takes to cast the ballot based on those parameters. Over 55550 cast ballots the median time was 0.020106587500777096 seconds as seen in Figure 21.

AUTHENTICATION In this phase we measured the time it takes for the Authentication Authority to verify and publish ballots. We assumed the voters had all participated honestly. Thus we reused the

output from the previous phase. Over 55550 cast ballots the median time of authentication processing was 0.18954966250021243 seconds as seen in Figure 21.

ANONYMIZATION In this phase we measured the time it takes for the mixes to shuffle and generate a proof. We reused the output from the previous phase as an input to the anonymisation phase. The results of anonymization experiments on a logarithmic scale can be seen in Figure 22 and the verification of the shuffle in Figure 23. The comparison of proof generation and verification can be seen in Figure 24. As expected, the process is linear to the number of voters. The process works on a set of ballots as a whole. Therefore we could not measure times for individual ballots but rather calculate the average over a batch of ballots. On average it takes 0.674519313820274 seconds to shuffle and generate a proof for a ballot. Similarly, the verification works over a set of ballots and a proof for the shuffle. On average it takes 0.5381966577999264 seconds to verify proof per ballot.

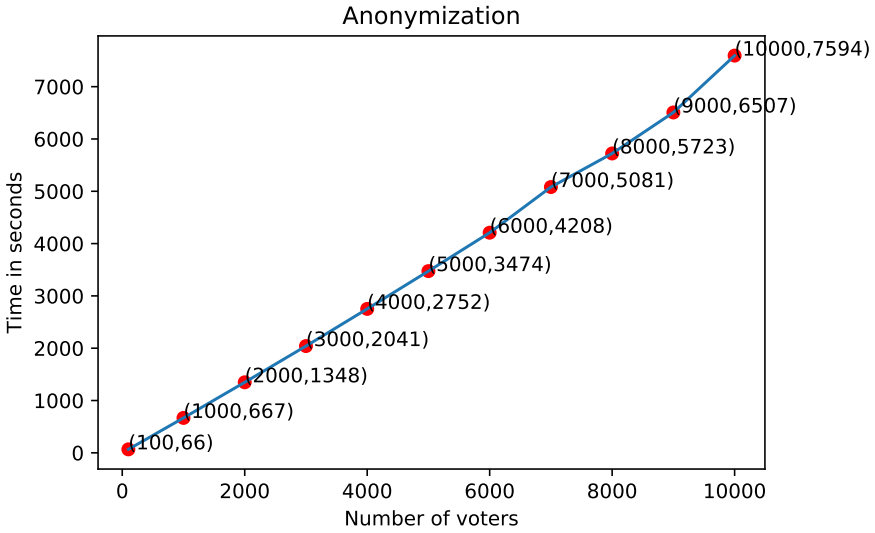

Figure 22: D4: Experiment Times of Anonymizing Ballots

DECRYPTION & TALLY In this phase we measured the time it takes for the mixes to create decryption shares and for the election authority to decrypt the ballot based on the decryption shares. The median time to create a decryption share is 0.007020432996796444 seconds and to decrypt a ballot is 0.001525457002571784 seconds as seen in Figure 25. The tally from decrypted ballots is simple counting and negligible compared to the rest of the operations measured.

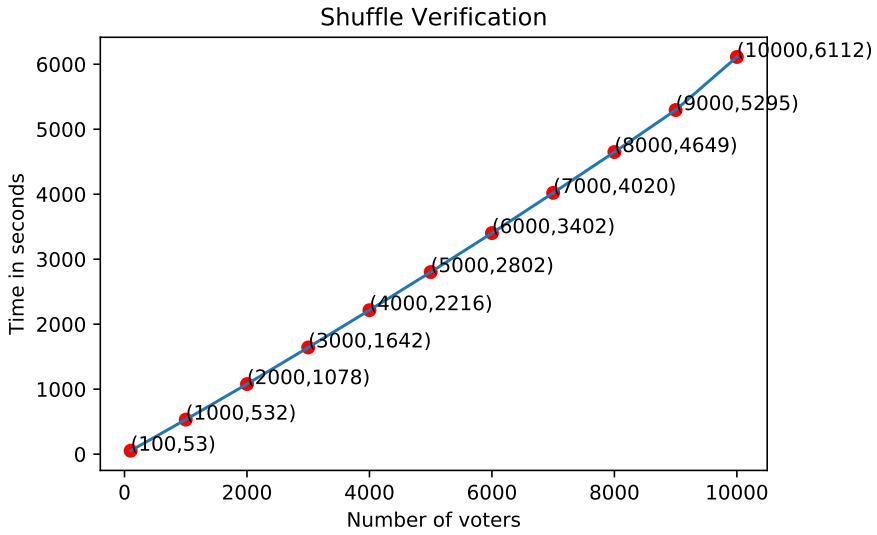

Figure 23: D4: Experiment Times of Verification of Shuffle

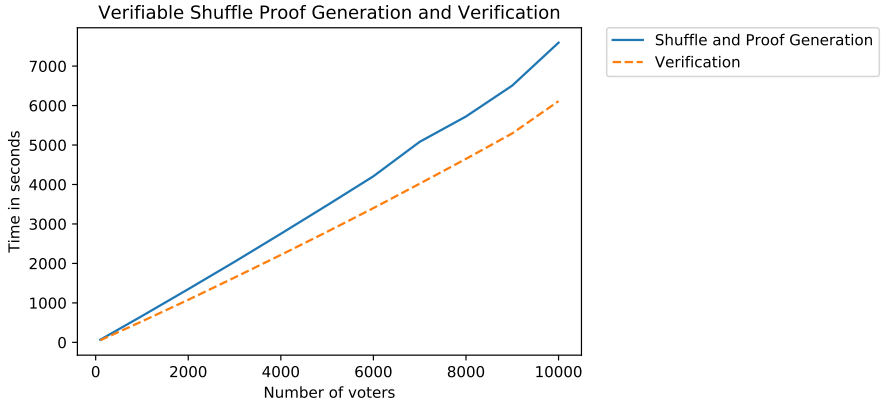

Figure 24: D4: Comparison of Proof Generation and Verification

7.4.2 *Storage Performance*

The overall storage and data transfer requirements of the implementation based on individual elements can be seen in Table 11 and the required data relative to the number of voters can be seen in Figure 25.

SETUP The setup phase requires the least amount of data to be handled. For each mix, Schnorr Proof needs 12288 bits and public key 3072 bits. Therefore in the case of three mixes, 46080 bits = 5760 bytes are required to be stored on the blockchain and handled by the network. The combined election key takes 3072 bits = 384bytes. In addition to these basic requirements, the Election Authority will also publish deadlines of the phases, the election question, and choices,

Phase	Actor	Data	Size (bits)
Setup	$\mathcal{M}$	Schnorr Proof	12288
Setup	$\mathcal{M}$	Public key	3072
Setup	$\mathcal{EA}$	Election Key	3072
Casting	$\mathcal{V}$	Ballot	7792
Authentication	$\mathcal{AA}$	DVRP	12288
Authentication	$\mathcal{AA}$	Re-encrypted Ballot	6144
Anonymization	$\mathcal{M}$	Shuffle (per ballot)	6144
Anonymization	$\mathcal{M}$	Proof (per ballot)	27648
Tally	$\mathcal{M}$	Decryption Proof	9216
Tally	$\mathcal{M}$	Decryption Share	3072

Table 11: D4: Storage Requirements of Blockchain

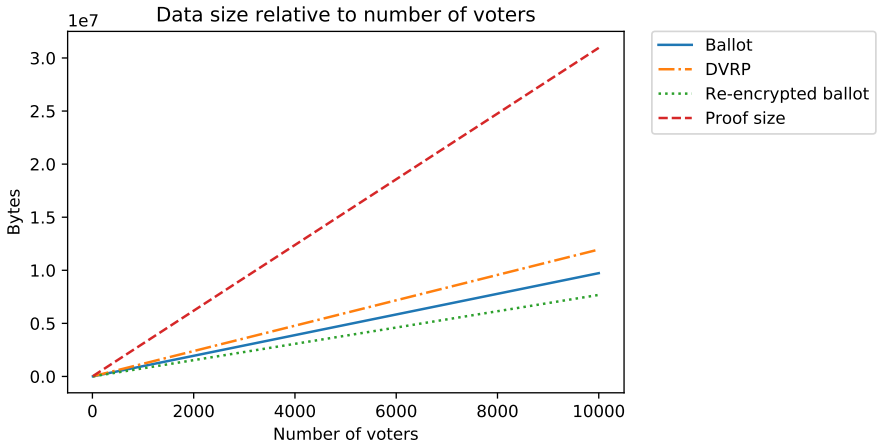

Figure 25: D4: Data Size Relative to the Number of Voters

which we consider to be negligible in comparison to the rest of the data.

CASTING In the casting phase each voter will cast their ballot. They are allowed to do so multiple times or even abstain, but we consider an honest execution of the protocol with a single ballot per voter. Each ECIES encrypted ballot takes $7792\text{bits} = 974\text{bytes}$.

AUTHENTICATION In the authentication phase the authority will store on the blockchain the DVRP for the voter and re-encrypted ballot for the voter and the mix network. The DVRP requires 12288 bits = 1536 bytes. The re-encrypted ballot will require 6144 bits = 768 bytes. The authority will have to additionally generate several random values, which it is assumed not to store. However, the authority is ex-

pected to keep a list of 256 bits = 32 bytes hash and ballot 6144 bits = 768 bytes locally. The list is for checking whether a ballot is the last one submitted by the voter and for private verification. It is assumed that this information will not be stored after the phase ends.

ANONYMIZATION In the anonymisation phase, each mix is going to generate a shuffle and proof and store on the blockchain the shuffled ballots and the proof. Ignoring the constant data necessary for each shuffle the mixes need to store per ballot 27648 bits = 3456 bytes for the proof and 6144 bits = 768 bytes per shuffled list.

TALLY In the decryption the mixes each decrypt a share of the ballot producing a share of 3072 bits = 384 bytes per ballot per mix, which amounts to 9216 bits = 1152 bytes for each ballot in case of 3 mixes. Furthermore, each mix creates a decryption proof of size 9216 bits = 1152 bytes, which amounts to 27648 bits = 3456 bytes. The decrypted vote is negligible in size in comparison to the rest of the data thus we ignore this for analysis.

7.5 DISCUSSION

The execution times measured in the experiments show that the runtime scales proportional to the number of voters in the anonymity set. The bottleneck of the design is the anonymisation part. Any other operation is either distributed already, can be trivially parallelised, or takes negligible time in comparison.

The naive implementation has demonstrated that a mix server can anonymise 10,000 votes in about 2 hours and 7 minutes. Given the median time of 0.67452 seconds per vote to generate a proof the estimated times relative to the number of voters are shown in Table 12. Thus, it would not be feasible for larger scale elections as it is given the time cost.

Nevertheless, it is still applicable to for example national elections due to the anonymity set sizes. As an example consider general elections of Netherlands in 2017. During the election, 10,516,041 valid votes were cast. There are about 10,000 polling stations (*stembureau*). The average polling station thus receives about 1,052 votes. The aggregate results of the polling stations are considered public and will be published when counted. Therefore, the shuffling could be applied based on the polling station and therefore would be feasible for a generic workstation.

An important factor in determining the feasibility is the amount of storage needed per actor and in total how much would be needed to store on the blockchain. As shown by the experimentation each vote requires 12110 bytes to store all the information through all the

# Votes	Time	Data	Bandwidth
10	6 seconds	118 KiB	11 bit/s
100	1 minute	1 MiB	112 bit/s
1000	11 minutes	12 MiB	1 Kbit/s
10 000	1 hour	115 MiB	11 Kbit/s
100 000	19 hours	1 GiB	110 Kbit/s
1 000 000	8 days	11 GiB	1 Mbit/s
10 000 000	78 days	113 GiB	11 Mbit/s
100 000 000	2 years	1 TiB	107 Mbit/s
1 000 000 000	21 years	11 TiB	1 Gbit/s

Table 12: D4: Scalability of the Design

phases. Given this amount and ignoring any constants, we can see the scalability of the storage relative to the number of votes in Table 12.

We conclude that the naive implementation is feasible as it is for a large-scale national election regarding storage. Up to 1 million votes the storage would be satisfied by an average phone, 10 million votes the required storage would be satisfied by an average laptop, and 100 million is feasible for desktop computers. By comparison, Bitcoin blockchain is 173 GB as of Q2 2018, and Ethereum blockchain exceeds 1TB as of May 2018.

Regarding communication, we have calculated the average bandwidth required if the election took place over 24 hours as seen in Table 12. The average bandwidth over 24 hours would be feasible up to 10 million votes on a consumer connection. This analysis is not considering the asymmetry of traffic throughout the day.

7.5.1 *Comparison to existing schemes*

Due to the diverse nature of the proposed schemes, the security requirements they satisfy, their scalability, and the detail of evaluation it is difficult to compare the schemes objectively. However, we attempt to briefly compare the schemes on privacy, verifiability, fairness, and performance.

As shown in Chapter 3, none of the existing schemes prevent the voter from creating receipts. Consequently, our scheme is the first such blockchain based voting scheme that achieves this privacy property. Concerning verifiability, there are not many differences between schemes and most of them satisfy this strongly. Our scheme also achieves a degree of coercion-resistance by allowing revoting. Schemes which only give the voter a single attempt are less robust to errors

and less tolerant to coercion. Currently, most proposed schemes only allow for a single attempt per voter.

Our scheme achieves fairness through the separation of the casting and tally phase. This property cannot be achieved in schemes which send votes as tokens, such as the blind signature or privacy oriented cryptocurrency based schemes. Fairness is an essential security requirement as it can significantly affect the outcome of the vote.

There is a broad spectrum regarding complexity and performance. Even though the literature does not contain detailed results, we can expect that token based schemes perform better regarding communication and computation. Generally, the obfuscation and masking based schemes can be expected to perform worse than our scheme, because it is expected that voters can participate throughout the process and communicate with everyone else. Ring signature based schemes do not scale well regarding storage. The throughput of the underlying cryptocurrencies limits privacy-preserving cryptocurrency based schemes. For example, in the case of ZCash 2.3 million votes could be recorded per day assuming nothing other than the ballot related transactions are mined.

In conclusion, our scheme has significant advantages concerning security, but some schemes claim or can be expected to have better performance. Therefore, the suitability of the scheme depends on the use case.

7.5.2 *Improvements*

The implementation could be improved in several aspects to gain significant performance enhancements.

First of all, the heaviest procedure regarding computation is shuffling. We have implemented the algorithm as described in the reference paper based on classical discrete logarithm modulo p group. An implementation adapted to Elliptic Curve Cryptography would see significant enhancements regarding all the critical aspects of the computational difficulty, storage requirements, and communication. The improvement is because for the same level of security the key sizes would be $3072/256 = 12$ times smaller.

Secondly, for validation, we implemented a naive version of the algorithm for shuffling. Fast implementation could take advantage of concurrent calculations. In nearly every step of the algorithm, computations are run for lists of values which are not dependent on each other. Therefore, the parallel execution could show significant improvements.

Thirdly, the implementation is done in a high-level language which has significant overhead. Doing it in lower level language and optimising for hardware can improve performance regarding computation time.

Finally, the anonymisation nodes are preselected before the election begins. Thus, they could be chosen such that they have an appropriate amount of resources available for the volume of voters in the election. The validation was done on consumer grade hardware to validate the feasibility for any participant to act as a mix. Furthermore, the verification of the shuffle takes a comparable amount of computations to shuffle and should be accessible to anyone with a reasonable amount of resources as well.

8

DISCUSSION AND FUTURE WORK

Voting is a fundamental component of democracy. Controversies often surround elections. For example, during the 2016 United States elections, it was reported that nation-state adversaries targeted voting systems in 20 states of which four occasions there was a successful intrusion[1]. No data manipulation or change has been reported. Another example is 2018 Russian elections, where immediately during the election evidence was posted online of widescale election fraud[2]. Most recently 2018 Zimbabwe presidential election stirred up controversy[3]. We do not provide these examples to take a political stance, but rather to demonstrate the fact that paper-based voting is controversial regardless of geographical location or the state of democracy in the country.

In all of these cases, the transparency of the process can be increased by cryptographic methods and electronic means. It would allow virtually anyone to verify the results and prove that votes were recorded honestly. The electronic means are also practical regarding cost and effort to conduct the process, carbon footprint, security, efficiency, improved accuracy (human errors in casting and counting), and convenience to the voters. According to the calculus of voting theory, the convenience would increase the turnout. It benefits the democracy as the will of the people is reflected more accurately.

The existing electronic voting solutions have some shortcomings. They usually require the voters to trust either centralised authority or appointed authorities. There are no mechanisms to verify the records shown to the voter are complete. It is not possible for voters to actively participate in the election beyond voting. The systems have single points of failure due to centralisation and therefore are not robust against denial of service attacks. Furthermore, there is a general lack of trust in electronic voting as demonstrated by legal bans in the Netherlands, Germany, and Norway.

A number of these problems can be solved by distributing the election process. First of all, the process needs to be made transparent so that anyone can see the entire log and nobody can remove records

1 *U.S. official: Hackers targeted voter registration systems of 20 states*. Chicago Tribune. Article available from http://www.chicagotribune.com/news/nationworld/ct-hackers-target-election-systems-20160930-story.html.

2 *Vladimir Putin secures record win in Russian presidential election*. The Guardian. Article available from https://www.theguardian.com/world/2018/mar/19/vladimir-putin-secures-record-win-in-russian-presidential-election.

3 *Zimbabwe's Opposition Calls Vote Results 'Fraudulent' in Tense Aftermath*. The New York Times. Article available from https://www.nytimes.com/2018/08/03/world/africa/zimbabwe-election-chamisa.html.

once added. Secondly, as the records are public, anyone with a reasonable computer should be able to verify the election results. Thirdly, voters with reasonable computing resources should be able to volunteer in the election recording and validation process. Furthermore, as the stakeholders are involved in the recording process the trust require in authorities is lowered. Finally, in comparison to the centralised solutions, the network is robust against denial of service attacks due to the distribution. To achieve these goals, we have proposed a scheme that utilises blockchain as the underlying public bulletin board.

Several blockchain-based electronic voting schemes have been proposed already. The proposed schemes generally achieve the verifiability requirements necessary for remote electronic voting. However, concerning privacy, they satisfy the weakest notion of ballot privacy. Most of them are designed on top of existing cryptocurrencies, which has several disadvantages.

- First, the cost of running the elections is pegged to the market price of the tokens, which has been proven to fluctuate highly. It makes predictions and budgeting for elections difficult.
- Second, as shown by CryptoKitties congestion[4] of Ethereum the network can be clogged by a single honest distributed application or in case of Bitcoin during the bubble of 2017 transaction queue grew in size to where it could take days before a transaction was included in the blockchain[5]. The cryptocurrency blockchains are not necessarily optimised for the number of messages/transactions necessary to send in a short period.
- Third, the tokens used for voting have real monetary value. Thus there is an incentive not to use them for the intended purpose of voting.
- Fourth, if the voting works by casting tokens, there is by definition a live tally, which is against fair voting principles.

In this chapter we first discuss how the research goal has been achieved in Section 8.1. Then we identify unresolved problems and potential directions for future work in Section 8.2. We conclude the thesis by final remarks in Section 8.3.

4 *CryptoKitties craze slows down transactions on Ethereum*. BBC News. Article available from https://www.bbc.com/news/technology-42237162.

5 *Bitcoin Mempool Woes Worsen as Over 220,000 Unconfirmed Transactions Remain Queued*. News BTC. Article available from https://www.newsbtc.com/2017/12/08/bitcoin-mempool-woes-worsen-220000-unconfirmed-transactions-remain-queued/.

8.1 DISCUSSION

We have given the context and motivation of the problem in the previous section. In this chapter we dicuss the research question and how it has been addressed by our work. To restate the research goal:

> *How can we achieve strong notions of privacy for voters in remote electronic voting setting while providing transparency to voters through an opportunity to participate in the process of election recording?*

We now proceed to investigate the subquestions and how they have been addressed in turn.

> *How to achieve strong privacy in remote voting supported by blockchain?*

Privacy notions in electronic voting are *Ballot Privacy*, *Receipt-Freeness*, and *Coercion-Resistance*. We achieve *Ballot Privacy* through ElGamal encryption and shuffling of the ballots. Through ElGamal encryption, we ensure that nobody can distinguish votes. The shuffling will break the link between the eventually decrypted ballot and the ballot submitted by the voter. *Receipt-Freeness* is achieved in our scheme through Re-Encryption, Designated Verifier Re-Encryption Proofs (DVRP) and Stealth Addresses which do not let the voter prove to anyone how they voted. Re-Encryption introduces new randomness, which the voter is oblivious to, to the ballot. The voter is convinced that their ballot has been honestly re-encrypted by the DVRP. The DVRP ensures that nobody else will be convinced. The Stealth Addresses are used to hide the intended recipient and offer deniability in the participation of elections.

> *How to increase the transparency by employing volunteers in the election validation and record keeping in a distributed manner?*

The transparency is increased by openness and equal opportunity to participate in the process. All the public messages related to the election are recorded in a distributed ledger. This ledger is append-only and maintained mostly by the voters. Provably secure dynamic proof-of-stake scheme Ouroboros ensures that any voter regardless of their processing power has equal chance to participate in the process. Therefore, in order to change which votes are included in the log, there would need to be a majority. If there is a majority of voters colluding, then it makes no sense to undermine the process, but vote honestly, and the same objective would be achieved.

> *How to make the process irrefutably verifiable?*

The verifiability of the process is ensured by the cryptographic primitives used. First of all, the voter is in charge of creating their ballot. Thus it will be cast-as-intended. DVRP and records on the blockchain are sufficient to convince the voter that it has been recorded-as-cast. Finally, the proof constructed in the shuffling process convinces the voter that their vote is tallied-as-recorded. In order to change the records, a majority is required. If there is a majority, then there is already consensus and attacking the system would not make sense as honest actions would require fewer resources and effort.

> *How to make the process simple for voters who only want to participate by voting?*

This subquestion has not been addressed directly. However, it is more of an actual implementation question. To make the process simple for voters is down to the voting software. Admittedly, the verification of cryptographic proofs might be well beyond the competency of anyone but experts. However, this could be simplified by software for example by displaying a red colour when proof does not verify or displaying green colour when proof does verify. Therefore, out scheme still requires trust in the experts who are developing the system.

Thus we conclude we have achieved the set research question, except one subquestion which is up to the real world implementation.

8.2 FUTURE WORK

We have identified several directions this research can be extended on in the future.

Fast implementation of the design. The validation was done by a naive implementation and could be improved in several aspects as noted before. This work could adapt the design to Elliptic Curve Cryptography (ECC) in several steps. The implementation itself can benefit from ECC, concurrency, and using lower level language. The performance regarding speed, storage, and bandwidth can then be validated for larger scale elections in practice.

Implementation on top of blockchain network. In this work, we did not implement the underlying blockchain network or the onion routing network. Therefore, it would be interesting to implement or adapt these parts to the validation as well. It would be a step towards a complete working version of the system. Then the network requirements and scalability can be validated more accurately in practice rather than just theoretically.

Improving the security properties of the protocol. The design does not satisfy the latest requirements such as accountability, coercion-resistance, or coercion-evidence. These are relatively new notions and are not straightforward in how they should be designed or whether they apply to the design as it is. Considerable research is necessary to

answer these questions. However, by allowing re-voting, the prerequisite for coercion-resistance and coercion-evidence is satisfied.

Improving the design to weaken the assumptions. We made numerous assumptions about the adversary and the general model. Some of these assumptions can be removed if the design is evolved. For example, implementing 2-factor authentication, or trusted hardware tokens can be used to remove the assumption that voter software and hardware are trusted. Self-sovereign identity could be used to eliminate Authentication Authority. Smart contracts could be used to remove several administrative tasks of the Election Authority. This work would be a step towards implementation applicable in the real world.

Use case implementation. We have described a generic election protocol design. Voting takes place under various conditions. For example, Dutch general elections have around 10,000 polling stations. The polling station aggregate data is considered public information. Therefore the design should be enhanced to take into account these specifics. In another example, there may be elections with multiple questions. A direction towards real application would be to choose a specific use case, adapt the design, and implement it for that case.

8.3 CONCLUDING REMARKS

We have outlined the motivation for distributed verifiable remote electronic voting, due to the problems surrounding traditional voting. The properties blockchain offers regarding integrity, consensus, and transparency makes it a prospective candidate for distributing the process and increasing transparency. Survey on the existing blockchain schemes revealed that privacy had not been adequately addressed. We proposed a scheme that satisfies strong privacy requirement receipt-freeness. We also provide security analysis on the privacy and verifiability. Complexity analysis shows that the communication and computation scale linearly to the voters. Validation demonstrated that the scheme could be applied to general elections regarding storage and run-time requirements. However, the scheme could still be improved in several aspects - performance, security properties, relaxing assumptions, and validation concerning complete implementation.

BIBLIOGRAPHY

[1] Ben Adida. "Advances in Cryptographic Voting Systems." AAI0810143. PhD thesis. Cambridge, MA, USA, 2006.

[2] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. "Secure Multiparty Computations on Bitcoin." In: *2014 IEEE Symposium on Security and Privacy*. 2014, pp. 443–458. DOI: 10.1109/SP.2014.35.

[3] UN General Assembly. "Universal declaration of human rights." In: *UN General Assembly* (1948).

[4] Adam Back et al. *Hashcash-a denial of service counter-measure*. 2002.

[5] Josh Benaloh. "Ballot Casting Assurance via Voter-initiated Poll Station Auditing." In: *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*. EVT'07. Boston, MA: USENIX Association, 2007, pp. 14–14. URL: http://dl.acm.org/citation.cfm?id=1323111.1323125.

[6] Josh Benaloh and Dwight Tuinstra. "Receipt-free secret-ballot elections." In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. ACM. 1994, pp. 544–553.

[7] Iddo Bentov and Ranjit Kumaresan. "How to Use Bitcoin to Design Fair Protocols." In: *Advances in Cryptology – CRYPTO 2014*. Ed. by Juan A. Garay and Rosario Gennaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 421–439. ISBN: 978-3-662-44381-1.

[8] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. "SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions." In: *2015 IEEE Symposium on Security and Privacy*. 2015, pp. 499–516. DOI: 10.1109/SP.2015.37.

[9] Sylvia Bishop and Anke Hoeffler. "Free and fair elections: A new database." In: *Journal of Peace Research* 53.4 (2016), pp. 608–616. DOI: 10.1177/0022343316642508. eprint: https://doi.org/10.1177/0022343316642508. URL: https://doi.org/10.1177/0022343316642508.

[10] Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. "An End-to-end Voting-system Based on Bitcoin." In: *Proceedings of the Symposium on Applied Computing*. SAC '17. Marrakech, Morocco: ACM, 2017, pp. 1836–1841. ISBN: 978-1-4503-4486-9. DOI: 10.1145/3019612.3019841. URL: http://doi.acm.org/10.1145/3019612.3019841.

[11] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. "Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator." In: *E-Voting and Identity*. Ed. by Ammar Alkassar and Melanie Volkamer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 111–124. ISBN: 978-3-540-77493-8.

[12] Dan Boneh. "The Decision Diffie-Hellman problem." In: *Algorithmic Number Theory*. Ed. by Joe P. Buhler. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 48–63. ISBN: 978-3-540-69113-6.

[13] Stephen Burgen. *Thousands protest and strike over Catalonia referendum violence*. Ed. by theguardian.com. [Online; posted 03-October-2017]. 2017. URL: https://www.theguardian.com/world/2017/oct/03/catalonia-holds-general-strike-protest-referendum-violence.

[14] D. Chaum. "Secret-ballot receipts: True voter-verifiable elections." In: *IEEE Security Privacy* 2.1 (2004), pp. 38–47. ISSN: 1540-7993. DOI: 10.1109/MSECP.2004.1264852.

[15] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. "Scantegrity: End-to-End Voter-Verifiable Optical- Scan Voting." In: *IEEE Security Privacy* 6.3 (2008), pp. 40–46. ISSN: 1540-7993. DOI: 10.1109/MSP.2008.70.

[16] David L. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." In: *Commun. ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563. URL: http://doi.acm.org/10.1145/358549.358563.

[17] David Chaum. "Blind Signatures for Untraceable Payments." In: *Advances in Cryptology*. Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman. Boston, MA: Springer US, 1983, pp. 199–203. ISBN: 978-1-4757-0602-4.

[18] David Chaum. "Blind signatures for untraceable payments." In: *Advances in cryptology*. Springer. 1983, pp. 199–203.

[19] David Chaum and Torben Pryds Pedersen. "Wallet Databases with Observers." In: *Advances in Cryptology — CRYPTO' 92*. Ed. by Ernest F. Brickell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 89–105. ISBN: 978-3-540-48071-6.

[20] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L Rivest, Peter YA Ryan, Emily Shen, and Alan T Sherman. "Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes." In: *EVT* 8 (2008), pp. 1–13.

[21] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. "On Some Incompatible Properties of Voting Schemes." In: *Towards Trustworthy Elections: New Directions in Electronic Voting*. Ed. by David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 191–199. ISBN: 978-3-642-12980-3. DOI: 10.1007/978-3-642-12980-3_11. URL: https://doi.org/10.1007/978-3-642-12980-3_11.

[22] C. Culnane and S. Schneider. "A Peered Bulletin Board for Robust Use in Verifiable Voting Systems." In: *2014 IEEE 27th Computer Security Foundations Symposium*. 2014, pp. 169–183. DOI: 10.1109/CSF.2014.20.

[23] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. "Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain." In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Cham: Springer International Publishing, 2018, pp. 66–98. ISBN: 978-3-319-78375-8.

[24] W. Diffie and M. Hellman. "New directions in cryptography." In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055638.

[25] T. Elgamal. "A public key cryptosystem and a signature scheme based on discrete logarithms." In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472. ISSN: 0018-9448. DOI: 10.1109/TIT.1985.1057074.

[26] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194. ISBN: 978-3-540-47721-1.

[27] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. "A practical secret voting scheme for large scale elections." In: *Advances in Cryptology — AUSCRYPT '92*. Ed. by Jennifer Seberry and Yuliang Zheng. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 244–251. ISBN: 978-3-540-47976-5.

[28] G. S. Grewal, M. D. Ryan, S. Bursuc, and P. Y. A. Ryan. "Caveat Coercitor: Coercion-Evidence in Electronic Voting." In: *2013 IEEE Symposium on Security and Privacy*. 2013, pp. 367–381. DOI: 10.1109/SP.2013.32.

[29] G. S. Grewal, M. D. Ryan, L. Chen, and M. R. Clarkson. "Du-Vote: Remote Electronic Voting with Untrusted Computers." In: *2015 IEEE 28th Computer Security Foundations Symposium*. 2015, pp. 155–169. DOI: 10.1109/CSF.2015.18.

[30] Jens Groth. "Efficient Maximal Privacy in Boardroom Voting and Anonymous Broadcast." In: *Financial Cryptography*. Ed. by Ari Juels. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 90–104. ISBN: 978-3-540-27809-2.

[31] F. Hao. "Anonymous voting by two-round public discussion." English. In: *IET Information Security* 4 (2 2010), 62–67(5). ISSN: 1751-8709. URL: http://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2008.0127.

[32] James Heather and David Lundin. "The Append-Only Web Bulletin Board." In: *Formal Aspects in Security and Trust*. Ed. by Pierpaolo Degano, Joshua Guttman, and Fabio Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 242–256. ISBN: 978-3-642-01465-9.

[33] Sven Heiberg, Ivo Kubjas, Janno Siim, and Jan Willemson. *On Trade-offs of Applying Block Chains for Electronic Voting Bulletin Boards*. Cryptology ePrint Archive, Report 2018/685. https://eprint.iacr.org/2018/685. 2018.

[34] Jörg Helbach and Jörg Schwenk. "Secure Internet Voting with Code Sheets." In: *E-Voting and Identity*. Ed. by Ammar Alkassar and Melanie Volkamer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 166–177. ISBN: 978-3-540-77493-8.

[35] Martin Hirt and Kazue Sako. "Efficient Receipt-Free Voting Based on Homomorphic Encryption." In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 539–556. ISBN: 978-3-540-45539-4.

[36] Freedom House. "Populists and autocrats: The dual threat to global democracy." In: *Freedom in the World Report* (2017).

[37] Freedom House. "Freedom in the World 2018. Democracy in crisis." In: *Freedom House* (2018).

[38] Jen-Ho Hsiao, Raylin Tso, Chien-Ming Chen, and Mu-En Wu. "Decentralized E-Voting Systems Based on the Blockchain Technology." In: *Advances in Computer Science and Ubiquitous Computing*. Ed. by James J. Park, Vincenzo Loia, Gangman Yi, and Yunsick Sung. Singapore: Springer Singapore, 2018, pp. 305–309. ISBN: 978-981-10-7605-3.

[39] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. "Designated Verifier Proofs and Their Applications." In: *Advances in Cryptology — EUROCRYPT '96*. Ed. by Ueli Maurer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 143–154. ISBN: 978-3-540-68339-1.

[40] Paul Cruz Jason and Kaji Yuichi. "E-voting System Based on the Bitcoin Protocol and Blind Signatures." In: *TOM* 10.1 (2017), pp. 14–22. ISSN: 1882-7780. URL: https://ci.nii.ac.jp/naid/170000148490/en/.

[41] H. L. Jonker and E. P. de Vink. "Formalising Receipt-Freeness." In: *Information Security*. Ed. by Sokratis K. Katsikas, Javier López, Michael Backes, Stefanos Gritzalis, and Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 476–488. ISBN: 978-3-540-38343-7.

[42] H. Jonker and J. Pang. "Bulletin Boards in Voting Systems: Modelling and Measuring Privacy." In: *2011 Sixth International Conference on Availability, Reliability and Security*. 2011, pp. 294–300. DOI: 10.1109/ARES.2011.50.

[43] Hugo Jonker, Sjouke Mauw, and Jun Pang. "Privacy and verifiability in voting systems: Methods, developments and trends." In: *Computer Science Review* 10 (2013), pp. 1 –30. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2013.08.002. URL: http://www.sciencedirect.com/science/article/pii/S1574013713000282.

[44] Ari Juels, Dario Catalano, and Markus Jakobsson. "Coercion-resistant electronic elections." In: *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. ACM. 2005, pp. 61–70.

[45] MyungSan Jun. "Blockchain government - a next form of infrastructure for the twenty-first century." In: *Journal of Open Innovation: Technology, Market, and Complexity* 4.1 (2018).

[46] MyungSan Jun. "Blockchain government-a next form of infrastructure for the twenty-first century." In: *Journal of Open Innovation: Technology, Market, and Complexity* 4.1 (2018), p. 7.

[47] Aggelos Kiayias and Moti Yung. "Self-tallying Elections and Perfect Ballot Secrecy." In: *Public Key Cryptography*. Ed. by David Naccache and Pascal Paillier. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 141–158. ISBN: 978-3-540-45664-3.

[48] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. "End-to-end verifiable elections in the standard model." In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 468–498.

[49] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol." In: *Advances in Cryptology – CRYPTO 2017*. Ed. by Jonathan Katz and Hovav Shacham. Cham: Springer International Publishing, 2017, pp. 357–388. ISBN: 978-3-319-63688-7.

[50] Marko Kovic. "Blockchain for the people: Blockchain technology as the basis for a secure and reliable e-voting system." In: (2017).

[51] David W Kravitz. *Digital signature algorithm*. US Patent 5,231,668. 1993.

[52] Annabell Kuldmaa. "On Secure Bulletin Boards for E-Voting." In: *Master. University of Tartu* (2017).

[53] Ranjit Kumaresan and Iddo Bentov. "How to Use Bitcoin to Incentivize Correct Computations." In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS '14. Scottsdale, Arizona, USA: ACM, 2014, pp. 30–41. ISBN: 978-1-4503-2957-6. DOI: 10.1145/2660267.2660380. URL: http://doi.acm.org/10.1145/2660267.2660380.

[54] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. "Accountability: Definition and Relationship to Verifiability." In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. CCS '10. Chicago, Illinois, USA: ACM, 2010, pp. 526–535. ISBN: 978-1-4503-0245-6. DOI: 10.1145/1866307.1866366. URL: http://doi.acm.org/10.1145/1866307.1866366.

[55] Lynn Landes. *Scrap the "secret" ballot - return to open voting*. Ed. by freepress.org. [Online; posted 05-November-2005]. 2005. URL: http://freepress.org/article/scrap-secret-ballot-return-open-voting.

[56] Byoungcheon Lee and Kwangjo Kim. "Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer." In: *Information Security and Cryptology — ICISC 2002*. Ed. by Pil Joong Lee and Chae Hoon Lim. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 389–406. ISBN: 978-3-540-36552-5.

[57] Helger Lipmaa, Guilin Wang, and Feng Bao. "Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction." In: *Automata, Languages and Programming*. Ed. by Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 459–471. ISBN: 978-3-540-31691-6.

[58] Yi Liu and Qi Wang. *An E-voting Protocol Based on Blockchain*. Cryptology ePrint Archive, Report 2017/1043. https://eprint.iacr.org/2017/1043. 2017.

[59] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. "A Smart Contract for Boardroom Voting with Maximum Voter Privacy." In: *Financial Cryptography and Data Security*. Ed. by Aggelos Kiayias. Cham: Springer International Publishing, 2017, pp. 357–375. ISBN: 978-3-319-70972-7.

[60] S. J. Murdoch and G. Danezis. "Low-cost traffic analysis of Tor." In: *2005 IEEE Symposium on Security and Privacy (S P'05)*. 2005, pp. 183–195. DOI: 10.1109/SP.2005.12.

[61] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

[62] Yomna Nasser, Chidinma Okoye, Jeremy Clark, and Peter YA Ryan. "Blockchains and Voting: Somewhere between hype and a panacea." In: (2018).

[63] C. Andrew Neff. "A Verifiable Secret Shuffle and Its Application to e-Voting." In: *Proceedings of the 8th ACM Conference on Computer and Communications Security*. CCS '01. Philadelphia, PA, USA: ACM, 2001, pp. 116–125. ISBN: 1-58113-385-5. DOI: 10.1145/501983.502000. URL: http://doi.acm.org/10.1145/501983.502000.

[64] C Andrew Neff. "Verifiable mixing (shuffling) of ElGamal pairs." In: (2003).

[65] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. "Anonymous connections and onion routing." In: *IEEE Journal on Selected Areas in Communications* 16.4 (1998), pp. 482–494. ISSN: 0733-8716. DOI: 10.1109/49.668972.

[66] Robert Riemann. "Towards Trustworthy Online Voting: Distributed Aggregation of Confidential Data." Theses. Ecole normale supérieure de Lyon, Dec. 2017. URL: https://hal.inria.fr/tel-01675509.

[67] Robert Riemann and Stéphane Grumbach. "Distributed Protocols at the Rescue for Trustworthy Online Voting." In: *CoRR* abs/1705.04480 (2017). arXiv: 1705.04480. URL: http://arxiv.org/abs/1705.04480.

[68] William H Riker and Peter C Ordeshook. "A Theory of the Calculus of Voting." In: *American political science review* 62.1 (1968), pp. 25–42.

[69] Kazue Sako and Joe Kilian. "Receipt-free mix-type voting scheme." In: *Advances in Cryptology—EUROCRYPT'95*. Springer. 1995, pp. 393–403.

[70] C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards." In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 239–252. ISBN: 978-0-387-34805-6.

[71] Adi Shamir. "How to Share a Secret." In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: http://doi.acm.org/10.1145/359168.359176.

[72] Claude E Shannon. "Communication theory of secrecy systems." In: *Bell Labs Technical Journal* 28.4 (1949), pp. 656–715.

[73] Yu Takabatake, Daisuke Kotani, and Yasuo Okabe. "An anonymous distributed electronic voting system using Zerocoin." In: (2016).

[74] Pavel Tarasov and Hitesh Tewari. *Internet Voting Using Zcash*. Cryptology ePrint Archive, Report 2017/585. https://eprint.iacr.org/2017/585. 2017.

[75] Inter-Parliamentary Union. *Declaration on Criteria for Free and Fair Elections*. Inter-Parliamentary Union, 1994.

[76] Eric M. Uslaner and Marc Hooghe. *Trust and Elections*. 2018. URL: http://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780190274801.001.0001/oxfordhb-9780190274801-e-17.

[77] Nicolas Van Saberhagen. *Cryptonote v 2. 0*. 2013.

[78] Aleksandar Vasovic and Mike Collett-White. *Crimea prepares for referendum under heavy military presence*. Ed. by reuters.com. [Online; posted 15-March-2014]. 2014. URL: https://www.reuters.com/article/us-ukraine-crisis-crimea/crimea-prepares-for-referendum-under-heavy-military-presence-idUSBREA2E09R20140315.

[79] Yifan Wu. "An E-voting System based on Blockchain and Ring Signature." In: *Master. University of Birmingham* (2017).

[80] Bin Yu, Joseph Liu, Amin Sakzad, Surya Nepal, Paul Rimba, Ron Steinfeld, and Man Ho Au. *Platform-independent Secure Blockchain-Based Voting System*. Cryptology ePrint Archive, Report 2018/657. https://eprint.iacr.org/2018/657. 2018.

[81] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. "Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System." In: *Applied Cryptography and Network Security*. Ed. by Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 441–457. ISBN: 978-3-642-38980-1.

[82] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. "Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System." In: *Applied Cryptography and Network Security*. Ed. by Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 441–457. ISBN: 978-3-642-38980-1.

[83] Zhichao Zhao and T.-H. Hubert Chan. "How to Vote Privately Using Bitcoin." In: *Information and Communications Security*. Ed. by Sihan Qing, Eiji Okamoto, Kwangjo Kim, and Dongmei Liu. Cham: Springer International Publishing, 2016, pp. 82–96. ISBN: 978-3-319-29814-6.