Anonymous Voting on Ethereum through Zero Knowledge Proof of Membership

created by

Alvaro Alonso Domenech

Inhaltsverzeichnis

1. Introduction **1**

2. Background **2**
- 2.1. E-Voting . . . 3
  - 2.1.1. Direct Recording Electronic systems . . . 3
  - 2.1.2. Internet Voting . . . 4
  - 2.1.3. E-Voting Properties . . . 4
  - 2.1.4. Future of E-Voting . . . 5
- 2.2. Blockchain . . . 6
  - 2.2.1. System Components . . . 7
  - 2.2.2. Smart Contracts . . . 9
  - 2.2.3. Blockchain Voting . . . 9
- 2.3. Zero-Knowledge Proving Schemes . . . 11
  - 2.3.1. ZK-Snark . . . 12
  - 2.3.2. Zokrates . . . 12

3. Related Work **13**

4. System Design **16**
- 4.1. Design Choices . . . 16
- 4.2. Overview . . . 17
- 4.3. Election Stages . . . 18
  - 4.3.1. User Registration . . . 18
  - 4.3.2. Election Deployment . . . 18
  - 4.3.3. Vote Authentication . . . 19

5. Implementation **19**
- 5.1. PoE generation . . . 20
- 5.2. Back-End . . . 21
- 5.3. Front-End . . . 23
  - 5.3.1. Election Deployment . . . 23
  - 5.3.2. Voting . . . 24

6. Evaluation **26**
- 6.1. Protocol . . . 26
- 6.2. Testnet Analysis . . . 27
- 6.3. Security Analysis . . . 29

7. Conclusion **30**

8. Future Work **31**

Bibliography **32**

A. Appendix **37**

B. Appendix **38**

1. Introduction

Early in 2019, Vitalik Buterin (2019) underlined the importance of taking ä first step toward more privacyön Ethereum [1]. There is a consensus in the Ethereum community that zero-knowledge (zk) proving schemes, specifically zero-knowledge succinct non-interactive arguments of knowledge (zk-snark), are the most promising cryptographic schemes to achieve more privacy. However, the use of zk-snark proofs remains sparse on blockchain-based applications. This study seeks to leverage the beneficial properties of zk-snark to solve a recurrent concern in one of blockchain's popular use cases, anonymous voting. Nowadays, there is a higher public awareness of the threats of sharing personal data on the internet. This concern is even more pronounced in blockchains, as the data can not be deleted once appended. This paper has been written at a time when the confidence in electronic voting systems could be at an all-time low. Two decades populated with continuous scandals of poorly implemented systems have lead many institutions to replace electronic voting systems with more secure paper ballot elections. At the same time, the deepening economic and political inequalities arising of an ever more interconnected world, have urged many institutions to undertake participatory approaches to better align with the interests of their members rather than that of the markets. Participatory politics are based on flatter hierarchies, bottom-up mentality, and topic-oriented discussions, which require a higher involvement from its citizens than the typical four years fixed terms of partisan politics. Additionally, the social distancing needed to overcome the current COVID epidemic has reinforced again the frequently underrated benefits of remote voting.

This work aims to create a verifiable and private voting system that allows eligible individuals to vote anonymously on a public blockchain in a simple manner. On the blockchain, because the data is tamper-proof, on public ledgers for the results to be legitimate, and anonymous so voters can vote freely. The paper introduces a new voting protocol leveraging one of the latest technology in cryptography, zk-snark. The protocol is designed to effectively run elections in a fully decentralized manner with fewer transactions and resembling the process of paper ballot elections for the convenience of voters. Additionally, the decentralized application (Dapp), *Snarky*, has been developed to identify the challenges and technical difficulties of the proposed protocol. The results of which indicate that a fully decentralized voting system that ensures anonymity is currently possible.

The paper proposes a new voting protocol to make voting on public blockchains private and efficient. The next chapter will review electronic voting systems, blockchains, and zero-knowledge proofs. It will be argued why using smart contracts together with zk-snark on the blockchain could make up for the mistakes of previous e-voting systems.

1. https://hackmd.io/@HWeNw8hNRimMm2m2GH56Cw/rJj9hEJTN?type=view

Chapter 3 follows with a review of the work previously done on verifiable and private voting systems. In chapter 4, the proposed protocol will be explained in detail. Chapter 5 dives in the technical discussion of *Snarky* and explains the major design choices done in the development process. Chapter 6 analyses the properties of the underlying protocol and assess its potential vulnerabilities by simulating a vector attack on *Snarky*. The paper closes with a summary of the conclusions and a discussion of how this study could contribute to future research.

2. Background

Since the 1970s, there has been a constant decrease in voter turnout among established democracies (Franklin 2004, 10). Trans-nationalization has stripped away much of the sovereignty states used to enjoy previously. Geys (2006) and Steiner (2010, 242) have found evidence that the level of integration of national economies into international markets and low probabilities of influencing the outcome affect voter turnout negatively. "Raising inequality in developed countries reinforces the perception that citizens in a number of supposedly consolidated democracies have not only grown more critical of their political leaders, they have also become more cynical about the value of democracy"(Foa und Mounk 2016, 7). Low turnout is a concern because it implies: (1) low participation by less privileged citizens, who are already at a disadvantage in terms of other forms of political participation. High-income individuals have greater access to political power and more capacity to put in place 'rules of the game' that promote their interests (Shaxson, Christensen und Mathiason 2012, 4). (2) Unequal participation means unequal influence. Statistically, right-wing parties tend to slightly benefit from low turnout while it hurts on average the left (Eijk und Egmond 2007). (3) Actual turnout tends to be lower than the official turnout figures suggest. High turnout rates contribute to horizontal and vertical accountability of politicians and foster legitimacy (Stockemer 2017, 1). To reverse the negative trend, governments focused their efforts on making the voting process more appealing. Using the latest advances in information technologies would increased participation for disadvantaged communities, an antidote to voter apathy, greater voter convenience in terms of voting time and location, access for people with disabilities, money-saving, and greater accuracy (Mohen und Glidden 2001, 2).

2.1. E-Voting

At the turn of the 21st Century, advances in information systems and their fast adoption lead many people to believe that the manual process of counting votes could be easily automated. It was commonly believed that electronic voting would boost voter turnout back, while drastically cutting the cost of organizing elections, saving millions of public funds to the state. Electronic voting (e-voting) refers to any collective decision method aided by modern information communication technologies. Many systems have been suggested, including protocols, software, hardware, and a combination of all of them. However, most systems fall mainly into one of the two categories: Direct Recording Electronic (DRE) systems and Internet Voting (i-voting) [1].

2.1.1. Direct Recording Electronic systems

DRE systems rely on computer devices to record the votes instead of paper ballots. Although DREs speed up significantly the vote counting, from the point of view of the voter, the process remains the same as in paper ballot elections. Voters need to go to the electoral college, register, and wait in line to get into a voting booth. Additionally, DREs raised numerous securities concerns. Compromised devices, code secrecy, lacking standards, inadequate testing of software, buggy implementations, hardware defects, and an overall lack of transparency of the process were some of the threats that Lauer (2004, 181) identified in his study. DREs are commonly used in elections all across the United States, and some pilot programs were launched in European countries. Norden und Famighetti (2015, 4) found that a majority of DREs used in 2015 in the U.S. was more than ten years old. Most of these machines were also found to be outdated, poorly tested, and lacking the required security standards. The effort required to develop and maintain DREs, as well as the vulnerabilities they add to the voting process, has led many people to questioned whether DREs are any better than paper ballot systems. In 2009 the German Supreme Court ruled the use of DREs on elections unconstitutional since the determination of the results could not be scrutinized reliably by anyone without specific computer knowledge (BVerfG 2009). If implemented correctly, DRE systems would make the process of ballot counting faster and more accurate. However, the resources and effort needed to ensure the proper functioning of such systems may well exceed the cost of running traditional elections based on paper ballots. Moreover, DRE systems make the election process more opaque and less accountable.

1. i-voting is a subset of e-voting methods. However, these terms are commonly interchanged in the literature when referring to internet voting

2.1.2. Internet Voting

Internet voting (i-voting) systems rely on the internet to exchange critical information. The biggest proposition of i-voting systems is that it enables voters to cast their votes remotely. Voters could participate in elections from the comfort of their homes, using their own devices. However, voting over an insecure network and device makes the system subject to a wide range of security threats like trojan spyware, vote spoofing, viruses, and man in the middle attacks (Lauer 2004, 182). An even bigger concern is the fact that most of the voting systems developed so far rely on centralized architectures. A central architecture implies that a component or few components have control over the entire systems (Rosenblatt und Hendler 1999). Centralized systems create *single points of failure* in which if the central component is compromised, the entire system is compromised (Atlam und Wills 2019). An attacker getting access to the vote storage or launching a denial of service (Dos) attack against the server could jeopardize the whole election. Estonia is the first and only state to have launched a remote i-voting system nationwide. Despite receiving numerous criticism from the academia and being targeted in 2008 to a large scale DoS attack(Dos) (Hastings u. a. 2011, 40), the system enjoys wide approval from political parties and the general public with online participation between 25% to 30% in the elections of 2016 (Clarke und Martens 2016, 140-141).

2.1.3. E-Voting Properties

Preceded by a wave of e-voting system proposal and projects, numerous studies (Ne Oo und Moe Aung 2013, Jonker und Pieters 2010, Delaune, Kremer und Ryan 2006b, Lee u. a. 2010, Ali und Murray 2016) were published defining desired properties an e-voting systems should possess. From the previous studies, I consolidated a list with the most referenced and general rules:

- **Completeness.** All valid votes are counted correctly.
- **Soundness.** The dishonest voter cannot disrupt the voting.
- **Privacy.** [2] There is no link between the voter's identity and the ballot.
- **Receipt-freeness.** A voter can not convince any other party of how she voted.
- **Coercion resistant.** No effective way people can influence the choices of others through bribery or physical coercion.
- **Eligibility.** Only eligible voters are allowed to vote.

2. As Jonker und Pieters 2010 noted, Anonymity is referred to as "privacyïn the electronic voting community.

- **Uniqueness.** No voter can cast his ballot more than once.
- **Individual verifiability.** A voter can verify that their vote was correctly counted.
- **Universal verifiability.** The published outcome is the sum of all the votes.
- **Efficiency.** The computations can be performed within a reasonable amount of time.
- **Fairness.** No partial result is available before the final result comes out.

Some of the above-listed properties are closely related. Delaune, Kremer und Ryan (2006a) proofed that privacy is a weaker property than receipt-freeness, which in turn is weaker than *coercion resistance*. Thus, coercion resistance implies *receipt-freeness*, and *receipt-freeness* implies *privacy* (the opposite does not apply). One can also argue that *eligibility* and *uniqueness* are required to have completeness. At the same time, some properties seem to stand in contraposition to each other (Ali und Murray 2016, 179). For example, it is hard to imagine voters correctly verifying their votes (*individual verifiability*) without being able to prove to others what they voted for (*receipt-freeness*). Also, *coercion resistant* makes it harder to audit the correct functioning of the system. The excluding nature of some of the listed properties supports the argument that there is no perfect voting system. Rather, the suitability of a voting system is largely dependent on the context in which the election takes place.

2.1.4. Future of E-Voting

It can be said that every e-voting solution raises understandable security concerns compared to paper ballot elections. The resources needed to launch a large scale cyberattack are significantly lower than on its physical counterpart. A single attacker could compromise the devices of many voters, launch a DoS attack on highly populated areas, or, in the worst case, gain access to the data where votes are stored. Moreover, result verification can only be done by people with very specific knowledge, reducing the probabilities of an attack being detected. E-voting systems are generally criticized for being opaque processes. Voters have no means to confirm that the machines have recorded their votes correctly, nor will they have any assurance of their votes not being changed later (Dill 2004, 1). Two decades of buggy implementations, poorly tested machines, lost votes, wrong votes, and allegations of foreign government interference have raised public awareness about the security threats arising from these systems. After the Iowa caucus debacle early this year, citizens' skepticism towards new technologies in elections might have hit an all-time high.

Although it is highly inadvisable to take critical decisions through e-voting, there is a larger set of decisions, taken on regional and local levels, where e-voting provides a cost-effective and scalable alternative to political representation. The aftermath of the financial crisis of 2008 portrayed how helpless nation-states of a globalize world were against faulty international markets. These faulty behaviors were caused mainly by a lack of audit mechanisms and abuse of trust in established institutions. Since then, many local governments have shifted their approach to more transparent processes with the citizen as the cornerstone of the policies. Major cities around the world have been taking measures to bridge the gap between citizens and global markets, embracing participatory processes such as common neighborhood spaces, participatory budgets, citizen initiatives, or referendums to keep citizens involved in the city's political life. Some, like Paris[3] or Madrid[4], provided centralized web services for its citizens to vote on many matters. In Germany, residents are called to the urns as in state elections, the"100% Tempelhofer Feld"[5] in Berlin, orßave the bees"("*Bienen retten*"[6]) in Bavaria were one of the latest initiatives. Both consultations were organized together with government elections for the sake of saving resources and promote voter turnout. The latter approach does not scale. Local consultations would be at most organized every two years. Only states of relatively small size and abundance of resources, like Switzerland, could regularly afford to run citizen consultations. Contemporary to the emergence of participatory democracy and based on the same principles of transparency and consensus, the blockchain was created. Blockchain is a new computing paradigm which properties could likely outperform the limitations of previous e-voting systems.

2.2. Blockchain

At the beginning of 2009, together with a previously published paper authored under the pseudonym of SSatoshi Nakamoto", the Bitcoin (btc) network was created. Less than a year after the financial crisis of 2008, Bitcoin introduced a secure and anonymous exchange system with no intermediaries, no central regulatory authority, and no trust needed between the parties whatsoever. Bitcoin relies on a decentralized network of peers that converge to a single order of transactions through cryptographic riddles and economic incentives. Apart from cryptocurrencies, the underlying architecture powering btc was found to serve other purposes and became known as "blockchain". "The blockchain is the decentralized transparent ledger with the transaction records—the database that is shared by all network nodes, updated by miners, monitored by everyone, and owned and controlled by no one"(Swan 2015, 1). The term blockchain was coined due to the fact that all transactions are grouped into blocks, and new blocks are chained to the last verified block. All verified blocks are linked to a previously verified block up until the genesis block, the block with the first transactions of the network, forming a chain of blocks that can be traced back to the very first block. Every network has it's own genesis block and is known by all members.

3. https://budgetparticipatif.paris.fr/bp/plugins/download/PB$_{inParis.pdf}$
4. https://decide.madrid.es/vota
5. https://www.thf100.de/
6. https://www.lbv.de/mitmachen/fuer-einsteiger/volksbegehren-artenvielfalt/

The most resilient characteristic of the blockchain is that it creates unique digital assets. In the digital world, assets can be easily duplicated and edited, generating numerous versions of one single asset. The multiplication of assets gives rise to the double-spending problem, where a single asset is spent multiple times. Blockchain solves the double-spending problem by using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions (Nakamoto 2008, 1). Blockchains are peer-2-peer (P2P) networks where every node has the same role and follows the same rules. There is no single point of failure in a system with no central entity, such as the one in the client-server model. This reduction benefits the fault tolerance and stability of the whole system, which is a necessity to guarantee perpetual service. Since every node is equal in a p2p-network, the lack of hierarchy increases the homogeneity of the system, which in turn benefits its security. Blockchains are also more resistant to DoS attacks, as the number of targets is also higher, and thus, the resources needed to launch such an attack. Moreover, the fact that every peer has all the information available helps to monitor the correct functioning of blockchains. Lastly, the blockchain is also defined as an immutable database. An attacker wishing to edit the transaction at a certain block will have to calculate an alternative chain at least longer than the current version the network is working on. To do that, the attacker would need to control a significant portion of the computing power the network (Xu 2016, 5).

2.2.1. System Components

Depending on the application, blockchains can be implemented in many different ways. We will focus on the main two criteria: management of the network and consensus mechanism.

Management and permissions

The most popular way to classify blockchain is by the way the network is managed. Public or permissionless blockchains are networks where anyone can join, of this type are most of the popular networks such as btc and Ethereum. More importantly, anyone can perform validations and send transactions. In this type of blockchain, everyone is assumed to be a potential fraudulent user, so the requirements in this class of network tend to be stricter. Private or permissioned blockchains require an authentication process to join the network. Additionally, these networks tend to have different roles with different levels of permission for certain actions. In permissioned blockchains, there is some certain level of trust between participants but not entirely, as in the case of companies. This trust allowed private chains to afford less expensive consensus protocols and higher transaction rates (Casino, Dasaklis und Patsakis 2019).

Concensus Mechanism

The consensus mechanism refers to the set of rules each node abides to arrive at a unique solution. This set of rules includes how new blocks are added to the chain, the verification of blocks, and how to handle concurrent versions of the chain. Ideally, the consensus protocol addresses the Byzantine Generals Problem Lamport, Shostak und Pease 2019, or of how faithful nodes arrive at an agreement among faulty or malicious participants. Because the number of consensus protocols exceeds the scope of the paper, I will explain the two most popular mechanisms nowadays, proof of work (PoW) and proof of stake (PoS).

PoW is the most popular mechanism as it was first introduced by btc. In PoW, nodes race each other to solve first a cryptographic puzzle in exchange for newly coined btc. This process is also referred to as "mining" as an analogy of gold mining. The puzzle consists of finding out a "nonce", that together with the hash of the last valid block generates a new hash that fulfills the difficulty requirements (Mingxiao u. a. 2017). In the case of bitcoin, the difficulty level is dependent on the leading number of zeroes of the hash. The difficulty of btc's PoW fluctuates so that new blocks are validated every 15 minutes on average. There is no efficient algorithm to calculate the nonce, so nodes have to try combinations until a correct hash is derived randomly. Once a new block is created, all other nodes start working on the most recent node of the longest valid chain. In the case of concurrent versions, miners vote in terms of their computing power, which version is the correct one. The version with the highest computing power will tend to verify new blocks faster, becoming larger than the rest and converging into a single version again. As the computing power of the network increase, so does the difficulty level of the puzzles, thus consuming enormous amounts of energy at a network size of that of btc. Critics argue that PoW benefits wealthy individuals, as people with more resources could buy more computing power.

PoS nodes still have to solve a puzzle; however, the difficulty of the puzzle does not change. Instead, the key to solve this puzzle is the amount of stake (coins x age of coin). Once a node adds a new block, the age of the coin is set to 0, and the node will have to wait a period of time until it can vote again (Mingxiao u. a. 2017). PoS is an energy-saving mechanism compared to Pow. It aligns the network's decisions to the interest of the individuals with a high stake in the network. PoS discourages attacks against the network, as the attackers would need to have a stake in the network (Voshmgir 2019). PoS also posses a fairer way to distribute the wealth of validating blocks. A node A with ten times more stake than node B will get ten times more profits, while with PoW, node B could see no profit whatsoever. For the above reasons, Ethereum, the second-biggest network after btc, is planning to move from PoW to PoS later this year[7].

7. https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/proof-of-stake/

2.2.2. Smart Contracts

The concept of smart contract was first introduced by Szabo (1997) to formalize contract clauses in software. Since it's inception, blockchain was designed to support a tremendous amount of possible transaction types (Nakamoto 2010), smart contracts being one of them. Smart contracts are programs in which source code is stored on the blockchain under a unique address and which execution can be performed by any node in the network. Ä contract can encode any set of rules represented in its programming language—for instance, a contract canexecute transfers when certain events happen"(Luu u. a. 2016, 1). Smart contracts are like traditional contracts without trust elements. One can be entirely sure that the agreement will be enforced if the requirements are met. "Legal disputes and administrative formalities could be avoided with more precision at the time of setting forth agreements, and with automated enforcement mechanisms"(Swan 2015, 15).

Ethereum is the most popular platform to write smart contracts. Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats, and state transition functions (Buterin 2014). Ethereum has its own cryptocurrency, called Ether, to be sent and received between accounts and to pay for “gas”; the cost associated with running the smart contract code. Ethereum has two types of accounts, user accounts, created and managed by real people, and smart contracts, with its states and methods being stored in the network.

2.2.3. Blockchain Voting

Voting was predicted to be one of the first fields disrupted by blockchain technologies. Ethereum's official website even has a guided tutorial to program a voting application as an introduction to Smart Contracts. It is easy to see why blockchain could improve previous e-voting systems. Guaranteeing that votes are not counted twice is of critical importance in voting systems. Blockchain's append-only and censorship-free database and lack of a single point of failure make it hard for an attacker to influence the results. Moreover, a voting platform on a public ledger would bring a level of transparency to the electoral process unseen before. Transparency legitimates results as it convinces rational losers that they lost fairly. More transparency also translates into more security as more people can verify the correct functioning of the system. Instead of relying on institutions, voters could verify their ballot was count correctly and assert the results of elections by themselves.

However, after ten years of development, no blockchain-based voting system has proven superior to previous e-voting systems. A non-trivial aspect of voting on blockchains is the protection of ballot secrecy[8]. Ballot secrecy was introduced in the second half of the 19th Century to reduce bribery and coercion to voters (Gerber u. a. 2013, 1) and

8. Ballot secrecy is analog to the term privacy defined in e-voting properties

is recognized by the United Nations as a crucial component of a fair electoral process (Franck 1992, 64). Public blockchains are said to be pseudo-anonymous by default since the address information can not be linked to a physical identity. Still, the reuse of the same address and multiple input transactions allow any observer to build detailed profiles of an account. In elections, where voters need to prove eligibility to receive a ballot, linking a blockchain address and the voters' identity makes blockchains non-anonymous and violates ballot-secrecy. In the context of blockchains, anonymity is also crucial since, by design, entries can not be deleted once verified. Information deemed harmless at a time could be used for unforeseeable purposes against an individual in the future. Blockchains neglect Art.17 of the European General Data Protection Regulation (GDPR 2018), most commonly known as the right to be forgotten. "GDPR is based on the assumption that [personal identifiable] data can be modified or erased... blockchains, however, render such modifications of data purposefully onerous in order to ensure data integrity and to increase trust in the network"(Finck 2019, 3).

Although seemingly suitable, there are surprisingly few active projects building a voting system on a public blockchain. The most remarkable being Democracy.earth[9]. Democracy.earth rallies to create censorship-free and border-less decentralized autonomous organizations (DAO) on Ethereum. These DAOs verify in a decentralized way that the members are real persons and that they haven't already signed up with another identity, in what they called "proof of humanity"[10]. To become a member of a DAO an applicant needs to provide personal information to be verified by existing members. Democracy.earth also promotes quadratic voting, where voters get a limited number of votes, and voters have to decide not only in which decisions but also how many resources they want to allocate to a certain topic. The decisions taking by every organization and individual is public to everyone. Democracy.earth suggests distributed flat hierarchies and marginal utilities as a way to renew decaying democratic societies. However, Democracy.earth's approach is highly incompatible with how traditional institutions are structured and seems unlikely that this approach could be implemented soon. Moreover, democracy.earth completely neglects the potential perils of making these results completely public.

Nowadays, most of the self-proclaimed "blockchain-based" voting systems tend to be companies reminiscing the e-voting contractors of the early 2000s. This group includes companies such as Voatz[11], Votem[12], Follow My Vote[13], and Agora[14]. These companies offer their own developed private blockchains, which are mainly used as append-only databases. Although most of them state that votes are completely anonymous, these systems use centralized servers, biometric and personal data in their authentication processes [15]. It could be additionally questioned how it is possible to be anonymously invited to a private ledger. The most remarkable project on this group is Voatz, which has run mul-

9. https://democracy.earth/
10. https://github.com/DemocracyEarth/paper32$_Proof_of_Identity$
11. https://voatz.com/
12. https://www.votem.com/blockchain-voting/
13. https://followmyvote.com/blockchain-voting-the-end-to-end-process
14. https://www.agora.vote/technology
15. https://cointelegraph.com/news/voatz-blockchain-app-used-in-us-elections-has-numerous-security-issues-says-report

tiple pilot programs on official election across different counties in the U.S. to ensure the participation of people who couldn't vote otherwise [16] (i.e., overseas / deployed citizens).

Although most of the blockchain-based voting systems on public chains underestimate the risks of putting personal data on public blockchains, attempts have been made to make public blockchains more anonymous. Recent progress on zero-knowledge (zk) proofs, especially on the succinct non-interactive argument of knowledge (zk-snark), have positioned zk-snark as the most promising cryptographic schemes to provide full anonymity on the blockchain. In 2017, Ethereum introduced in its Byzantium fork precompiled contracts to make the computation of ZK-SNARKs more efficient.

2.3. Zero-Knowledge Proving Schemes

Developed first by Goldwasser, Micali und Rackoff (1989), zero-knowledge (zk) proofs refer to the set of cryptographic proofs that convey no additional knowledge other than the correctness of the proposition in question. In zk proofs, there are two agents, a prover and a verifier. The prover wants to convince the verifier he is in possession of a secret, without disclosing the actual value of such a secret. For a zk proof to work, it needs to fulfill the following (Blum, Feldman und Micali 2019, 336):

- **Completeness.** if the statement is true, then an honest verifier can be convinced of it by an honest prover.
- **Soundness.** If the prover is dishonest, they can't convince the verifier of the soundness of the statement by lying.
- **Zero-Knowledge.** If the statement is true, the verifier will have no idea what the statement is.

Zk proofs might sound at first counter-intuitive and hard to understand. The example of the "cave of Ali Baba"[17] is frequently used to portrait the essence of zk proofs. There is a circular cave with a door at the back, which only opens with a secret password. The prover wants to convince the verifier he knows the password without revealing the secret word to the verifier. The prover enters the cave first and takes one of the paths that leads to the door. The verifier, not knowing which path the prover has taken, situates on the point where the paths diverge and yell to the prover from which path he should come back. If the prover knows the password, the prover will come from the side specified by the verifier, regardless of the path he previously took. If the prover doesn't know the password, he will come from the correct path if he previously took by chance that path. At random, this probability is 50%. If the process is repeated n times, the chances of a dishonest prover coming every single time form the path indicated are of 2^{-n}. The verifier

16. https://sos.wv.gov/FormSearch/Elections/Informational/West-Virginia-Mobile-Voting-White-Paper-NASS-Submission.pdf
17. https://cryptowiki.net/index.php?title=The_simplest_example_is_the_cave_of_Ali_Baba_(Ali_Baba)

could repeat this process until he is confident enough that the prover is telling the truth. The proof is complete and sound, as the process is repeated a high number of times the probability of the prover coming every time from the correct path by chance is almost 0. It is also zero-knowledge because, after so many trials, the verifier still doesn't know the password.

2.3.1. ZK-Snark

As in the previous example, early zk proofs involved the prover and the verifier to exchange information over multiple rounds. The proofs required of synchronous communication which is incompatible with the asynchronous nature of the online world. Bitansky u. a. (2012) first introduced the term "Zero Knowledge succint non-interactive arguments of knowledge" (zk-snark), to refer to zk proofs in which no interaction is needed between prover and verifier. Succint stands for conciseness, the proof can be verified in milliseconds, and the length of proof can be easily transmitted over the internet, while non-interactive means that the proof is sent in a single message.

zk-snark consist of the 3 different algorithms: G, P, V (Petkus 2019, 61). G is the setup function, P is used by the prover to generate the proof, and the verifier generates v to assess the correctness of the proof. G takes the proof F that needs to be transformed to a zk-snark and a parameter w, which must remain confidential under any circumstance, as inputs, and outputs pk and vk to be used respectively by P and V. w is also referred as *toxic waste*[18], since an attacker who gets access to it could generate fake valid proofs. Fake valid proofs are proofs that assess to TRUE although the prover might not possess the required knowledge. With zk-snark the setup phase is of critical importance. Additional security measures should be taken in order to protect w from being leaked. The development team of Z-cash, a cryptocurrency using zk-snark to shield private transactions, created and documented a formal setup ceremony[19] involving burning the laptops used in order to ensure that the toxic waste was completely destroyed. The generation of zk-snarks involved extensive computations, which some have defined as "moon math"[20]. The project Zokrates aims to bridge the knowledge gap needed to generate zk-snarks.

2.3.2. Zokrates

Zokrates[21] tool-box allows to write and deploy zk-snark proofs on Ethereum easily. Zokrates provides a high-level language to write proofs and exports the verifier program as a smart contract. The Zokrates language is very similar to C++ or Java, allowing anyone with basic knowledge of programming to generate their own zk-snark and build around

18. https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6
19. https://z.cash/technology/paramgen/
20. https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649
21. https://zokrates.github.io/

them Dapps. Zokrates language allows to specify which inputs of the proof should remain public and which should be obfuscated behind the zk proof. Zokrates proof generation consists of five steps, which workflow is consistent with the standard procedure to generate zk-snark previously explained. These steps are:

- **Compilation.** Compiles zokrates high-level language into arithmetic circuits.
- **Setup.** Generates a trusted setup from the arithmetic circuits. This step outputs two keys, the proving key needed by the prover to generate proofs, and the verifier key required to create a verification contract. The setup is by far the most computationally intense step.
- **Verifier export.** the command creates the verifier smart contract in solidity from the verifier key. The verifier is then in charge of deploying the contract.
- **Witness generation.** A witness is a valid combination of inputs for which the arithmetic circuit asserts to true. Following the signature specified in the zokrates program, generate-witness returns a boolean for the given certificate.
- **Proof generation.** With the proving key and a valid witness generates a SNARK from the arithmetic circuit. The proof is stored with a fixed structure that is independent of the signature of the program. The only recognizable figures in the proof are the variables set as public inputs in the zokrates program.

3. Related Work

There has been extensive research done in the field of private and verifiable voting systems. However, the field has remained mainly theoretical as these systems use advanced cryptographic schemes that few people understand. The first systems relied on homomorphic encryption and mix-nets to achieve privacy and tamper-proof tallies for verifiability. Firstly introduced by Cohen und Fischer (1985), homomorphic encryption is a form of encryption that allows performing operations on encrypted data and re-encryption. Homomorphic encryption allows to perform operations on public keys without access to a secret. Initially proposed by Chaum (1981), mix-nets are server networks where each server obfuscates the relation between its inputs and outputs by applying a random permutation. The shuffling process to achieve anonymity of mix-nets has been extrapolated to other methods and is nowadays commonly referred to as “mixing”. A notable project in this group was the JCJ system (Juels, Catalano und Jakobsson 2010), which was the first known coercion-resistant i-voting system. With an in-person voter registration phase prior to the election, a proof of eligibility could be re-used to create fake profiles, which

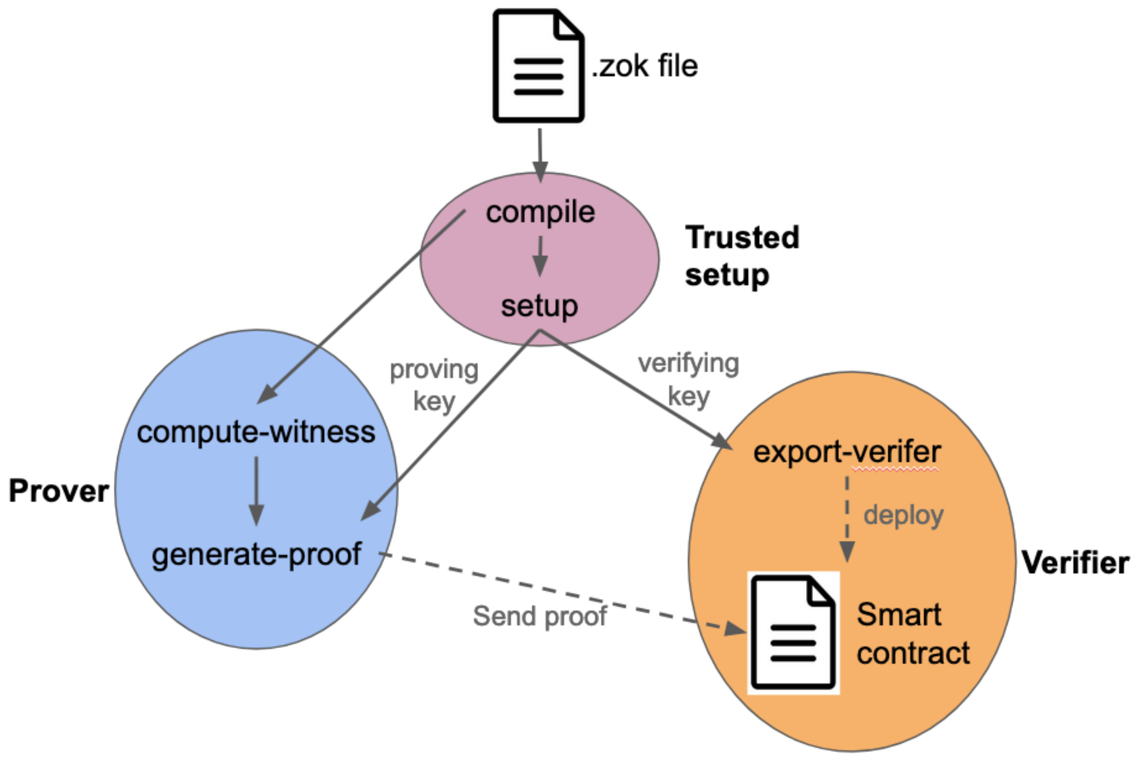


Abbildung 2.1.: Zokrates proof generation workflow

could only be distinguished by the trustees (Ali und Murray 2016, 25). These systems were built on very refined protocols based on multiple cryptographic primitives. However, These projects often made assumptions, which practicability can be questioned. Moreover, the complexity of such protocols made it very difficult to discover possible security flaws within and between system components. The risk of hidden single points of failure was still latent, while the voting process, which in principle simple, was highly over-engineered.

The irruption of blockchain technologies had a big impact on the research field as blockchains provide tampered-proof tallies and solve the double spending problem by design. The code homogeneity and fault tolerance of distributed ledgers also reduced significantly the complexity required to create verifiable systems. The focus thus shifted to make blockchains more private. Monero[1] became one of the most popular cryptocurrencies by being one of the first offering untraceable transactions through linkable ring signatures. Multiple proposal (Wang u. a. 2018, Lai und Wu 2018, Wu 2017) suggested using ring signatures in the context of elections. However, Rivest, Shamir und Tauman (2006) showed that the signature size grows at least linearly with respect to the ring size, limiting its use to small scale environments. This last fact led the Ethereum community to focus on providing support for zk-snark over ring signatures in the future.

1. https://www.getmonero.org/

Takabatake, Kotani und Okabe (2016) were the firsts to suggest zk-snark in order to provide anonymity in elections. They proposed a voting system where voters exchange Bitcoins tokens to a common pool of Zerocoins (Zcash predecessor) and converted back to anonymized BTC tokens (Figure 3.1). The functionality of exchanging the currencies resembles, in a distributed manner, the functionality of previous mixing-nets. Tarasov und Tewari (2017) wrote a protocol based on Zcash. The implementation achieved anonymity at the expense of transparency. User's identity could not be extracted from their votes, but neither the results of the elections could be traced publicly. Moreover, the proposal relied on an off-chain administration system to extend the functionality of the transaction system to an election scenario, creating a single point of failure in the system. These early protocols were written at a time where Ethereum was still nascent. The protocols rely on strict human behaviors to guarantee full anonymity. These steps were mostly mechanical processes that could have been easily automated through smart contracts. McCorry, Shahandashti und Hao (2017) proposed a boardroom voting scheme for Ethereum leveraging zk-snark, although the approach provides solid robustness against attackers, its scope is reduced to small groups as all voters are required to cast a vote in order to calculate the tally. All of the proposals discussed in this section require multiple interaction rounds between the election organizers and voters, countering the main purposes of i-voting systems, convenience.

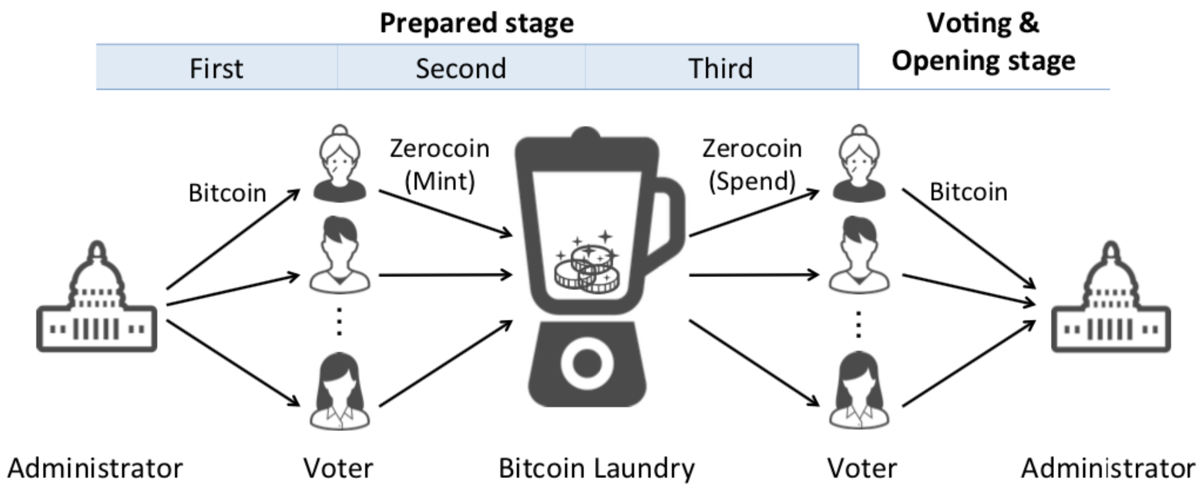


Abbildung 3.1.: Protocol suggested by Takabatake, Kotani und Okabe 2016 using Zerocoin

Early in 2019, Vitalik Buterin (2019) underlined the importance of taking ä first step toward more privacyön Ethereum. Using the mixing concept used by mix-nets, Buterin proposed a Minimal Mixer(MM) smart contract prototype using zk-snark. With which users could send fixed-quantity transactions (tokens) to the MM and receive the tokens in newly created accounts back. The MM would allow replicating the protocol proposed by Takabatake, Kotani und Okabe (2016) in Ethereum. However, the MM display on an election scenario the following weaknesses: Firstly, voters have to link their identities to a blockchain address in order to receive a ballot. Election organizers could access personal information about the voters by looking at the history of the addresses provided. Secondly, the utilization of the MM can not be imposed nor embedded in a workflow. Voters could still cast a vote without the use of the MM, and their preferences would be disclosed to the organizers and indefinitely recorded on the blockchain. The identities of abstaining voters would also be exposed. Lastly, the degree of anonymity achieved is highly dependent on the number of voters who use the MM within a specific time frame. The MM requires a high degree of coordination and trust among voters, and the costs of remaining anonymous are bared by the participants.

Regardless of their effectiveness, it can be argued that all of the proposals discussed in this chapter disregard simplicity and convenience in their pursue of privacy and verifiability. These two factors could well explain why e-voting adoption has remained low so far. Complex processes require a high level of trust and can raise skepticism if results can not be easily explained. Inconvenience increases the burden of change for a process that performs reasonably well.

4. System Design

In this protocol, a modified version of the MM is introduced, called the Token Pool(TP). Instead of having voters in charge of mixing their ballots (Figure 3.1), all the votes are deposited in the TP by the election organizer, and voters collect their ballots in a self-service fashion.

4.1. Design Choices

Since no voting protocol can fulfill all the desired properties, an ideal voting system should possess, the context in which the election takes place is of critical role when designing a voting protocol. In our case, the goal is to develop a platform that could be used by local governments and allows citizens to decide on specific matters regularly. It is assumed that the system will be used by governments in established democracies, so institutions are stable and trusted, the rule of law is followed, and punishments for deviant behaviors are applied effectively. In such a setup, where the likelihood of bribery or coercion is reasonably low, it could be argued that the benefits of a transparent voting system overweight the strictest forms of anonymity. Still, ballot secrecy should be guaranteed.

With regard to the above scenario, I now proceed to specify the requirements a voting system should have. First of all, the system should be complete and sound, meaning the ballots are counted correctly, and the system is secure against third party attacks. Secondly, the system should keep the ballots private while ensuring that only eligible individuals are allowed to vote. Moreover, the process should be transparent and efficient, validating the result of the election for every stakeholder in a reasonable amount of time. Lastly, the process should require the least amount of effort for voters to encourage turn out. Leveraging and smart contracts on Ethereum, this chapter presents a voting protocol that fulfills such requirements.

4.2. Overview

There are two parties taking part in the protocol, the election organizer and the voters. The election organizer wants to ensure that only eligible individuals are allowed to vote. For example, all residents of a town over the age of 18. The organizer has set up a way to verify which individuals fulfill the requirements and has a tally of all verified voters previous to the election. Voters want to cast their ballots anonymously while being able to verify the results of the election by themselves. The organizer knows who the eligible voters are but has no knowledge of their blockchain address, so he cannot distribute the ballots. Instead, the organizers transfer all the votes to the TP. For a voter to receive the ballots from the TP, she will need to prove the TP through a zero-knowledge proof that she is part of the set of eligible voters without disclosing exactly which one (Fig 4.1). We will refer to this identity proof as “Proof of Eligibility” (PoE). The TP provides k-anonymity (Samarati und Sweeney 1998), meaning that an address sending a valid PoE to the TP only discloses that the address is owned by an eligible voter but not by whom. In order to avoid double-spending, the TP commits all received valid proofs and checks that none of them are used more than once. Once a vote is received, voters could send their votes to the candidate of their choice and make sure their votes are counted correctly.

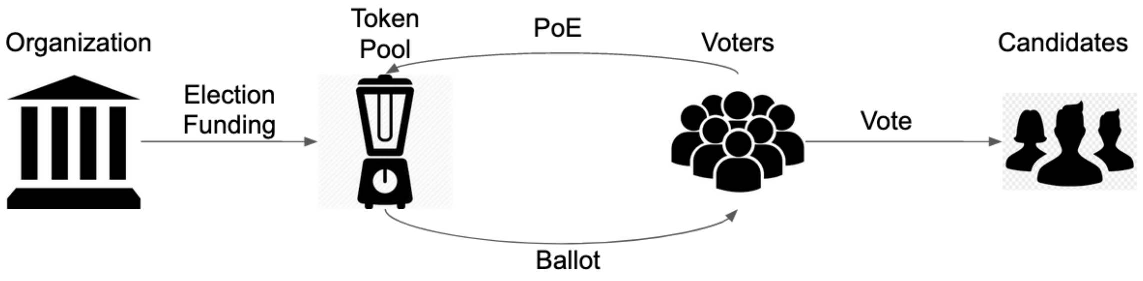


Abbildung 4.1.: election workflow

4.3. Election Stages

The process has been broken down in three main stages, and each step will be explained in detail in this section. There are three agents involved in the election, two of them being human controlled accounts, An institution I and a set of n eligible voters $V = \{V_1, V_2, ..., V_n\}$. The third agent is the smart contract token pool TP. I is responsible for calling the election, supplying the votes, and consolidating the tally. TP receives the election ballots and distributes a ballot to every user, which proved to be part of the eligible voters. V collect their votes and send them to the candidate of their choice (see Figure 4.2).

4.3.1. User Registration

Even though this stage precedes the actual election, the setup phase plays a critical role in ensuring ballot secrecy. It begins with I, establishing the set of rules which determine whether an individual is qualified to take part in the election, as well as the candidates participating. For an individual to be part of V, he has fulfilled the criterion set up by I. The assessment of eligibility is not part of the protocol, and the full responsibility of the task falls under I responsibilities. Depending on the risk of users impersonating fake identities, the verification process could range from simple a simple internet sign up to in-person identification.

Once verified, each voter needs to follow the following steps. Each V_i generates a public (pk_i) and secret key (sk_i). V_i shares pk_i with I and keeps sk_i secret. I publishes all the pk_i and creates an election commitment R from all the pk_i. V can then assert whether the registration was successfully completed by checking their own pk_i in the tallying board. It is of vital importance that V_i uses an asymmetric encryption scheme to register in the election. Otherwise, I could potentially impersonate any V_i. It is ultimately sk_i that compels the verifier of the PoE that the sender generated the pair of keys.

4.3.2. Election Deployment

Since I possesses no information about the addresses of V, I deposits the total amount of votes in the TP. The TP can verify the correctness of a given PoE and record every valid proof received. The PoE is based on the well-known set-membership problem. The set-membership problem can be stated as follows: Given a commitment to a value σ, prove in zero-knowledge that σ belongs to some discrete set Φ (Camenisch, Chaabouni und Shelat 2008, 234). Applying to our specific case, the verifier has to ensure that the person who is requesting a vote belongs to V, without disclosing her identity. Thus, given the proof of eligibility π, the verifier must be able to assess the following:

1. π proves that the voter is in possession of a sk that construct pk.

2. π proves that pk is used to derive R.

4.3.3. Vote Authentication

To receive a ballot, an eligible voter has to create a valid PoE. For that, she needs to provide her sk_i and pk_i The possession of a valid secret to one of the eligible public keys, proof that the person was the one generating the pair. To prevent a voter from receiving multiple votes with the same eligibility proof, the TP commits every valid proof received and asserts first that the proof was not previously used. Once the TP verifies the correctness of the proof, the TP sends a vote to the prover's address. Note that the TP sends the vote to whoever sends a valid PoE. However, neither the TP nor any other party is able to determine the prover's identity from all the eligible voters as the keys are obfuscated behind the PoE. Once V_i has received the vote, she can send it to any candidate of her choice. V_i can verify that her vote was count correctly by looking at the transactions on the blockchain.

It must be noted that to guarantee the anonymity of V_i, the address from which V_i sends the PoE should not have been used before in any other transaction, nor it is linked to any personal data and ideally won't ever be used again. Assuming the last requirement is fulfilled, the only statement that can be made about an address receiving a ballot is that it belongs to an eligible voter.

5. Implementation

In order to assess the viability of the proposed protocol, the app *Snarky* has been developed as a proof of concept. The assumptions taken on the previous chapter have been thoroughly scrutinized while implementing *Snarky*. The process helped to refine the model and identifying technical bottlenecks not taken previously into account. The technology stack used to develop *Snarky* relies mainly on Zokrates to generate the zk-snark PoE, Ethereum smart contracts as the application's back-end, and the javascript framework ReactJS[1] to provide for *Snarky*'s front-end. The knowledge acquired in the development phase has been summarized around the three components.

1. https://reactjs.org/

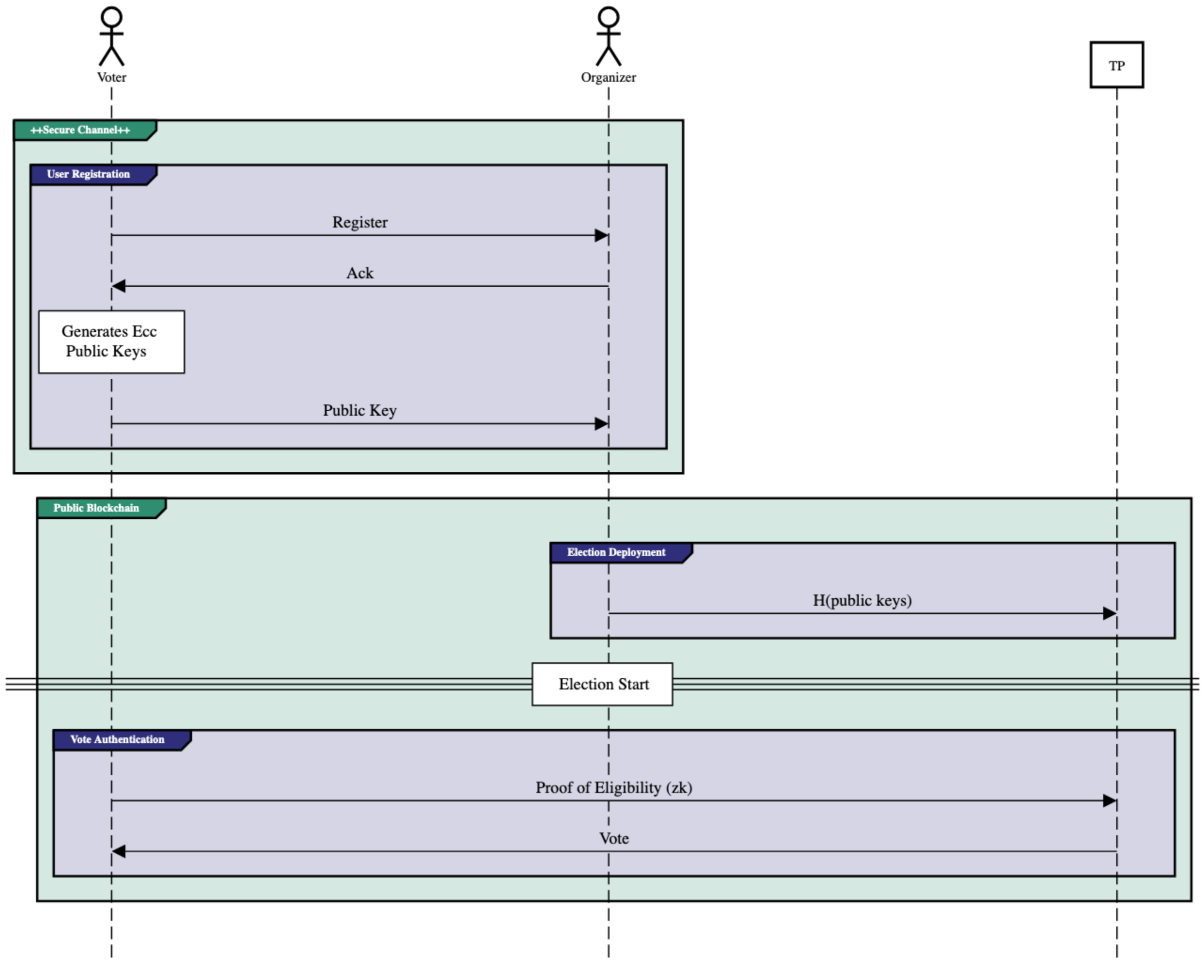


Abbildung 4.2.: Sequence diagram of protocol stages

5.1. PoE generation

The PoE asserts two arguments: first, the prover knows one of the public keys that formed the election commitment; second, the prover knows the secret that generated the public key. Number two is the convincing argument that the prover is the person that registers for the election. Without the secret, I could theoretically impersonate any voter. To ensure secrets can not be obtained from the public keys, *Snarky* only accepts public keys generated with an elliptic curve cryptography (ecc) scheme. In ecc, the public key consist of two points, point x and point y, thus the pair of keys is, in fact, a triplet. To proof the first argument, *Snarky* creates a (binary) Merkle tree from all hashed public keys. The Merkle root of the tree is the election commitment R. An ecc triplet and a Merkle proof that results in R are needed to generate a valid witness. The length of the Merkle Proof is equal to the depth of the Merkle tree, which in turn depends on the number of eligible voters. so for an election with n voters, the number arguments needed to generate a witness is then equal to $3 + \lceil \log_2(n) \rceil$.

The number of conditions the PoE has to verify increases every time the number of voters doubles. This means that a different verifier is needed for every Merkle tree depth. However, few verifiers are needed to encompass any election size. A verifier with a tree depth of 33, could accommodate more voters (2^{33}) than the current global population (estimated to be over 7 billion). Figure A.1 in appendix A shows the code template to generate PoEs in Zokrates. Having R as an argument in the PoE allows reusing the same verifier for all elections of the same size range. Moreover, by making R a public argument in Zokrates (Figure A.1), it can be easily asserted that a PoE corresponds to a certain election without the need of calling the verifier.

The only issue encountered with Zokrates, is that it currently offers support for the G16 proving scheme (Groth 2016) on the browser. The G16 proving scheme is known to be subject to malleability, meaning that an attacker with access to a valid proof could easily generate fake valid proofs. This problem would be solved by using malleability-resistant proving schemes such as GM-17 or PGHR13 (Groth und Maller 2017), which at the moment are only supported by Zokrates command-line interface (cli). Since the PoE is generated on the browser when voters insert their keys, *Snarky* uses at the moment the G16 scheme. An attacker could a PoE already verified and multiply it by a number in order to generate fake PoEs. Since the problem is not case-specific, but affect a large part of the Zokrates community, it can be assumed that support for other proving schemes will be introduced. This vulnerability does not conflict with the purpose of the study. However, the issue should be kept in mind when implemented in a real-life scenario.

5.2. Back-End

Most Dapps frequently rely on additional servers to carry out many of the application's computations for the sake of saving up resources. The addition of a centralized server adds to the system numerous security concerns and a single point of failure to the system. Having additional servers overviewing certain tasks also obscure the electoral process, as voters would have to trust the servers on doing what is supposed to. That is why the back-end of *Snarky* relies entirely on Ethereum smart contracts. The data is publicly accessible to anyone by design, to avoid edition and promote public scrutiny. Running many operations on Ethereum can, however, turn costly as every write operation requires gas and gas is expensive. So that the use of *Snarky* remains affordable, the functionality of the back-end has been optimized to reduce as much as possible gas consumption. The functionality of the back-end is split in the following smart contracts:

- **Verifier.** The verifier contract is solely responsible for verifying the correctness of the PoE. The verifier contract has been generated by Zokrates (see Figure 2.1). As mentioned in the previous section, elections with different Merkle trees need different verifiers. But, except for certain parameters used to assert the correctness of the inputs, the code and structure are identical among verifiers with different tree depths. The succinct property of zk-snark makes it possible to create an abstract verifier class and instantiate a verifier for each election size. Instead of having different contracts for different election sizes, there is one contract that serves all, and thus only one contract needs to be compiled. The different parameters can then be securely stored on Ethereum. This implementation manages efficiently and in a cost-effective way, the process of creating and managing the multiple verifiers for different election sizes. Verifiers need to be deployed only once and can be re-used ever after. Moreover, the deployment time goes down to milliseconds as the contract is already compiled.

- **Election.** In *Snarky* the election contract is the counterpart of the TP in the protocol. The contract contains all the necessary data for the correct functioning of the TP: the election commitment R, the list of eligible voters, the verifier's address, the list of candidates and it's respective votes. At deployment, the election will register itself under the address of the transaction's sender. For every PoE received, the election asserts that the public input matches the election root and ensures that the proof has not been previously used. Once this is verified, the contract calls the verifier to assert the correctness of the proof. If passed, the election will register the proof, update the vote count of the candidates, and send some ether to the message sender.

- **Register.** The register contract keeps track of every election deployed on *Snarky* and maps its address to the address from which the election was deployed. The contract adds an election to an already known address or creates a new mapping if the address is creating an election for the first time. The register also keeps track of the deployed verifiers to ensure they are only deployed once. The register also supports the search feature by returning all the elections deployed by a given address.

In the protocol, Once the correctness of the PoE is verified, the voter would receive the voting token. Since the transfer of tokens in Ethereum can be regarded nowadays as trivial, the development of the voting token has been omitted in *Snarky* for simplicity purposes. Instead, the voting contract keeps a counter of the votes for each candidate. The voting token has an important role in the protocol as with it; voters can verify that their votes were counted correctly and assess the outcome of the election independently. However, the concern of securely exchanging transactions has been addressed since bitcoin inception and is considered for the purpose of this study trivial.

A topic not touched by the protocol is the gas costs associated with every transaction in the network. In Ethereum the gas fees are charged to the transaction's sender by design. This means that users have to pay the transaction fees from their own funds where the PoE is sent to the election contract. The problem with users paying to receive their ballots has deeper implications than a question of social justice or mere finances. If voters need to pay gas to receive their ballot, they must have bought ether in the first place. Nowadays, many countries require cryptocurrency exchanges to impose strict identification procedures on users for taxation purposes. So, in the end, each voter's level of privacy will depend on how the voters individually acquired the gas for the transaction. Another approach to this problem would be to implement a gas relay interface in solidity. The gas relay interface[2]

2. https://github.com/ethereum/EIPs/pull/2454

would act as a middle man between the voters and the election contract. Users would call the gas relay, and the gas relay would perform the transaction in the name of the voter. In the next chapter, it will be explained why this approach is not a suitable solution.

5.3. Front-End

Apart from providing the user interface (UI) to interact with the smart contracts on Ethereum, the front-end of *Snarky* is responsible for taking the heavier computation tasks, such as the PoE generation. Through the UI the two main actions of the protocols can be performed: deploying an election and casting a vote. Although in the protocol two different roles where defined, an institution and a voter, these roles are merely symbolic. Anyone with an Ethereum address could be both election organizer and voter. For users to manage their Ethereum accounts securely, *Snarky* relies on Metamask[3] for the logging service. Metamask includes a secure identity vault, providing a user interface to manage your identities on different sites and sign blockchain transactions. Metamask browser extension needs to be installed before using *Snarky*. On *Snarky*'s homepage, Metamask will ask the user to log to their Ethereum account, if not logged. Before any transaction is sent to the Ethereum network, Metamask will pop on the user's browser, requesting her consent with the transaction cost breakdown. If accepted, the user will pay the transaction cost, and the transaction will spread over the network awaiting verification. The next two sections describe in detail the user paths and operations performed by the UI.

5.3.1. Election Deployment

On *Snarky* any Ethereum address can create an election. The legitimacy of the results will depend on the members' willingness to accept and abide by the collective decision. To become an election organizer, the only requirement needed is to have sufficient funds to distribute the votes, a list of candidates, and the list of public keys of all verified users (Figure B.2). The voting app will first compute the election commitment R. R is the Merkle root of all the hashed public keys from eligible voters. Before deploying the election contract, *Snarky* queries the register in case a verifier with the same tree depth was already deployed. If a verifier was deployed, the election is deployed with the address of the existing verifier. If not, first, a verifier with the corresponding parameters to the election size is deployed, followed by the election contract. Meaning the first organizer of an election of a certain size will incur an extra cost of deploying the verifier of the election, additionally to the cost of deploying the election itself. However, the verifier could be reused for any posterior election of the same size. Deploying an election on Ethereum takes a matter of seconds at *Snarky*.

3. https://metamask.io/

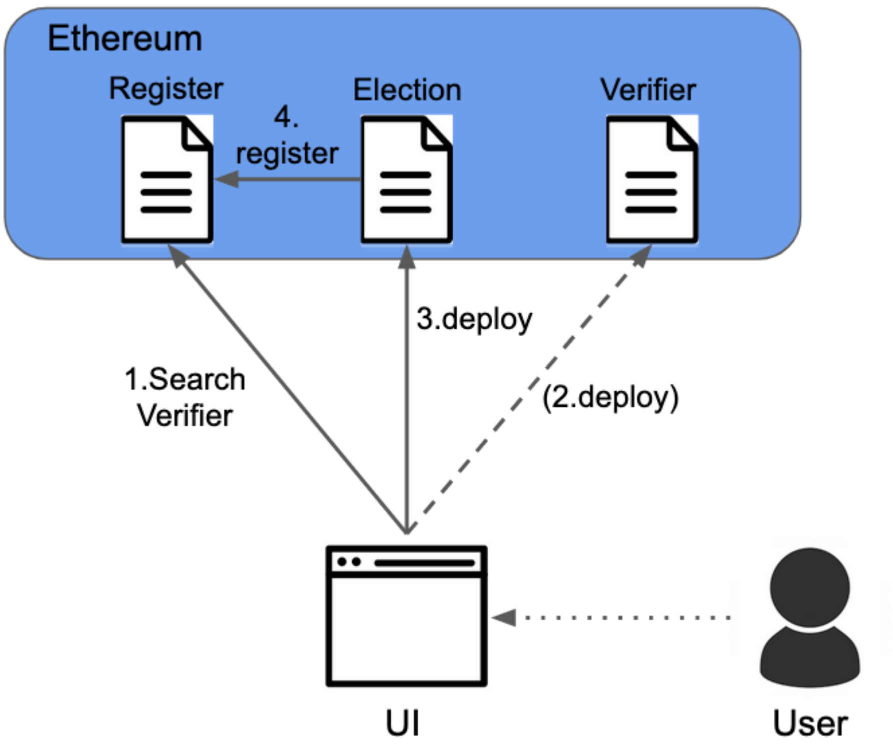


Abbildung 5.1.: Election deployment workflow

5.3.2. Voting

It is assumed that election organizers will run multiple elections with a stable electorate over time. Voters won't know the address of the specific election but would know the address of the institution. *Snarky* supports a search feature allowing voters to search by the organizer's address (Figure B.1). The search result returns all the elections associated with that address. Once the appropriate election is found, a voter will have to input her choice candidate and the ECC triplet of private and public keys used in the registration phase. With the keys, *Snarky* builds the Merkle proof that results in R. After verifying that the certificate of ECC keys and Merkle proof creates a valid witness, the proving key for the PoE is computed, and the PoE is generated (prover path Figure 2.1). The proof is then sent to the election smart contract through a transaction. Shortly after, the counter of the candidate selected should increase in case the proof is correct (Figure B.3).

The proving key generation is by far the most time and energy-consuming action on *Snarky*. Alone for an election with eight voters, the generation of the PoE takes around 9 minutes. In contrast to the verifier contract's deployment, the proof generation can not be so easily optimized to run efficiently. The reason for that is the file size of the proving keys needed to generate a PoE. Although the resulting PoE is very concise, the proving key grows linearly with the proof's complexity. The deeper the election Merkle tree, the bigger the proving key of the PoE. Figure 5.3 shows, the proving key size grows by 50 MB every time the number of voters double. So even though the proving key experience

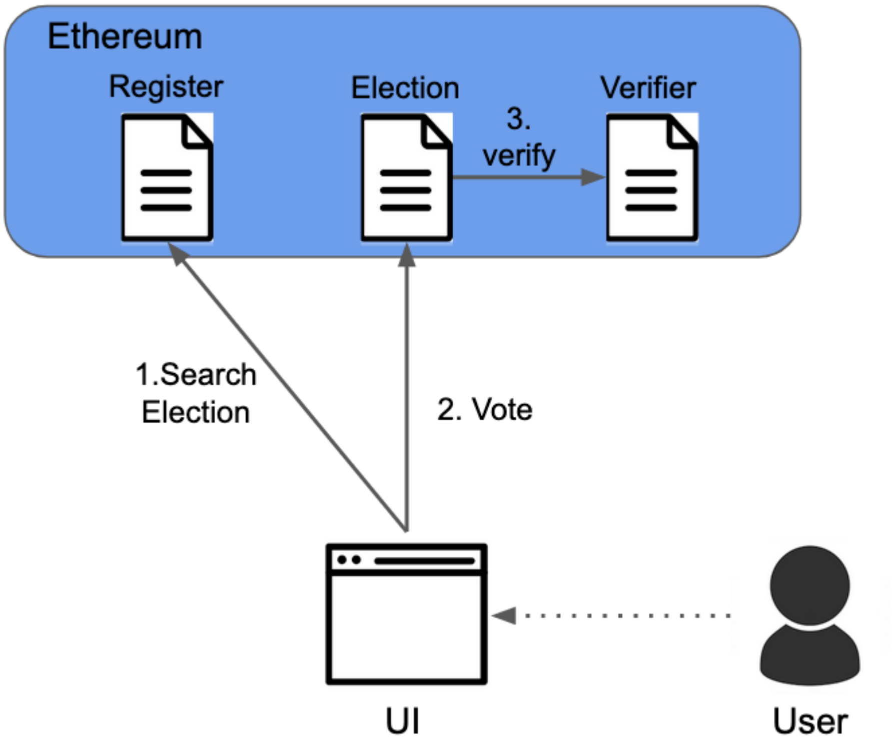


Abbildung 5.2.: Voting workflow

a logarithmic grow to the number of voters, an election of 1024 voters would take over 600 megabytes of disc memory. A file of such size can not be stored on Ethereum nor a file in the front-end. A possible solution would be to store the different proving keys is a distributed file storage system such as the Interplanetary File System (IFPS)[4] or Swarm[5]. Apart from speeding up the PoE generation process (assuming a good internet connection), the storing of the proving key in IFPS could also enable the implementation of more robust proving schemes like the GM-17. The proving keys and verifier contract for the PoEs would be generated with Zokrates cli instead of in the browser as of now. The addition of a distributed file storage to *Snarky* has not been implemented for this study but would be recommended for future works.

4. https://ipfs.io/
5. https://swarm-guide.readthedocs.io/en/latest/index.html

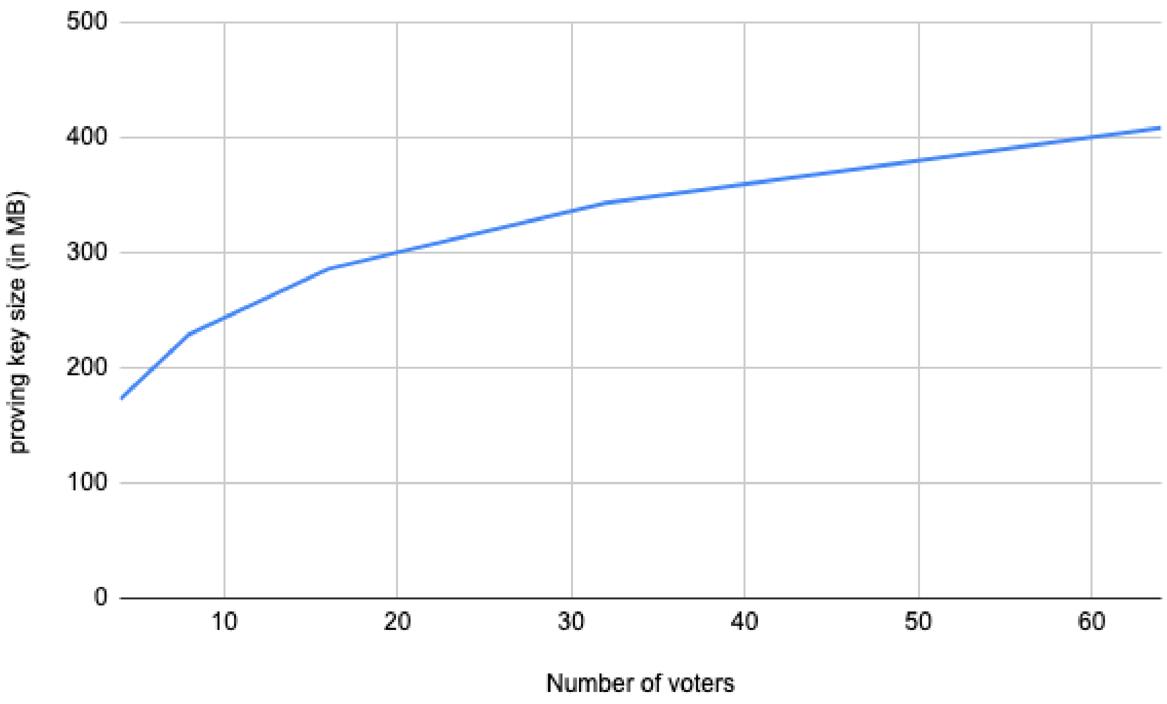


Abbildung 5.3.: PoE's proving key size vs size of electorate

6. Evaluation

The paper proposes a novel approach for distributing tokens to verified individuals anonymously. The assumptions taken at the protocol's definition have been later tested when developing the application *Snarky*. Thus, the evaluation will first focus on the protocol, followed by a cost analysis of the transactions of *Snarky* on the Ropsten network. Lastly, a vector attack will be simulated to discuss the significant threats facing *Snarky*.

6.1. Protocol

The paper suggests a protocol where newly created addresses can proof eligibility without disclosing any personal information through a zero-knowledge set-membership proof. So far, the address used to send the PoE is not linked to personal data, the protocol is *private* because nobody could link a casted ballot to a physical identity. It also is *eligible*, only persons who have been added to the tally can create a valid PoE and thus can cast a ballot. The protocol fulfills *uniqueness* as only one vote is issued for a valid PoE. It is *individually* and *universally verifiable* as voters can audit the correctness of their votes and the number of votes collected by the candidates. It is *efficient* because the actions to be performed by the voters are minimal. Voters just need to register for the election (unavoidable for every kind of election), collect their ballot from the TP, and transfer it to the candidate of their choice. *Receipt-freeness* was purposely sacrificed in favor of *verifiability.* Thus the protocol is not *coercion-resistant*, as users become a transaction confirmation once the transaction has been added to the blockchain. The protocol can be subject to bribery and coercion; a voter could sell her private key away for private gain. The weakness against coercion is not only inherited in this protocol but a common characteristic of most remote voting system. It is hard to assess whether a voter is being coerced in her own home. For this reason, this protocol is only advisable, where the threat of bribery and coercion is low. The protocol does not fulfill fairness either as voters could check on real-time the confirmed votes added to the blockchain.

Although technically very similar, the TP holds multiple advantages over the MM on elections. First, the TP requires less effort and, thus, fewer transactions. There is a single supplier who wants to distribute assets uniformly among a specific set of users. The election organizer sends one transaction with n tokens to the TP, instead of one token to each of the n eligible voters and back to the MM. In an election with n eligible voters and no abstention, the number of transaction made with the MM would be of $5n$ (receive the token from the organizer, send the token to the MM, send the PoE to the MM, receive the token from MM, and send the token to a candidate; for each voter). With the TP, the total transactions would be $3n+1$ (the last 3 phases for each voter plus the initial deployment of the organizer). That is an almost 40% reduction in transactions while maintaining all the functionality. Second, the TP protects the identity of abstaining voters. Since the PoE prevents finding out who has voted, it also prevents assessing who has not. The anonymity set of the MM is, in best case, equal to the anonymity set of the TP, in case of no abstention, and lower elsewhere. Lastly, The TP allows the election to happen in a non-interactive manner. With the TP voters have only to collect their ballots and send it to their preferred candidate, once the election has started. This requires little behavioral changes from the voter's side, as these actions already take place in paper ballot elections. Both the TP and the MM use the same mechanism to obfuscate the token holders' identity, pooling together a set of uniform assets and distributing them to unrelated addresses. However, the TP is based on premises better aligned with the logic of zk proofs. An honest verifier (I) has to identify faithful provers (eligible voters) among potentially many attackers.

6.2. Testnet Analysis

With the purpose of estimating the cost of using *Snarky*, the application's contracts were deployed on the Ropsten network along with multiple elections. For the tests, a gas price of 20 Gwei was chosen based on Metamask guidelines. Figure 6.1 breaks down the cost of the different actions on different elections. The graph shows that the cost of deploying the verifier contract and casting a vote remains constant at approximately 0.025 eth and 0.0065 eth, respectively. The prices are consistent with *Snarky*'s implementation as the structure of the PoE, and the verification code is not affected by the size of the election. The cost of deploying an election grows linearly with the number of voters, as the election contract contains the list with all eligible users. The cost of deploying the election is strictly higher as the cost of deploying the verifier since the verifier is a dependency of the election contract. Deploying an election has a fixed cost of 0.024 eth and a variable cost of 0.0004 eth for every eligible voter. Taking the 2-month average price for ethereum[1] of 150 €, casting a vote on *Snarky* would cost approximately 0.97 €.

In order to better assess cost inquired by *Snarky* on a real-life scenario, the initiative *100% Tempelhofer Feld* has been taken as an example. Almost 2.5 million citizens were called to the urns in Berlin in 2014 to decide about the future of one of the city's landmarks. In the initiative, 1.15 million citizens participated (Berlin 2014), and the costs of the initiative were budgeted at 1.6 million euros [2]. On *Snarky* the same election would have cost almost 1.27 million euros. The shy improvement in costs reflects two underlying problems, Ethereum's high gas prices and the lack of transparency in traditional democratic processes.

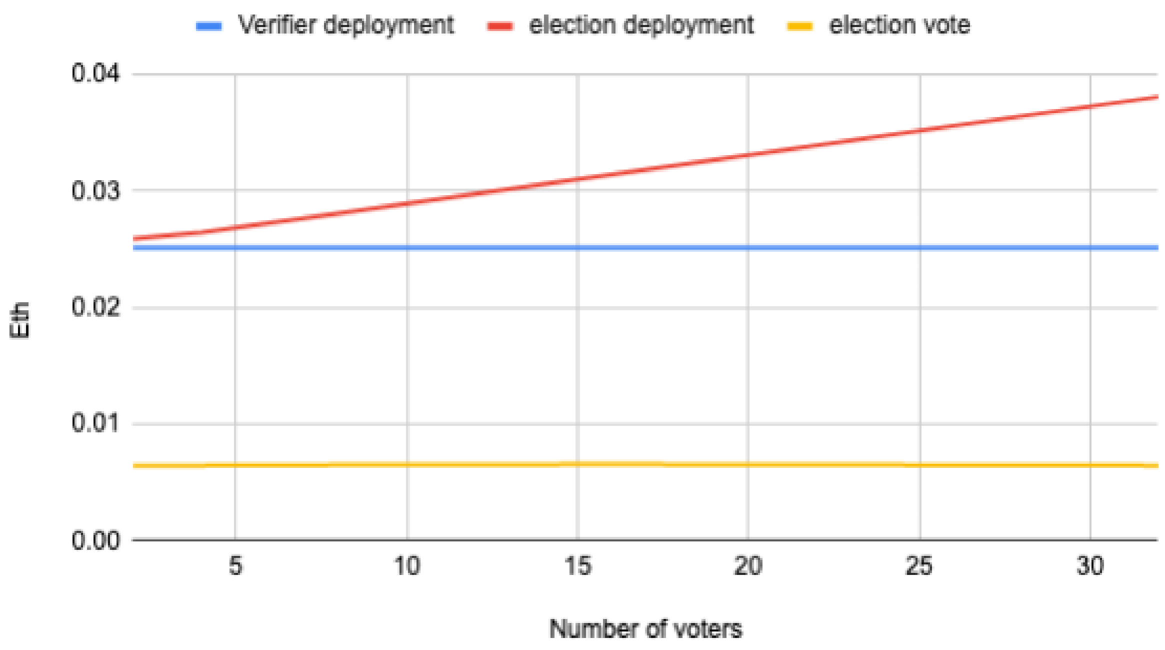


Abbildung 6.1.: PoE's proving key size vs size of electorate

1. 01.05.2020 taken as the reference date. source: https://cointelegraph.com/ethereum-price-index
2. Although this number is said to be in reality much higher as the referendum was organized together with the elections to the European Parliament.
source: https://www.tagesspiegel.de/berlin/volksentscheide-in-berlin-die-direkte-demokratie-kostet-millionen/11192724.html

6.3. Security Analysis

In *Snarky*, as with all i-voting systems, an internet connection could potentially lead an attacker to gain unauthorized control over the user's device. A compromised device could alter the right functioning of the application or replace the transaction's content without the user's knowledge. A coercer could influence the outcome of an entire election by infecting a large number of voters' devices. The number of potential threats coming from the internet surpasses the scope of this study. For the sake of the study, it will be assumed that the devices used to vote are secured and the focus will be shifted to the potential vulnerabilities around the protocol and the Ethereum network.

As with any application using zk-snark, the main concern in *Snarky* is the extraction of the toxic waste. With the toxic waste, an attacker could create any number of fake valid proofs. Even with a strong proving scheme as GM-17, Zokrates makes the proof generation so simple that an attacker with a rough understanding of how the PoE is constructed could replicate the original program. In fact, the Zokrates source code for the PoE is embedded in the UI in *Snarky* in order to generate valid PoEs. Given a Zokrates program, the toxic waste of the zk-snark could be easily reproduced by running the compile and setup commands. The PoE source code could be removed from the UI by storing the pre-compiled proving keys in a distributed file storage system. However, the risk of the proof been backward engineered would still remain. The solution would be to add a random element in the setup of the PoEs and have and audited ceremony, similar to Z-cash trusted setup, to ensure the keys are completely destroyed and can't be recovered.

The protocol assumes that anyone can create a blockchain account and start making transactions. This is partially true, as Ethereum requires a gas fee in ether to send transactions. In most exchanges, the purchase of ether requires a strict identification process to avoid money laundry. Buying fresh ether from an established exchange would then defeat the purpose of the PoE as the account would be tightly connected to a physical identity. Initially, it was thought that a gas relayer would allow voters to send the PoE without having to pay gas. Nonetheless, gas relayers can be subject to a form of DoS attack, called "gas griefing"[3] attacks, in which an attacker would purposely send invalid PoEs with the purpose of depleting all the relayer's gas, hindering the verification of other votes. To avoid gas griefing attacks, identity-based relayers are used where only a limited amount of agents are allowed to call the relayer. However, the use of zk-snark identity proofs is not suitable for gas relayers. As the act of verifying the correctness of a zk-snark proof requires gas, the cost has to be either bared by the relayer, exposing itself to gas griefing attacks or by the transaction's sender, contrary to the relayer's purpose. The use of a gas relayer is, for our purpose, inappropriate. On the other hand, casting a vote on *Snarky* costs about 1 €, which can be considered an affordable price for citizens in established democracies. Moreover, making the transaction's senders pay for their own gas provides a financial disincentive for possible attackers. A payback system could also be implemented, senders of valid PoEs get a fixed amount of ether back to account for the gas previously

3. https://consensys.github.io/smart-contract-best-practices/known_attacks/insufficient-gas-griefing

paid. Eligible voters would get their ether back while deterring attackers. Nonetheless, voters should still pay special attention to use an address that is not linked to their real identity. Mixing ether from a frequently used account or purchasing ether in exchange for cash and depositing it in a newly created account would be enough to guarantee privacy.

As the price of voting is relatively low, gas prices could be set higher to shorten the confirmation of transactions. Apart from discouraging gas griefing attacks, setting high prices would also make the system more robust against "front-running"[4] attacks, unverified transactions being overwritten, or of 51% attacks. Another consideration to take into account is the value of the vote token. In order to discourage vote hijacking and abstention, the vote token should have no value, and its use should be bounded to specific elections or organizations.

7. Conclusion

The paper proposed a novel voting protocol that ensures ballot secrecy on public blockchains. The TP reduces by up to 40% the transactions needed in comparison with a mixer implementation. The TP also resembles more the process of traditional elections requiring less effort from voters. The TP provides higher anonymity than the MM. The TP does not involve new techniques; rather, it leverages well-known techniques to make the process simpler and more efficient. The reduction complexity results in a more robust system as security flaws are easier to recognize and, thus, prevented. Moreover, an easy to understand voting system increases its widespread and legitimizes its outcomes.

This study also posses one of the few fully decentralized voting apps known to date, which provides ballot secrecy. While developing *Snarky*, the study confronted technical issues not addressed in the protocol. The chances of an attacker replicating the toxic waste of the PoE in *Snarky* are reasonably high. The paper reasons why zk proofs are incompatible with identity-based gas relayers; and argues that a well-thought system of financial incentives could deliver better protection against possible attacks on smart contracts. Ethereum's high-priced gas makes *Snarky* quite costly compared to other e-voting solutions, but still economical compared to traditional elections. *Snarky* showed that the level of anonymity achieved by the protocol is limited by the extent to which an address is detached from its owner's identity. In Ethereum, mixing some ether previous to casting a vote or buying anonymously small amounts of ether would be sufficient to guarantee privacy.

4. https://consensys.github.io/smart-contract-best-practices/known_attacks/front-running

It is also worth noting the double standards on voting systems exposed by De Tullio, Romano und Vaccaro (2020). While electronic voting systems have to be precise, traceable and transparent, it is acceptable for paper ballot to be none of the previous. Online elections can indeed be subject to a broader range of attacks with severer consequences than their physical counterparts. On the other hand, online voting could enable new forms of representation on different organizational levels. At the same time, blockchains like Ethereum have yet to reach product maturity. The first phase of Ethereum 2.0 is expected to roll out later this year, with major improvements on transaction volumes, a new EVM, and a new consensus mechanism. The current limitations of Ethereum might be shortly obsolete. The conclusions of this study reinforce the idea that in the near future communal issues, neighborhood projects, and regional proposals could be all managed securely from a Dapp.

8. Future Work

Lastly, I mention shortly some cases for which the content of this paper might be relevant. Firstly, the proposal's effectiveness is highly hindered by the fact that it is very complicated to get ether anonymously, even in small quantities. Having alternatives in place to allow easy access to small amounts of ether to send transactions would benefit this proposal and the widespread usage of Ethereum's Dapps in general. More places where ether is sold in exchange for cash like atms[1] could be a solution. As previously mentioned, the protocol could be used for other purposes than voting; the PoE could be used for numerous kinds of logging mechanisms. This study also posses one of the few fully decentralized voting apps known to date, which provides ballot secrecy. Social entrepreneurs can learn from the experience gained in this paper. The research has mainly focused on replicating traditional elections with ballot secrecy in a decentralized way; however, this system could be extended to accept different voting rules, allowing to experiment with new forms of representation, such as liquid democracy. Lastly, this work exemplifies the potential zk proofs have in ensuring more privacy on the blockchain. Further research on zk-snark and zk-stark (zk-snark without trusted setup), as well as the development of tools such as Zokrates, could increase the adoption of techniques to make more decentralized, private and secure applications.

1. https://coinatmradar.com/

Literaturverzeichnis

Ali, Syed Taha, und Judy Murray. 2016. „An overview of end-to-end verifiable voting systems“, Bd. 173. CRC Press.

Atlam, Hany F., und Gary B. Wills. 2019. „Chapter Three - Intersections between IoT and distributed ledger“. Herausgegeben von Shiho Kim, Ganesh Chandra Deka und Peng Zhang, Advances in Computers, 115:73–113. ISSN: 0065-2458. https://doi.org/https://doi.org/10.1016/bs.adcom.2018.12.001. http://www.sciencedirect.com/science/article/pii/S0065245818300688.

Berlin, Landeswahlleiterin. 2014. „Pressemitteilung der Landeswahlleiterin vom 5. Juni 2014“. Geschäftsstelle der Landeswahlleiteri. https://www.wahlen-berlin.de/abstimmungen/VE2014_TFeld/presse/20140605VE.pdf.

Bitansky, Nir, Ran Canetti, Alessandro Chiesa und Eran Tromer. 2012. „From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again“. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference,* 326–349. ITCS ’12. Cambridge, Massachusetts: Association for Computing Machinery. ISBN: 9781450311151. https://doi.org/10.1145/2090236.2090263. https://doi.org/10.1145/2090236.2090263.

Blum, Manuel, Paul Feldman und Silvio Micali. 2019. „Non-interactive zero-knowledge and its applications“. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali,* 329–349.

Buterin, Vitalik. 2014. „A next-generation smart contract and decentralized application platform“. *white paper* 3 (37).

Buterin, Vitalik. 2019. „We need a first step toward more privacy“. Besucht am 25. November 2019. https://hackmd.io/@HWeNw8hNRimMm2m2GH56Cw/rJj9hEJTN?type=view.

BVerfG. 2009. Judgment of the Second Senate of 03 March 2009 - 2 BvC 3/07. paras.(1–116). https://www.bundesverfassungsgericht.de/SharedDocs/Entscheidungen/EN/2009/03/cs20090303_2bvc000307en.html.

Camenisch, Jan, Rafik Chaabouni und Abhi Shelat. 2008. „Efficient Protocols for Set Membership and Range Proofs“. In *Advances in Cryptology - ASIACRYPT 2008,* herausgegeben von Josef Pieprzyk, 234–252. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-89255-7.

Casino, Fran, Thomas K. Dasaklis und Constantinos Patsakis. 2019. „A systematic literature review of blockchain-based applications: Current status, classification and open issues“. *Telematics and Informatics* 36:55–81. ISSN: 0736-5853. https://doi.org/https://doi.org/10.1016/j.tele.2018.11.006. http://www.sciencedirect.com/science/article/pii/S0736585318306324.

Chaum, David L. 1981. „Untraceable electronic mail, return addresses, and digital pseudonyms“. *Communications of the ACM* 24 (2): 84–90.

Clarke, Dylan, und Tarvi Martens. 2016. „E-voting in Estonia“. *Real-World Electronic Voting: Design, Analysis and Deployment.*

Cohen, Josh D., und Michael J. Fischer. 1985. *A robust and verifiable cryptographically secure election scheme.* Yale University. Department of Computer Science.

De Tullio, Maria F., Diego Romano und Erica Vaccaro. 2020. „Electronic Voting With Blockchain: An Experience From Naples, Italy“. Besucht am 20. April 2020. https://cointelegraph.com/news/electronic-voting-with-blockchain-an-experience-from-naples-italy.

Delaune, Stephanie, Steve Kremer und Mark Ryan. 2006a. „Coercion-resistance and receipt-freeness in electronic voting“ (Juli): 12 pp.–42. ISSN: 2377-5459.

Delaune, Stephanie, Steve Kremer und Mark Ryan. 2006b. „Verifying Properties of Electronic Voting Protocols“. In *in ”Proceedings of the IAVoSS Workshop On Trustworthy Elections (WOTE’06),* 45–52.

Dill, David L. 2004. „Electronic voting: An overview of the problem“. *Computer Science, Stanford University.*

Eijk, Cees van der, und Marcel van Egmond. 2007. „Political effects of low turnout in national and European elections“. *Electoral Studies* 26 (3): 561–573. ISSN: 0261-3794. https://doi.org/https://doi.org/10.1016/j.electstud.2006.10.003. http://www.sciencedirect.com/science/article/pii/S0261379406001016.

Finck, Michèle. 2019. „Blockchain and the General Data Protection Regulation“. *Panel for the Future of Science and Technology* (Juli): 1–120. https://doi.org/10.2861/535.

Foa, Roberto S., und Yascha Mounk. 2016. „The democratic disconnect“. *Journal of Democracy* 27, Nr. 3 (Juli): 5–17. https://doi.org/10.1353/jod.2016.0049.

Franck, Thomas M. 1992. „The emerging right to democratic governance“. *American Journal of International Law* 86 (1): 46–91.

Franklin, Mark N. 2004. *Voter turnout and the dynamics of electoral competition in established democracies since 1945.* Cambridge University Press.

GDPR. 2018. „2018 reform of EU data protection rules“ (25. Mai 2018). Besucht am 18. November 2019. https://gdpr-info.eu/art-17-gdpr/.

Gerber, Alan S., Gregory A. Huber, David Doherty und Conor M. Dowling. 2013. „Is there a secret ballot? Ballot secrecy perceptions and their implications for voting behaviour“. *British Journal of Political Science* 43 (1): 77–102.

Geys, Benny. 2006. „Explaining voter turnout: A review of aggregate-level research“. *Electoral studies* 25 (4): 637–663.

Goldwasser, Shafi, Silvio Micali und Charles Rackoff. 1989. „The Knowledge Complexity of Interactive Proof Systems“. *SIAM Journal on Computing* 18 (1): 186–208. https://doi.org/10.1137/0218012. eprint: https://doi.org/10.1137/0218012. https://doi.org/10.1137/0218012.

Groth, Jens. 2016. „On the Size of Pairing-based Non-interactive Arguments“. https://eprint.iacr.org/2016/260.

Groth, Jens, und Mary Maller. 2017. „Snarky Signatures:
Minimal Signatures of Knowledge from Simulation-Extractable SNARKs“. https://eprint.iacr.org/2017/540.

Hastings, Nelson, Nelson Hastings, René Peralta, Stefan Popoveniuc und Andrew Regenscheid. 2011. „Security considerations for remote electronic UOCAVA voting“, https://doi.org/10.6028/NIST.IR.7770.

Jonker, Hugo, und Wolter Pieters. 2010. „Anonymity in Voting Revisited“. In *Towards Trustworthy Elections: New Directions in Electronic Voting,* herausgegeben von David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski und Ben Adida, 216–230. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-12980-3. https://doi.org/10.1007/978-3-642-12980-3_13. https://doi.org/10.1007/978-3-642-12980-3_13.

Juels, Ari, Dario Catalano und Markus Jakobsson. 2010. „Coercion-resistant electronic elections“. In *Towards Trustworthy Elections,* 37–63. Springer.

Lai, Wei-Jr, und Ja-Ling Wu. 2018. „An efficient and effective Decentralized Anonymous Voting System“. *CoRR* abs/1804.06674. arXiv: 1804.06674. http://arxiv.org/abs/1804.06674.

Lamport, Leslie, Robert Shostak und Marshall Pease. 2019. „The Byzantine generals problem“. In *Concurrency: the Works of Leslie Lamport,* 203–226.

Lauer, Thomas W. 2004. „The risk of e-voting“. *Electronic Journal of E-government* 2 (3): 177–186.

Lee, Kwangwoo, Yunho Lee, Dongho Won und Seungjoo Kim. 2010. „Protection Profile for Secure E-Voting Systems“. In *Information Security, Practice and Experience,* herausgegeben von Jin Kwak, Robert H. Deng, Yoojae Won und Guilin Wang, 386–397. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-12827-1.

Luu, Loi, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena und Aquinas Hobor. 2016. „Making Smart Contracts Smarter“. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,* 254–269. CCS ’16. Vienna, Austria: Association for Computing Machinery. ISBN: 9781450341394. https://doi.org/10.1145/2976749.2978309. https://doi.org/10.1145/2976749.2978309.

McCorry, Patrick, Siamak Shahandashti und Feng Hao. 2017. „A Smart Contract for Boardroom Voting with Maximum Voter Privacy“. Januar.

Mingxiao, Du, Ma Xiaofeng, Zhang Zhe, Wang Xiangwei und Chen Qijun. 2017. „A review on consensus algorithm of blockchain“. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* 2567–2572.

Mohen, Joe, und Julia Glidden. 2001. „The Case for Internet Voting“. *Commun. ACM* (New York, NY, USA) 44, Nr. 1 (Januar): 72–ff. ISSN: 0001-0782. https://doi.org/10.1145/357489.357511. https://doi.org/10.1145/357489.357511.

Nakamoto, Satoshi. 2008. „Bitcoin whitepaper“. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019).*

Nakamoto, Satoshi. 2010. „Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG“. Besucht am 25. März 2020. https://bitcointalk.org/index.php?%20topic=195.msg1611#msg1611.

Ne Oo, Htet, und Aye Moe Aung. 2013. „Implementation and analysis of secure electronic voting system“. *Int. J. Sci. Technol. Res* 2 (3): 158–161.

Norden, Lawrence D., und Christopher Famighetti. 2015. *America's Voting Machines at Risk.* Brennan Center for Justice at New York University School of Law.

Petkus, Maksym. 2019. „Why and How zk-SNARK Works“. *CoRR* abs/1906.07221. arXiv: 1906.07221. http://arxiv.org/abs/1906.07221.

Rivest, Ronald L., Adi Shamir und Yael Tauman. 2006. „How to leak a secret: Theory and applications of ring signatures“. In *Theoretical Computer Science,* 164–186. Springer.

Rosenblatt, Julio K., und James A. Hendler. 1999. „Architectures for Mobile Robot Control“. Herausgegeben von Marvin V. Zelkowitz, Advances in Computers, 48:315–353. ISSN: 0065-2458. https://doi.org/https://doi.org/10.1016/S0065-2458(08)60023-6. http://www.sciencedirect.com/science/article/pii/S0065245808600236.

Samarati, Pierangela, und Latanya Sweeney. 1998. „Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression“.

Shaxson, Nicholas, John Christensen und Nick Mathiason. 2012. „Inequality: You don't know the half of it“. *Tax Justice Network* 19, Nr. 2 (Juli): 2.

Steiner, Nils D. 2010. „Economic globalization and voter turnout in established democracies?“ *Electoral Studies* 29 (3): 444–459.

Stockemer, Daniel. 2017. „Electoral participation: how to measure voter turnout?“ *Social Indicators Research* 133 (3): 943–962.

Swan, Melanie. 2015. *Blockchain: Blueprint for a new economy.* Ö'Reilly Media, Inc."

Szabo, Nick. 1997. „Formalizing and Securing Relationships on Public Networks“. *First Monday* 2, Nr. 9 (September). https://doi.org/10.5210/fm.v2i9.548. https://journals.uic.edu/ojs/index.php/fm/article/view/548.

Takabatake, Yu, Daisuke Kotani und Yasuo Okabe. 2016. „An anonymous distributed electronic voting system using Zerocoin“.

Tarasov, Pavel, und Hitesh Tewari. 2017. „THE FUTURE OF E-VOTING“. 12 (Dezember): 148–165.

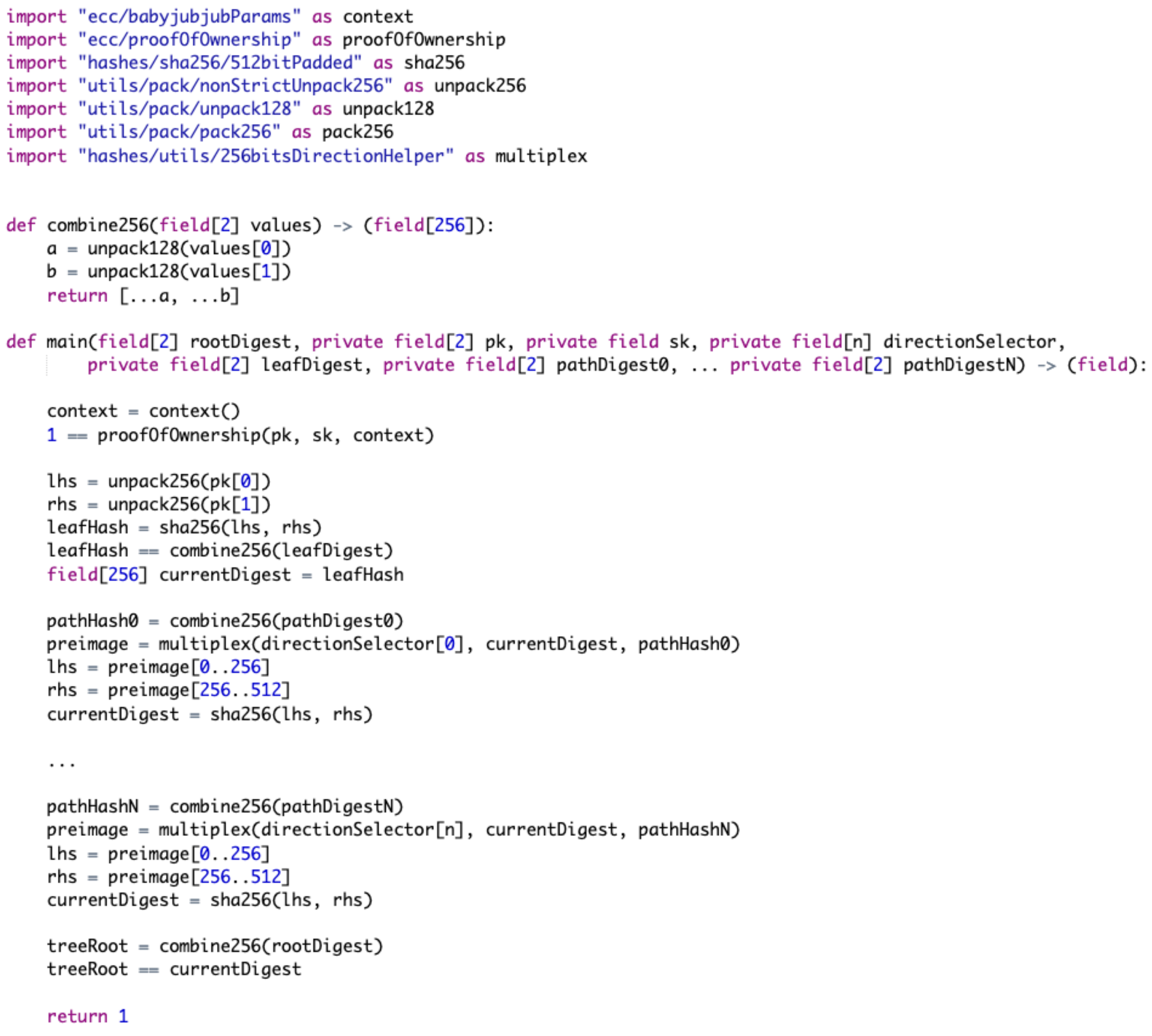
Voshmgir, Shermin. 2019. *Token Economy: How Blockchains and Smart Contracts Revolutionize the Economy.* Shermin Voshmgir. ISBN: 9783982103822. https://books.google.de/books?id=-Wp3xwEACAAJ.

Wang, Baocheng, Jiawei Sun, Yunhua He, Dandan Pang und Ningxiao Lu. 2018. „Large-scale Election Based On Blockchain“. 2017 International Conference On Identification, Information And Knowledge In The Internet Of Things, *Procedia Computer Science* 129:234–237. ISSN: 1877-0509. https://doi.org/https://doi.org/10.1016/j.procs.2018.03.063. http://www.sciencedirect.com/science/article/pii/S1877050918302874.

Wu, Yifan. 2017. „An e-voting system based on blockchain and ring signature“. *Master. University of Birmingham.*

Xu, Jennifer J. 2016. „Are blockchains immune to all malicious attacks?“ *Financial Innovation* 2 (1): 25.

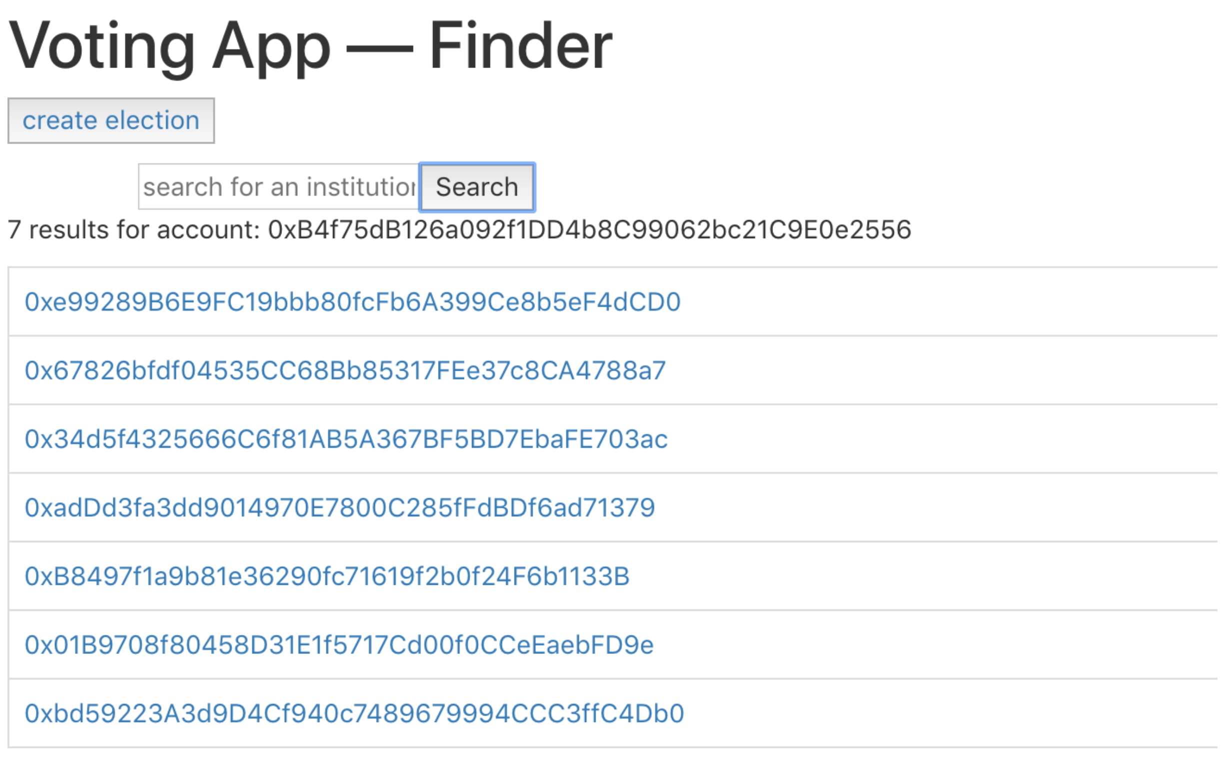
A. Appendix



```
import "ecc/babyjubjubParams" as context
import "ecc/proofOfOwnership" as proofOfOwnership
import "hashes/sha256/512bitPadded" as sha256
import "utils/pack/nonStrictUnpack256" as unpack256
import "utils/pack/unpack128" as unpack128
import "utils/pack/pack256" as pack256
import "hashes/utils/256bitsDirectionHelper" as multiplex


def combine256(field[2] values) -> (field[256]):
    a = unpack128(values[0])
    b = unpack128(values[1])
    return [...a, ...b]

def main(field[2] rootDigest, private field[2] pk, private field sk, private field[n] directionSelector,
        private field[2] leafDigest, private field[2] pathDigest0, ... private field[2] pathDigestN) -> (field):

    context = context()
    1 == proofOfOwnership(pk, sk, context)

    lhs = unpack256(pk[0])
    rhs = unpack256(pk[1])
    leafHash = sha256(lhs, rhs)
    leafHash == combine256(leafDigest)
    field[256] currentDigest = leafHash

    pathHash0 = combine256(pathDigest0)
    preimage = multiplex(directionSelector[0], currentDigest, pathHash0)
    lhs = preimage[0..256]
    rhs = preimage[256..512]
    currentDigest = sha256(lhs, rhs)

    ...

    pathHashN = combine256(pathDigestN)
    preimage = multiplex(directionSelector[n], currentDigest, pathHashN)
    lhs = preimage[0..256]
    rhs = preimage[256..512]
    currentDigest = sha256(lhs, rhs)

    treeRoot = combine256(rootDigest)
    treeRoot == currentDigest

    return 1
```

Abbildung A.1.: PoE zokrates file

B. Appendix



Voting App — Finder

create election

search for an institution | Search

7 results for account: 0xB4f75dB126a092f1DD4b8C99062bc21C9E0e2556

0xe99289B6E9FC19bbb80fcFb6A399Ce8b5eF4dCD0

0x67826bfdf04535CC68Bb85317FEe37c8CA4788a7

0x34d5f4325666C6f81AB5A367BF5BD7EbaFE703ac

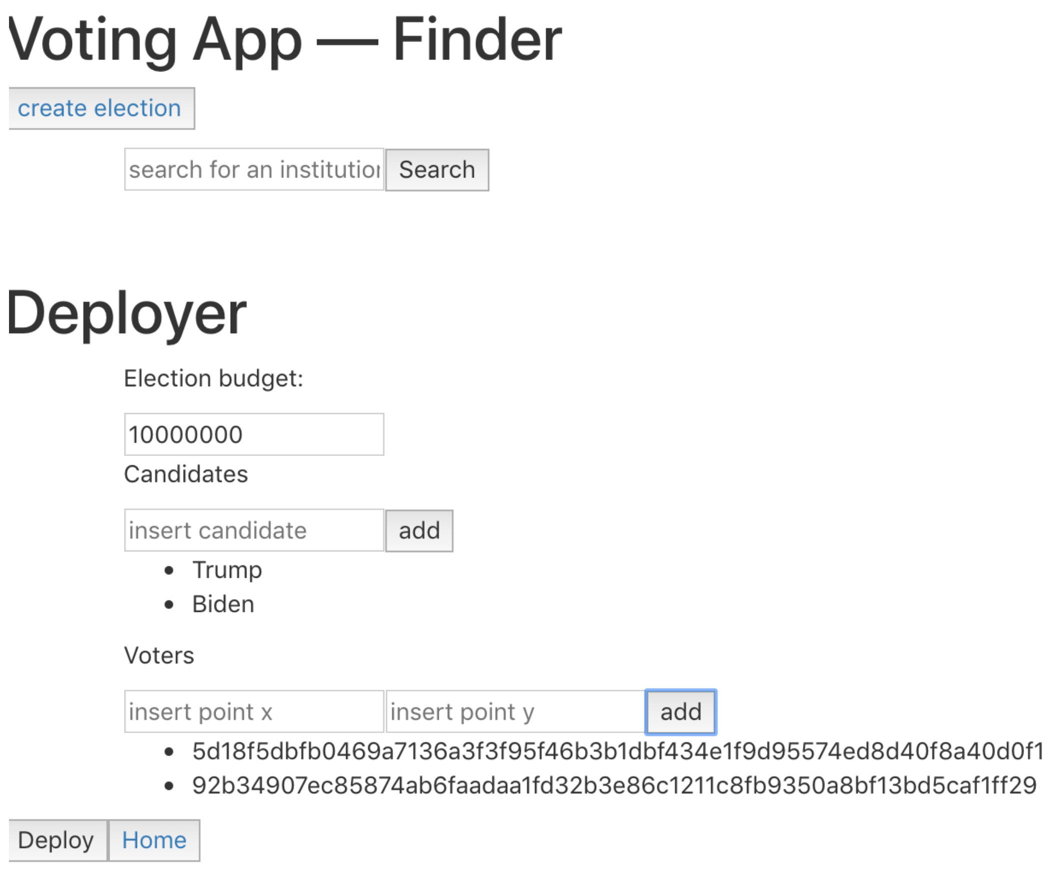
0xadDd3fa3dd9014970E7800C285fFdBDf6ad71379

0xB8497f1a9b81e36290fc71619f2b0f24F6b1133B

0x01B9708f80458D31E1f5717Cd00f0CCeEaebFD9e

0xbd59223A3d9D4Cf940c7489679994CCC3ffC4Db0

Abbildung B.1.: Search by institution on *Snarky*



Voting App — Finder

Deployer

Election budget: 10000000

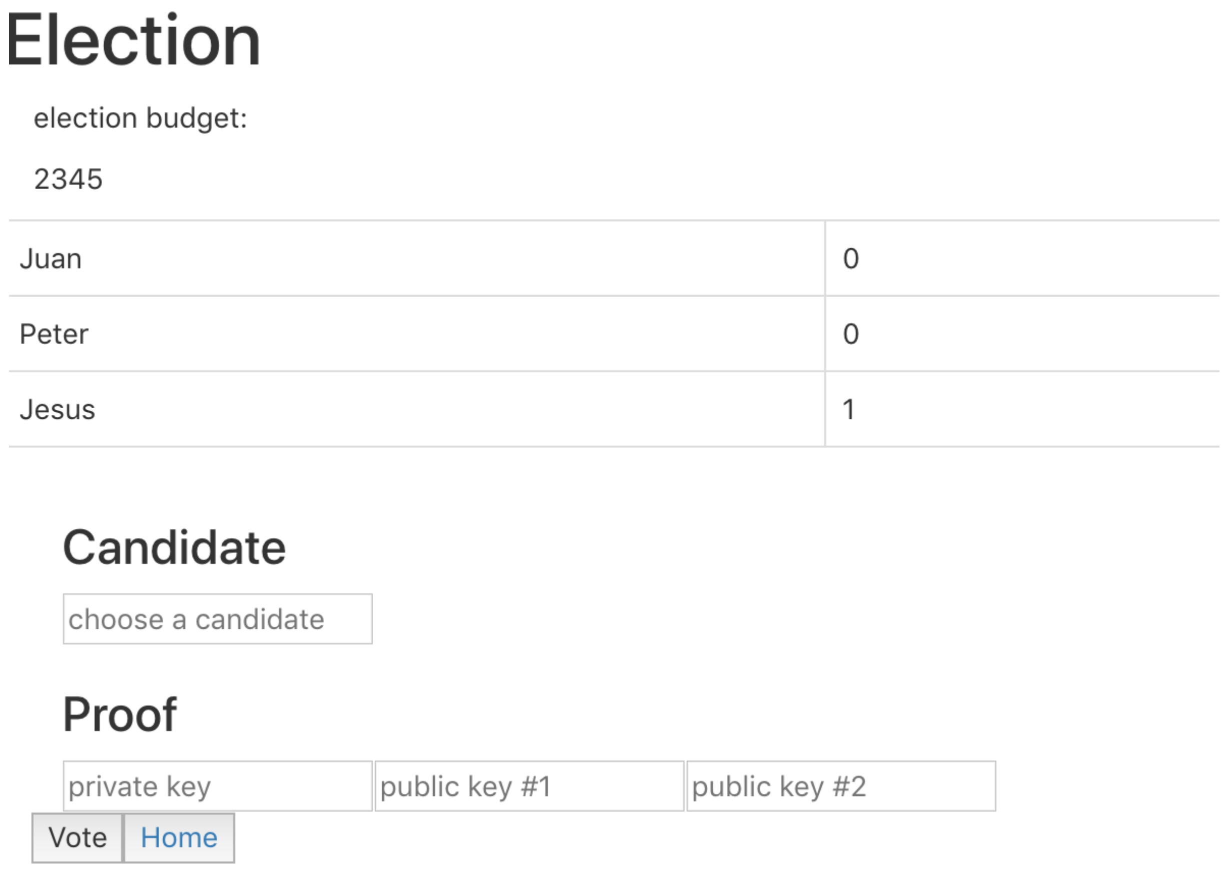
Candidates

- Trump
- Biden

Voters

- 5d18f5dbfb0469a7136a3f3f95f46b3b1dbf434e1f9d95574ed8d40f8a40d0f1
- 92b34907ec85874ab6faadaa1fd32b3e86c1211c8fb9350a8bf13bd5caf1ff29

Abbildung B.2.: Election deployment on *Snarky*



Election

election budget:

2345

Juan	0
Peter	0
Jesus	1

Candidate

choose a candidate

Proof

private key | public key #1 | public key #2

Abbildung B.3.: Voting on *Snarky*