University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Design and Prototypical Implementation of a Verifiable Remote Postal Voting System

Elexa Heggli
Zurich, Switzerland
Student ID: 18-926-741

Supervisors: Christian Killer, Jan von der Assen, Prof. Dr. Burkhard Stiller
Date of Submission: July 18, 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Abstimmungen und Wahlen haben einen hohen Stellenwert in unserer Gesellschaft; ob es sich nun um eine Abstimmung im kleinen Rahmen oder um eine weltweite digitale Wahl handelt, sie bietet die Möglichkeit, die Meinung der Beteilligten darzustellen. Dies ist unter anderem für die demokratische Politik von entscheidender Bedeutung. Die Gewährleistung des Schutzes der Privatsphäre der Wähler*innen und der Prüfbarkeit des Wahlergebnisses ist der Schlüssel zur Aufrechterhaltung von Demokratien auf der ganzen Welt. Daher untersuchen viele Expert*innen Mittel, um diese beiden und andere wichtige Aspekte in den derzeitigen politischen Wahlsystemen zu gewährleisten. In der Schweiz geben 90% der Wählerinnen und Wähler ihre Stimme über das System der Briefwahl ab. Dieses System bietet einen hohen Schutz der Privatsphäre, lässt aber die Überprüfbarkeit des Wahlergebnisses vermissen.

Eine Analyse des brieflichen Abstimmens in der Schweiz deckt dessen Gefahren auf und gibt Einblick in Verbesserungsmöglichkeiten. In dieser Arbeit wird ein Prototyp vorgeschlagen, der die Verifizierbarkeit des Wahlergebnisses erhöht und gleichzeitig die Privatsphäre der Wähler*innen und die Benutzerfreundlichkeit des Systems wahrt. Inspiriert von verwandten Arbeiten, verfolgt der Prototyp die Wahl Umschläge durch UUIDs mittels QR-Codes, die von Wahlbehörden und Wählerinnen und Wähler gescannt werden. Ein Smart Contract auf der Ethereum-Blockchain verfolgt die UUIDs während der Wahl, um die Grösse der nötigen Vertrauen in die Wahlbehörden zu minimieren. Der Prototyp fügt dem aktuellen briefliche Abstimmen die Überprüfbarkeit hinzu und mildert einige Bedrohungen, insbesondere die Fälschung von Wahlberechtigungsnachweisen oder Wahlumschlägen.

Voting holds an essential value in our society; whether it is a small-scale in-person vote or a worldwide digital election, it provides the means to portray the voter's opinions. Which, amongst other areas, is vital in democratic politics. Ensuring the privacy of the voters and the accuracy of the voting outcome is key to maintaining democracies all around the world. Therefore, many experts investigate means to provide these two and other vital aspects to current political voting systems. In Switzerland, 90% of voters cast their ballot through the Remote Postal Voting Scheme. A scheme that provides high privacy but lacks verifiability of the voting outcome.

An analysis of the SPVS uncovers its threats and gives insight into possibilities for improvement. This thesis introduces a prototype to increase voting verifiability while maintaining the voter's privacy and the system's ease of use. Inspired by related work, the prototype tracks the envelopes through UUIDs in QR codes to be scanned by voting officials and voters. A smart contract on the Ethereum blockchain keeps track of the ids

throughout each voting instance to minimize the amount of trust placed in the voting authorities. The prototype adds Eligibility Verifiability to the current SPVS and mitigates some threats, especially forgery of voting credentials or the voting envelopes.

Contents

Chapter 1

Introduction

Voting is at the foundation of democracies worldwide, and it is vital to ensure the accuracy and privacy of these voting processes to maintain the public's trust. Fair voting is only present if every citizen feels safe to vote according to their wishes and if the published results accurately represent how the public voted. In a perfect world, voting only occurs in person at a polling station; it is the most secure way to vote. However, house-bound citizens and citizens living abroad must also be able to vote; this is where remote voting originates. Regarding political voting in general, the privacy of the voters and the accuracy of the outcome aren't the only aspects that need to be considered. Amongst other factors, the voting turnout can't be overlooked. The increasing trend towards remote voting is a definite step towards improving voter turnout by providing the comfort of voting anywhere at any time. In the US 2020 elections, for example, 46% of the voters voted by absentee or postal voting (the other option being in person) [6]. The global covid-19 pandemic has only quickened this trend.

Remote voting can be categorized into two forms: postal vs. internet voting, with some hybrid solutions. The remote voting trend began with postal voting, e.g., in Switzerland, it was launched in the late 1970s and then anchored in the law in 1994 [31]. Before the covid-19 outbreak, around a quarter of all countries worldwide had used postal voting for their national elections, and almost all used a form of paper ballots. However, the different countries' views and approaches are very split in internet voting. While it is banned in some countries, many have already done some trials in the past. Currently, four countries allow voting via the internet: Armenia, Canada, Estonia, and Switzerland; however, in some cases, this is restricted to a certain population (e.g., citizens living abroad) [34]. A big concern with internet voting is the trust placed in these complex systems, which are usually more difficult for an average voter to understand. In Norway, for example, has had some trial runs for Internet voting systems. They were stopped due to a lack of trust from the public and mainly from the politicians involved [2].

When introducing a new voting system, it is always a trade-off between different equally important factors. For example, a significant advantage of remote postal voting is its easy cast-as-intended verification, meaning it is given that the vote cast by the voter is the vote she indented to cast (e.g., by marking an X in the corresponding field). This and many other privacy and security aspects are important when evaluating voting systems.

These voting processes often represent a big power play in the political world, and the threat of manipulation is omnipresent. Most of the beforementioned voting processes have been heavily under attack. In many countries, there have been allegations of voting fraud both in postal and internet voting systems [36, 37, 38]. Therefore, many parties are interested in and are working on improving voting processes worldwide. There has been a big movement toward digitalizing these voting processes to attempt to have more proof of accuracy for the public and incentivize more citizens to vote. Often these improvements come at a cost, though. The environments become more complex whenever technology is added and more vulnerable to new attacks. Even if a perfect cryptographic verification is put in place and a voting system is 100% accurate and private, there is always the possibility that the voter's device used to place a ballot electronically could be targeted. On top of that, with the quickly evolving technological world, cryptographic systems that are considered safe now and might not be so safe in the years to come. Therefore, this paper focuses on improving the current postal voting systems to reap the many benefits of postal voting and attempt to improve its accuracy. This will be done specifically on the Swiss use case.

The Swiss postal voting system (SPVS) is a special case. Not only does Switzerland have direct democracy with very frequent votes (around 4 per year), but its federalization also hinders the generalizability of these voting processes. Killer and Stiller summarized the SPVS into seven steps: Setup, Delivery, Casting, Storage, Tallying, Validation, and Destruction. In those steps, they went on to classify all the possible threats to the voting process into 14 threats shown in 6.1. There have been several cases in Switzerland where the current SPVS has been misused, according to one of these threats identified by Killer and Stiller. For example, in 2021 in Frauenfeld, there were reports of manipulation of stored ballots; in 2019 in Geneva, there were allegations of destroying and forging stored ballots; and in 2017 in Wallis, there have also been reports of the theft and forging of ballots [3, 14]. While the SPVS can claim many of the privacy benefits of a postal voting system, it lacks verifiability (accuracy of the voting outcome), i.e., a lot of trust is placed on the many authorities who run the whole voting process. This paper aims to maintain all the security standards of the current SPVS based on Stiller and Killer and attempt to improve its verifiability. [23]

1.1 Description of Work

This thesis focuses on designing, implementing, and evaluating a smart contract and interfaces to gain eligibility verifiability in the current SPVS. It is a proof of concept showing how it is possible to improve verifiability while maintaining the current privacy standards. The thesis is structured into 4 parts: research, system design, implementation, and evaluation. The research provides an understanding of the voting field with an in-depth look at the SVSP. An analysis of these different voting systems worldwide and their evaluations concerning security and privacy follow. In the system design, various means of improving verifiability are tested, and a few security stages are defined. A prototype is designed to focus on eligibility verifiability (EV) and mitigating the SPVS' threats. In the implementation, the smart contract and its interfaces for the stakeholders are implemented

in a test environment. And lastly, the newly designed voting process is evaluated and compared with previous designs based on the foundations of voting processes.

1.2 Thesis Outline

Chapter 2 provides a foundation for relevant aspects of voting systems with an in-depth description of the current SPVS and the basics of the technologies used in this paper.

Chapter 3 summarizes related work in the voting environment and compares some of the previously proposed approaches to improve voting systems through different means.

Chapter 4 presents the requirements for designing the prototype and highlights the steps in the design process.

Chapter 5 demonstrates how the prototype was implemented. It describes its setup, provides insights on the smart contract, and then explains the different layers of the User Interface.

Chapter 6 summarizes the results of the implemented prototype along with the criteria from the design.

Chapter 7 discusses the beforementioned results and evaluates the prototype's usability in the Swiss postal voting system.

Chapter 8 concludes the thesis and provides a summary of the work and ideas for future work.

Chapter 2

Background

2.1 Foundations of Voting Systems

To ensure the privacy and accuracy of the voting systems, the following four foundations of voting systems have been identified: privacy, verifiability, software independence, and accountability. Over the past few years, different experts have incremented and built on top of these foundations. The challenge when designing a voting system is the trade-off between these foundations. As Chevallier-Mames et al. state, complete privacy can't co-occur with verifiability; hence it is always the aim to achieve standards as high as possible in all four areas. [8]

2.1.1 Privacy

In the 19th century, most countries introduced laws making privacy in public voting obligatory to stop bribery and coercion. [11] One of the first definitions is 'Ballot Privacy', when an election doesn't leak any information about the voters except for the outcome of the vote. I.e., there is no way to detect how any person voted except if the voting result is unanimous for either side. A further step to combat any manipulation threats is to verify that the voting process is 'Receipt-Free', i.e., when a voter has no chance of proving how they voted even if they go against the protocol in an attempt to create a proof. [5] This privacy standard is a strong counter incentive for blackmailing a voter into voting in a specific way. If voters can't produce evidence of how they voted, that threat is averted. Receipt-freeness assumes that the attacker isn't in the room with you though. 'Coercion-Resistance' covers receipt-freeness and adds protection against forced abstention, forced randomized voting (an example is introduced in the related work section), and forcing voters to give up their voting credentials to cast a vote for them. With these increments, it gets increasingly more challenging to maintain its requirements and protect the voter's integrity. There are many more definitions of privacy depending on the specific voting systems; however, the three presented are the most popular. [22]

2.1.2 Verification

Another equally important aspect of voting systems is ensuring that the election outcome correctly portrays the public's votes, confirming that every voter's choice is accurately tallied and represented in the published results. In voting systems, this is called 'Verifiability', and there are usually two different increments used in the related work. Either it is measured by 'Individual Verifiability' (IV) and 'Universal Verifiability' (UV) or in three steps as the 'End-to-end Verifiability'. Individual Verifiabibility implies that every voter can check that their vote is tallied correctly in the final result. Universal verifiability defines that anyone can verify that all the ballots were correctly tallied, and the voting outcome corresponds to this tallying.

The second concept, 'End-to-End Verifiabilitiy', consists of the following three definitions: 'Cast-as-Intended' (CaI) covers the first part of the voting process. It enables voters to independently check that their choice is correctly cast in the present voting system. E.g., in postal voting, this is automatically given since every voter can verify the correctness of their own markings on the paper ballot. However, in Internet Voting, this definition carries more significance since there might be a false layer on top of the actual voting site, which leads to a wrong choice being recorded. The next step is called 'Collected-as-cast' or 'Recorded-as-cast' (RaC), when voters can check that their vote is received the way they intended to cast it. The third and last step is defined as 'Counted-as-collected' or 'Tallied-as-recorded' (TaR), which verifies that the final tally in the voting results corresponds to how the tallying office received it [5].

These definitions of verifiability don't cover a crucial part of voting systems: the eligibility of the voters. Kremer et al. coined the term 'Eligibility Verifiability' (EV), which ensures that the public can verify that every voter is an eligible voter (whoever is classified as eligible in the corresponding elections) and that each voter solely cast a single vote [24, 25]. To achieve this verifiability, some systems introduce an audit trail (this can be done electronically, on paper, or as a combination of the two) to keep track of the voting process from beginning to end. Different standards of verifiability apply depending on who has access to this audit trail, whether it is only the authorities, each individual voter of their ballot, or the public, and how secure it is. [11]

2.1.3 Software Independence and Accountability

The last two foundations are often neglected, even though they are equally important to enable a 'fair' voting system. Software independence ensures that if there is an undetected error or change in the software used, it can't cause an undiscovered error in the election outcome. [5] Accountability incentivizes the voting authorities to uphold all the correct voting practices, as they will be held accountable if they intentionally or unintentionally misuse their power to tamper with the voting process. In most voting systems, a lot of trust is placed in the people in these authority positions; the public can only guarantee this trust if they are held accountable for their actions. [22] This includes having mechanisms in place to handle situations where fraud is suspected and to be able to trace the missuses to the attackers.

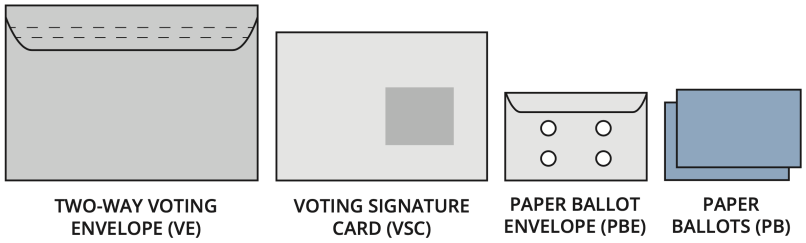

Figure 2.1: Representation of neccessary artifacts for SPVS depicted by Killer and Stiller [23]

2.2 Swiss Postal Voting System

In the direct democracy of Switzerland, voting is part of everyday life for eligible voters. National votes are held two to four times a year, and cantonal and municipal votes can occur even more often. According to the Swiss Federal Chancellery, each voting iteration costs about 7.5 million CHF [39]. 90% of voters use the remote postal voting (RPV) system provided by the government, cantons, and municipalities. This system requires a lot of trust in the authorities and its external suppliers (ES). A trust that most voters give in Switzerland. Another factor that heavily influences the voting processes in Switzerland is its federalism. While the national government has strict laws about voting, each canton and municipality has some influence in creating their voting procedures. Therefore a universal SPVS doesn't exist and also isn't documented anywhere. Fortunately, Stiller and Killer have summarized the SPVS in a step-for-step Postal Voting Process Flow (PVPF) based on some significant assumptions. Additionally, they analyzed the whole process for its risks and threats. [23] Stiller and Killer's paper will be used as a reference for the SPVS throughout this thesis.

As a quick overview, this paragraph will summarize the current Swiss PVPF. As beforementioned, the PVPF is divided into seven steps:

- **A. Setup** This is a rather large phase, it includes retrieving the Electoral Register (ER) and producing the different parts of the voting envelopes as shown in 2.1: the Two-Way Voting Envelope (VE), Voting Signature Card (VSC), the Paper Ballot Envelope (PBE) and the Paper Ballots (PB). This production is often outsourced to an external supplier, who then prepares the envelopes for dispatch and delivers them to the postal service.
- **B. Delivery** The Swiss Post (SP) is responsible for the secure delivery of the VEs.
- **C. Casting** Once the citizen has received their envelope, they have three different means to cast their vote: I. by sending the envelope by post, II. by throwing the envelope into their municipality's letterbox, or III. by casting a vote in person at the urn of their municipality.
- **D. Storage** In Switzerland, all votes are received until the official Voting Day. The VEs from the letterbox and post are brought to a safe storage location; this impor-

tant step is often not prioritized due to lack of funding, and many fraud allegations have been made in this phase.

- **E. Tallying** The municipality and the local election office (EO) handle the process of tallying. The SPVS works with a two-way envelope system; the outer envelope contains the voter's identification (address, name, and signature) and the anonymous paper ballot envelope, which is opened separately to avoid any ties between a ballot and the voter. Most municipalities hire helpers to assist with the manual counting, although a few also use high-precision scales or e-Counting systems to tally the votes. It is important to note that the EO assumes that every envelope received at the tallying office hasn't been tampered with and has been sent by an eligible voter

- **F. Validation** Once tallying is finished, the cantonal government gathers the results, transfers them to the Federal Chancellery (FC), and publishes them to the Cantonal Gazette (often software provided by ESs is used to transfer the results). When no valid appeals concerning the results are in process at the Swiss Federal Court, the official results are published in the Federal Gazette.

- **G. Destruction** With that, the process is sealed, and the ballots that have been stored until now are destroyed.

In each voting process, there are trade-offs between privacy and verifiability. Either the design attempts to make the votes as private as possible, or there is a heavier focus on providing proof of correctness. Deciding which direction to take also depends on the circumstances involved in the voting process. When analyzing the SPVS, it becomes apparent that its focus lies more on privacy than verifiability, as with most paper-based postal voting systems. The SPVS provides ballot privacy based on a few assumptions, including the trust placed in the swiss postal service and the authorities involved in the voting process. The government doesn't publish any information concerning the elections except for the outcome and some voting statistics. Whether receipt-freeness is given in a postal voting system is a bit disputed, however, since the only way to produce proof is to take a video, it is often counted as receipt-free [20]. The SPVS isn't coercion-resistant; an approach that other countries make to improve coercion-resistance is to allow voters to vote multiple times (therefore, if an attacker coerces a voter into voting a certain way, the voter can easily cast another vote later on) and then only counting the last ballot casted on time.

On the other hand, only one small part of verifiability is given in the SPVS, namely 'Cast-as-Intended Verification', which is an automatic byproduct of voting in paper-based systems. The SPVS doesn't provide further proof of whether the ballot cast is received or tallied correctly. There isn't a lot of research into software independence and accountability in the SPVS; however, there is a lot of problematic software involved which (based on the summarized SPVS flow [23]) has many security threats, e.g., the software used to prove the citizen registry. Therefore it can be assumed that software independence isn't given.

2.3 Blockchain

Blockchain technology was Satoshi Nakamoto's solution to the problem of establishing trust in a distributed system. Unlike a centralized system, it does not rely on a single party who owns and controls all processes. The anonymous author introduced a distributed storage of time-stamped artifacts signed with digital signatures, which cannot be manipulated without being detected. The blockchain is a possible solution to many current problems, such as integrity, authentication, validation, and more. It is a public database; several parties update and distribute the artifacts, such that multiple sources demonstrate that no data has succumbed to manipulation. A 'block' resembles the data stored, e.g., the transaction data of sending someone Ether is added as a block. The 'chain' represents the linked blocks, where each block is connected to its parent. Tampering can't remain undetected since a single block cannot be manipulated without changing all following blocks [43].

2.3.1 Ethereum Smart contracts

The Ethereum blockchain uses a mechanism called proof-of-work. Every entity that wants to add a block must solve a complex puzzle (in a process called 'mining') using much computational power. In order to execute a transaction, a block is added to the blockchain once it has been mined. The new block is verified, and the blockchain is updated to a new state for the entire network. In Ethereum, the state everyone agrees on is the Ethereum Virtual Machine (EVM); everyone involved records a copy of its state. Ethereum's native cryptocurrency is Ether (ETH), which regulates the computations market. Smart Contracts are programs uploaded to and run by the network to make computations on the EVM. These programs run with specific parameters only when certain conditions are met. For example, a Smart Contract could create a digital asset whenever ETH is sent to a specific user. Anyone can develop Smart contracts and publish them on the network. The data is stored in the blockchain, and the published must pay the computation fees needed. Anyone can then call these smart contracts (within the right conditions) and pay a fee to have them executed [43].

2.3.2 Technologies

In this paragraph, the technologies used for this paper are shortly introduced. A popular development framework for the Ethereum blockchain is called Truffle. It provides fixed scripts to deploy smart contracts and for testing them. The SCs in the truffle framework are coded in the Ethereum programming language called 'Solidity'. A possible option to deploy Smart Contracts is Infura, a Blockchain Development Suite [19]. The web3 javascript library is installed to have a front-end web app interact with the blockchain. A new instance of web3 is initialized; using the eth library in web3, one can then call a contracts function to create a local copy of the solidity smart contract to use in the front-end. To achieve this, the contract address of the deployed smart contract is needed and an application binary interface (ABI). The ABI is a JSON representation of all the

inputs, functions, and methods of the deployed Smart Contract, and it acts like a passport between the Ethereum Blockchain and the front-end javascript. It lets the front-end know which interactions are available on the smart contract before it tries to call them [43].

2.3.3 Ethereum Gas

Running a transaction on the Ethereum Blockchain is not free; the blockchain uses 'gas' to regulate the cost of transactions. Ethereum gas is to smart contracts, what fuel is for a car. It powers the code to run computations on the Ethereum network. In order to calculate how expensive a transaction on the blockchain is, different information needs to be accessible. Firstly, each action on the blockchain has a fixed gas price; for example, a financial transaction costs 21'000 gas. This figure is then multiplied by the current cost of gas. Gas is paid in ether, but the unit used is gwei, a denomination of ether. Gwei stands for Giga Wei which represents 0.000000001 ETH (10^{-9} ETH). Often default gas prices are used; however, the more one pays for the gas, the quicker the blockchain runs the transaction. Now, multiplying the fixed gas price with the cost of gas in gwei, one receives the amount of ether needed to power this transaction. The last step is to convert the amount of ether to the current price of ether in the preferred currency (e.g., USD or CHF) [42].

When deploying a smart contract, it is possible to set a gas limit to have an upper bound for the cost one pays for a particular transaction. Since gas prices also heavily depend on the time of day, the day of the week, and otherworldly influences, it is often complicated to estimate the cost of transactions. Hardhat is a development tool used for any Ethereum blockchain; it serves as an alternative to Truffle. One of Hardhat's advantages is its gas reporter plugin, which estimates the gas price of the functions in the deployed Smart Contract.

2.4 QR code

A widely used means for encoding, sharing, and decoding data are QR codes. I.e., "Quick Response" codes, a two-dimensional matrix code that provides many advantages such as high data storage capacity, omnidirectional readability, error correction, etc. Different versions can be chosen depending on the needed size, 21x21 up to 144x144, and the required error correction, 7% up to 30% of approximate error correction. The data is stored in a regular square array consisting of function patterns and an encoding region, surrounded by a quiet zone border. The encoding area contains the version information, format information, and the data and error correction codewords. Although QR codes have many advantages, we must also consider the disadvantages, especially in the context of postal voting. First, all the users need to have a QR scanning tool, which might limit the audience. Most important, however, are the security issues. When scanning a QR code, one can never be entirely sure where this code is going to lead them. [40]

Chapter 3

Related Work

3.1 Attempts to Improve Voting Systems

Trying to achieve high standards in the four foundations of voting processes is the apparent move to improve voting schemes worldwide. An essential aspect of ensuring these four principles is prioritizing securing the systems against all kinds of attackers. However, many additional attributes contribute to a well-functioning voting system. For one, it should be easy to vote for all eligible citizens and maintain the public's trust to encourage a high voter turnout. All other stakeholders involved, such as the government authorities and third parties managing it, should also have a straightforward process to work with to avoid intentional or unintentional manipulations/fraud. Evaluating the cost and time efficiency of different voting processes is always beneficial to keep voting systems within a reasonable cost and time frame.

Researchers have suggested different approaches to improve voting systems throughout the past few decades. The attempts are split into in-person systems and remote voting systems. Both methods can be paper-based, electronic, or a hybrid of the two. The following related work section will provide a short insight into the different options, with a focus on the second path related to the work done in this paper. This paper will assess each voting process according to the four foundations of voting systems and the additional features mentioned above. As a side note, not all voting systems are designed for governmental use. Many areas in societies include voting, places where there are different and often fewer security and privacy concerns.

Often there are complex cryptographic proofs involved to ensure the privacy and verifiability standards in voting systems. While these proofs might convince experts of the system's safety, these complex mathematical mechanisms often do not help increase public trust. Therefore, some related work introduced alternatives to give voters proof that their votes are private and represented authentically. These mechanisms include verification with short strings, continuous auditing (used in Prêt á Voter), and cast-or-audit systems. In the Markpledge system, voters make their choice electronically, and the machine then prints the ballot containing all encryptions along with a commitment to the encrypted ballot choice. A voter can insert the provided short string to verify that the machine

saved the correct answer [30]. Similar to continuous auditing explained in the Prêt á Voter paragraph below, cast-and-audit systems enable voters to choose whether to cast their prepared vote or to have it decrypted. These alternatives give voters more security, and the more voters who test the voting systems in this way, the more securer the system itself becomes [21].

3.2 Poll site systems

Poll site voting systems often have specific public locations which voters need to visit to record and cast their ballots. In-person voting is where voting has its origins, it started as simply marking your choice on a paper ballot and dropping it into the ballot urn, and nowadays, there are many digital poll site systems in use. It also has many advantages over remote voting; for example, voter authentication and accountability are often more straightforward in the in-person setting [5]. However, privacy and verifiability concerns remain even in in-person voting systems. Therefore many proposals have been made to combine poll site voting with cryptographic schemes to ensure higher security standards, many of them applicable for remote voting [27].

3.2.1 Prêt á Voter

Prêt á Voter is an E2E poll site voting system that uses a system of ballots listing candidates in pseudo-random encrypted orders. The paper lists all candidates on the left side and leaves space to vote on the right side. When a voter marks a candidate, it serves as the encryption of the vote. The encryption of the candidate ordering on the ballot together with the position of the voter's mark is used to tally the correct vote. Ryan et al. make use of continuous auditing to prove to the users who aren't cryptographic experts that their vote is private. A voter can purposely spoil a ballot by handing in a blank ballot and letting it be decrypted to verify that the order and, therefore, their vote is encrypted. They can then choose another ballot and cast their vote. The voter marks the spaces on the right-hand side of the paper and tears it in half. They shred the left-hand side containing the candidate ordering to ensure ballot privacy, place the right half of the form into the voting device, and receive an encrypted receipt after the elections. Voters can check the WBB and confirm their votes with the receipt which ensure E2E verifiability, and public auditors can perform verifiability checks [27, 33, 21]. Various papers have built upon Prêt á Voter to improve some of its issues, and it has also been used as a basis for Internet voting platforms, such as vVote, which was used in trials in Victoria, Australia in 2014 [29]. The Prêt á Voter is receipt-free (since the receipt is encrypted) but not coercion resistant; it is an example of a system susceptible to forced randomization. Since the candidates are listed pseudo-randomly on the ballot, the coercer can force multiple voters to mark the top candidate, which the receipt can prove. All of these votes will balance each other out due to their randomness; this can significantly impact the outcome of the vote if a particular group of voters is forced to vote this way.

3.2.2 Tracker

Gjøsteen et al. take it a step back and regard in-person paper-based voting systems as non-trivial security-critical logistics problems. In this regard, they tested cryptographic solutions from supply chain management and proposed two theoretical ideas to increase the system's resilience against fraud. The first simpler scheme is based on digital signatures, and the second suggests using a Tracker to monitor the supply chains. Both methods focus on creating a secure audit trail for the ballots cast at voting precincts. They rely on all involved election officials verifying the ballot's path and tracking each ballot along every step. For example, the steps include receiving the ballot at the precinct, moving it to storage, taking it out of the storage, and so on.

Digital Signatures schemes comprise three algorithms to verify the authenticity of a message's sender and the authenticity of the message itself (i.e., that it hasn't been altered since the sender signed it). The Key Generation Algorithm creates two random keys: a secret key and a verifying key. The Signature Algorithm takes a message and the secret key from before and outputs the message and a signature. Lastly, the Verification algorithm has the message, the signature, and the verification key as input and outputs whether the signature is valid. Implementing this onto paper-based voting systems prevents voting officials from validating, casting, or counting forged ballots. However, the ballot storage size on each ballot increases with each election official who verifies the ballots' path and therefore also adds their signature - leading to a high computational cost of verification.

Their second idea is based upon the Tracker system, created in the logistics field to trace materials through supply chains and verify their paths' validity. Trackers use cheap RFID tags (like stickers) to place on the paper ballots; these tags aren't linked to an identifiable voter to maintain voter privacy. The key to these tags is given to every authority along the supply chain to verify the ballot's path. Using this key, the voting officials can verify the signature on the ballot with the signature expected according to the correct path. RFID uses a polynomial evaluation mechanism, ensuring that the signatures don't grow like the digital signatures; therefore, this design is more efficient.

Both theories aim to detect and not prevent voting fraud by building on top of paper-based voting designs. However, there was no further investigation on a physical application of the proposed theories on any RPV system. Making it difficult to provide an analysis of the privacy and verifiability standards of the two theories [26].

3.3 Remote electronic voting systems

Remote voting systems are separated into paper-based (such as the current SPVS), fully electronic systems, and a hybrid of the two. Proposals to improve remote electronic voting systems have been appearing for over 30 years, based on varying aspects of cryptography to provide security and privacy while protecting against election manipulation. For example, voting systems have relied on blind signatures, a method used to sign a vote without revealing its content [17]. Mixnets schemes, which shuffle a ballot with several other ballots to ensure privacy, have been used in projects like Scantegrity [7] and Helios [1].

Some rely on homomorphic encryption (combining votes before decryption); a famous example is El Gamal [21, 33, 27]. Since these systems often require complex cryptographic theory, some voting systems include ballot auditing processes to show voters that their votes are private and verified. Processes like verification using short strings, continuous auditing (used in Prêt á Voter), and cast-or-audit enforce more trust in the cryptographic systems [21].

Helios is one of the only voting systems used in practice; [4], it is an open-source, web-based voting scheme designed by Adida for elections with a small risk of coercion [1]. As mentioned above, Helios is based on a mixnet scheme combined with a simplified version of the Benaloh challenge. To cast their vote, voters mark their choice in the provided space. The voter can then choose to audit or cast the encrypted ballot the system has prepared. Once they wish to cast their ballot, the voter must authenticate themselves. The encrypted vote is sent to the public bulletin board for individual verifiability (IV). Using mixnet, the ballots are mixed and decrypted and posted on the bulletin board together with all supporting proofs. The original Helios has been updated and improved for different elections, such as in Belgium University's presidential elections [21]. Based on the informal analysis, Helios is deemed verifiable, i.e., it fulfills individual and universal verifiability. Eligibility Verifiability varies from election to election as some additional properties need to be met, such as a list of the eligible voters. However, the Helios version analyzed here doesn't fully provide ballot privacy, as some proofs are lacking, and attackers could successfully infiltrate the voting system [4].

3.4 Remote hybrid solutions

Both remote electronic voting and remote postal voting have their advantages and disadvantages. In an effort to reap the benefits of both, some proposals have surfaced trying to combine the two. The aspect of having a paper-based audit trail for privacy and an electronic addition for further verifiability seems to be a good match.

3.4.1 McMurtry

McMurtry et al.'s proposal is based on paper voting similar to the SPVS and combines paper-based assurance with cryptographic verification. It significantly improves verifiability in paper-based voting schemes. When casting their vote, voters mark their choice electronically, which is then encrypted by the ElGamal encryption scheme. They cast their encrypted ballot by sending the encrypted vote electronically, printing it out, and sending it by post. The electronic vote is posted on a web bulletin board (WBB) used for universal verification to verify the paper ballot and the tallying process. This scheme uses easy CaI verification, resistance to cryptographic failure, and a low level of trust needed for the paper ballot. However, it manages to defend the ballot from a cheating postal service or electoral authority by having a digital verification of the ballot. It adds honest-but-remembering receipt freeness, RaC verifiability to the paper-based voting scheme and protects its integrity against entirely corrupt authorities.

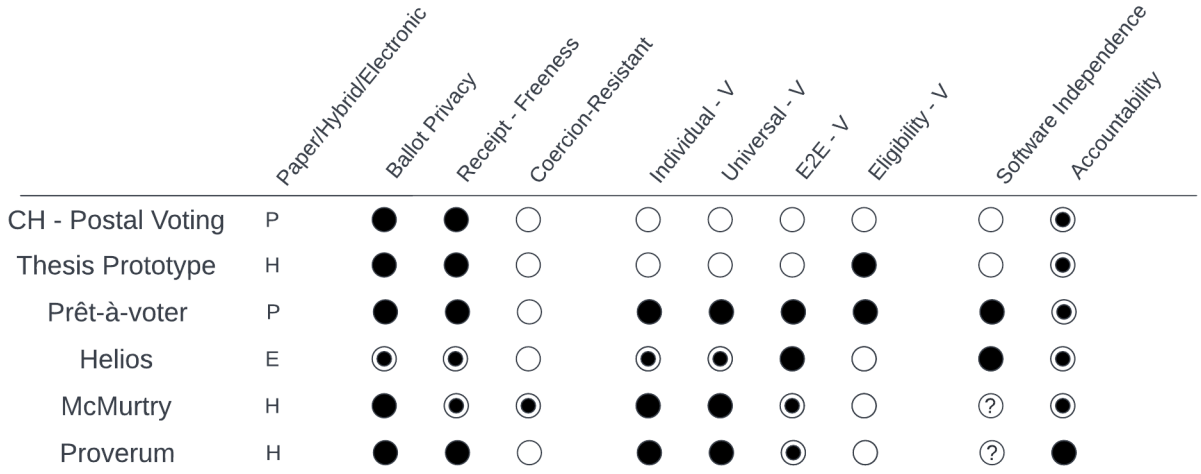

Figure 3.1: Foundations of Voting Systems Analysis

3.4.2 Proverum

Another option to improve verifiability in remote postal voting is introducing a Citizen Management System. Proverum, for example, is an approach that combines immutable audit trails (in a private environment) with multiple permissioned distributed ledgers (DLs) using decentralized identity management. It is proposed for the Swiss postal voting system to establish Hybrid Public Verifiability. The main goal is to allow the public to verify the voting processes while preserving a private verifiable audit trail. Decentralizing these processes establishes more trust and transparency in the voting system.

Chapter 4

Design

4.1 Requirements

As established in the chapters above, this thesis will build on the postal voting system to reap all of its benefits and attempt to lessen the amount of trust needed by giving the voters more proof of verifiability. To narrow down the aim of this thesis, requirements were derived to achieve the overarching goal: to improve the current SPVS. By setting the requirements and analyzing the current threats of the SPVS, a design can be made.

- **Maintain the current privacy and verifiability standards**
  When designing voting processes, trade-offs between those two standards must be made. However, this paper servers as a proof of concept to verify whether the SPVS can uphold additional standards without losing any current privacy or verifiability. With that, an effort should be made to maintain the current software independence and accountability.
- **Improve either privacy or the verifiability standards or both**
- **Prevent as many current threats as possible**
  The aim is to mitigate as many threats analyzed by Killer and Stiller in the current SPVS as possible in the simplest way possible [23]. As well as trying to avoid creating any new threats.
- **Ease of use for all stakeholders involved**
  The design shouldn't introduce new burdens for both the voting authorities and the voters, and it shouldn't lessen the trust placed in the system by the voters by introducing a lot of new, complex technology. It is crucial that voters of all ages and backgrounds still feel comfortable and safe to vote to ensure voter turn-out isn't decreased.

Taking a closer look at the threats analyzed by Killer and Stiller in their paper on the SPVS, it becomes apparent that most threats happen before the tallying phase, i.e., from

A. Setup to the D. Storage phase, eleven out of the fourteen threats are present. An even more alarming analysis shows that all threats except the first one (TE1) are caused by tampering with or forging the Voting Envelope (VE) or citizen data. Therefore introducing some audit trail with a voter id management system seems to be the simplest solution to preventing as many threats as possible. This idea significantly overlaps with the Kremer et al. definition of 'Eligibility Verifiability' (EV). Consequently, this paper aims to enforce EV while maintaining the abovementioned requirements.

4.2 Design Process

In trying to introduce EV to the current SPVS, different security stages, each with its benefits and disadvantages, were created and evaluated. The idea of each stage is presented in the following list, starting with the smallest change.

- **Stage 1** QR codes for all eligible voters on the VSC
  The idea of this stage was to print a QR code with the voters' credentials on the VSC, which the tallying office can then scan. However, the tallying office usually doesn't have access to the citizen records, so no cross-checking of the received envelopes can occur, which annuls the whole purpose.

- **Stage 2** QR codes posted on private Smart Contract
  A secure way to enable the tallying office access to verify the eligibility of the received envelopes is to introduce a smart contract containing the voting records. When scanning the QR code, the envelope is automatically cross-checked with the smart contract data to verify the reliability of the EV. This stage prevents many security threats; however, it isn't enough for eligibility verifiability since there is no proof for the public. It does mitigate some threats analyzed by Killer and Stiller [23] .

- **Stage 3** QR codes posted on a public Smart Contract
  In addition to stage 2, the voters get access to the Smart Contract to track their envelope. Voters can scan the QR code with an app, and then on tallying day, they can verify if the envelope has been received at the tallying office. On a results page, the publlic can see all UUIDs used to vote for Elegibility Verifiability.

- **Stage 4** QR code on the VSC and a QR on the VE for postal tracking
  To mitigate more aspects of the threats: TE6, TE7, TE8, TE9, TE10, an additional QR code is added to the outer layer of the voting envelope to be tracked by the Swiss Post. The first QR on the VSC tracks the envelope at three stages: when printing, optionally when casting by the voter, and when tallying. With the second QR code, the steps in between are covered better.

- **Stage 5** A decentralized Identity Management Software
  Introducing identity management, e.g., Proverum would also cover eligibility verifiability and additional security standards; however, a lot more complexity is involved. [24]

This paper focuses on the third stage since it is the simplest method to fulfill the above-mentioned requirements.

A Smart Contract is just one possible means to convey the ER to the tallying office. The municipality could also send it in the same way as in the A. Setup, where the municipality sends the ER snapshot by e-mail (which happens in many municipalities) [23]. However, this option has a few significant downsides. First, it is a big privacy concern since sending confidential information like the name and addresses of all eligible voters by e-mail is a significant security breach. Secondly, it would introduce new threats to the threat events because it would be pretty easy to tamper with or add non-eligible or fake voters to this list to commit voting fraud. Another downside is the amount of added trust which has to be placed in the voting authorities. Total control is again given to the employees sending, receiving, and using this ER snapshot in the tallying office. A step further into digitalization would be to host a server online to add this ER snapshot, rather than just sending it by e-mail. This step adds a layer of security; however, this solution still requires the same trust placed on the authorities since the valuable information is still controlled from a centralized point. It would also still easily be possible to manipulate the snapshot.

Therefore using smart contracts on the blockchain is the optimal solution to reap the benefits of the added security of the decentralized database. It significantly reduces the possibilities of manipulation since the audit trail on the blockchain is immutable; no authority can go in and change the information without it being detected [12].

4.3 Design Overview

The design adds to the following five phases of the SPVS, abstractly summarized in Figure 4.1.

- **A. Setup** A randomized integer is created for every voter, which on the one hand is posted to the initialized blockchain and, on the other hand, added to the snapshot of the ER so the external supplier can turn this randomized integer into a QR code and print it on the VSC.
- **C. Casting** When casting their ballot, a voter has an additional choice of scanning the QR from the VSC with an app on their phone, which then verifies the authenticity of the voting envelope.
- **E. Tallying** At the tallying office, an additional step is performed when opening the outer envelope (VE): the QR code is scanned, and the randomized integer is then verified on the smart contract to 1. verify that the integer exists and 2. to verify that this integer hasn't already been used to cast a vote before. If those checks go through, the ballot is verified and tallied as usual. If it isn't, the appropriate authorities will be called to analyze this use case. Additionally, the voter can go on the same app to check whether or not the tallying office received their ballot. Also, the authorities can be notified if the ballot isn't received if a voter tracks their ballot with the app provided.

- **Results** After the Verification phase the public gets access to all the random integers used in the voting process, along with the voting results.
- **G. Destruction** Along with destroying the paper ballots, the randomized integers on the smart contract are also destroyed once the Results phase is complete.

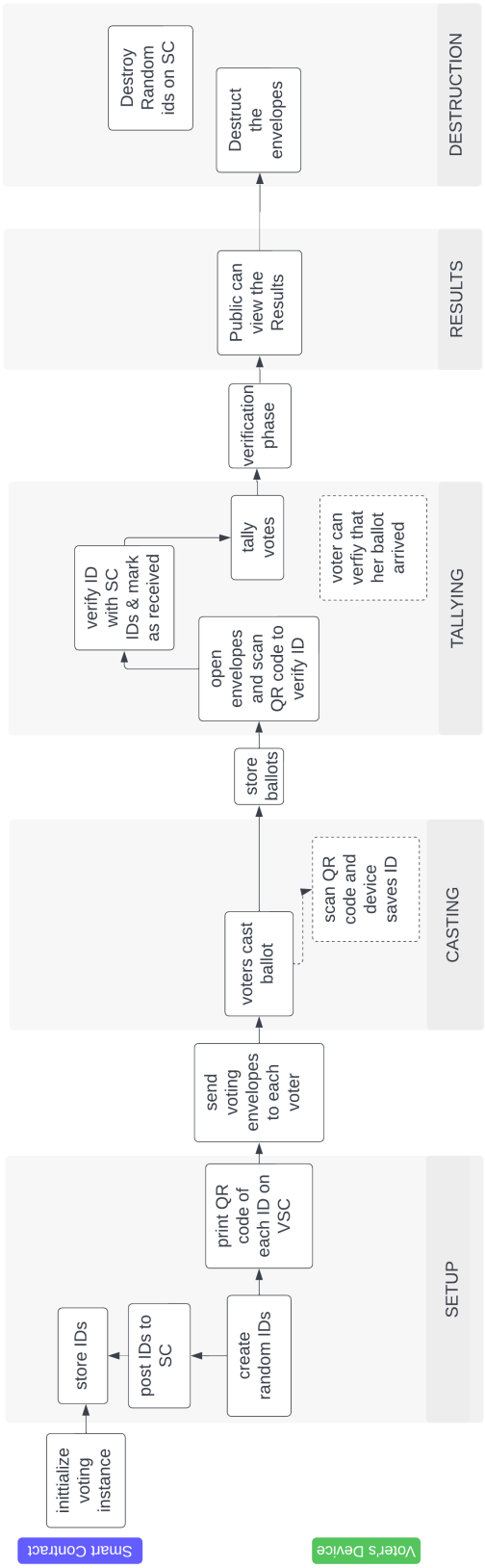

Figure 4.1: Stakeholder Callgraph Design Overview

Chapter 5

Prototype Implementation

5.1 Preparation

A voting cycle begins in the Setup phase, and different authorities must complete specific tasks before instantiating a new voting instance on the Smart Contract. Before sending a snapshot of the ER to the external supplier (ES), the municipality authorities need to create as many unique random integers as there are eligible voters. The integer isn't linked to the voters in any way (it doesn't contain the voter's id or any other credentials); it is solely there to represent an eligible voter. UUID v4 generators generate random integers. While the previous UUID versions (1-3) depend on some metadata of the device used, the 4th version is almost entirely random (6 of the 128 bits are used to indicate the UUID's version). The advantages of the fourth id version are its anonymity and randomness, leading to an unpredictable set of integers that is impossible for an attacker to guess. However, there are disputes about its security since there is a slight chance of generating duplicate ids. Some sources don't recommend using UUIDs v4 as 'security capabilities', i.e., when the only requirement to receive access to a system is to possess the correct id [9, 10]. Since no other access is given to the owner of any UUIDs, there aren't any concerns about using these ids as random integers. A second security check is in place when registering all ids into the smart contract since the SC doesn't allow the registration of the same id twice. So a secure Smart Contract can mitigate this security hazard.

Once the UUIDs are generated, they are added to the Smart Contract in the steps below and sent to the external suppliers (ES) and the ER. It is important to note that the integers aren't matched to any voter yet; the two documents (list of integers and the ER are sent separately). When receiving the documents, the external supplier generates QR codes for each UUID and then randomly prints them on the VSC. It is required of the ES not to record which id was matched to which voter in any way.

5.2 Smart Contract

The implementation of the prototype is split into two parts: the smart contract (SC) and the user interface (UI). The smart contract is built in the Solidity programming language on the Ethereum Blockchain using Truffle and Infura to deploy a demo version of the contract and MetaMask as the wallet. It is a small, concise smart contract made up of a few essential pillars:

- the metadata (name and date) of the voting instance
- an eligible voter datatype containing the UUIDs and a boolean whether it's already been used to vote
- the eligible voters' registry mapping: consists of all the UUIDs as keys that point to the corresponding eligible voter datatype
- the SC voting phases, which can only be entered in the predefined order (the phases differ slightly from the voting phases introduced by Killer and Stiller and are discussed in the paragraph below[23])
- the functions which allow interaction with the abovementioned variables

There are two types of functions in the smart contract: reading data from the SC and writing data onto the SC. Three modifiers ensure the security of the SC; two ensure that each function can only be called in a specific phase, and the third restricts which addresses may call which functions. Two of these modifiers are shown in the listing 5.2. For instance, some functions can only be called by the address which instantiated the new voting cycle. Therefore, each function is maximally restricted by who has permission to call it and in which voting phase the function is allowed to be called. For example, registering eligible voters for a new voting instance can only be done in the Setup phase; the function isn't callable in any other phase, and the corresponding authorities can only call it.

```
1 modifier onlyOfficial(){
2     require(msg.sender == ballotOfficialAddress);
3     _;
4 }
5
6 modifier inPhase(Phase _phase){
7     require(phase == _phase);
8     _;
9 }
```

Listing 5.1: Smart Contract Modifiers

Another aspect of ensuring the security of the variables is the privacy quantifier by Solidity itself. Variables defined with this quantifier are only visible inside the Smart Contract and can't be called by any external source [41]. By making all the vulnerable variables (e.g., the list of all UUIDs) private, the only way to access its information is through getter functions. These getter functions can be restricted with modifiers to, for example,

only allow access to the citizen registry after the Tallying phase is complete. In 5.2 the `allIntegers` variable is set to private, and its information can only be accessed through the `fetchUUIDbyIndex` function in the Results phase.

```
1   string[] private allIntegers;
2
3   // Results: Fetch all IDs one by one
4   function fetchUUIDbyIndex(uint index)
5     public
6     inPhase(Phase.RESULTS)
7     view
8     returns (string memory)
9   {
10     return allIntegers[index];
11  }
```

Listing 5.2: Smart Contract Visibility Quantifier

There is only one voting instance running at a time. Each voting instance must go through all SC phases in the following order: Uninitialized, Setup, Voting, Tallying, Results, and Destruction. Some phases directly align with the Swiss Postal Voting Phases, and some serve different purposes. The phases ensure the encapsulation of different functionalities to certain time frames. Each phase is further explained below.

- **Uninitialized** In this phase, there is no current voting cycle, no data exists on the blockchain, and it is ready to be initialized for a new voting instance.
- **Setup** The Smart Contract Setup phase directly corresponds to the SPVS' Setup phase. Voting Officials can initialize a new voting cycle by giving it a name and a voting date, the address that initializes the instance is the official ballot address. The second part of the Setup phase is posting all UUIDs on the blockchain, which the voting officials can only perform in this phase. Once all ids are entered, a button click can push the voting cycle into the next phase.
- **Voting** This phase corresponds to three SPVS phases delivery, casting, and storage phase. During this time, no new ids can be added, no id data can be queried, and no ids can be verified as received yet. It is an in-between phase to restrict the beforementioned functionalities to specific time frames in a voting cycle.
- **Tallying** On the official tallying day, the voting officials can move the voting cycle into the Tallying phase, which corresponds to the Tallying and Verification phase of the SPVS. In this phase, ids can be verified and marked as received. Once tallying is finished and the Verification phase has passed, the smart contract can be pushed into the next phase.
- **Results** This is an added phase after SPVS Verification is complete, where all the election data is available for online review. Here the results and all the ids are posted to ensure Eligibility Verifiability.
- **Destruction** Once a certain time has passed, and the latest before the next election begins, the Destruction phase begins. All voting cycle data is erased in this phase, and the smart contract is back to the Uninitialized.

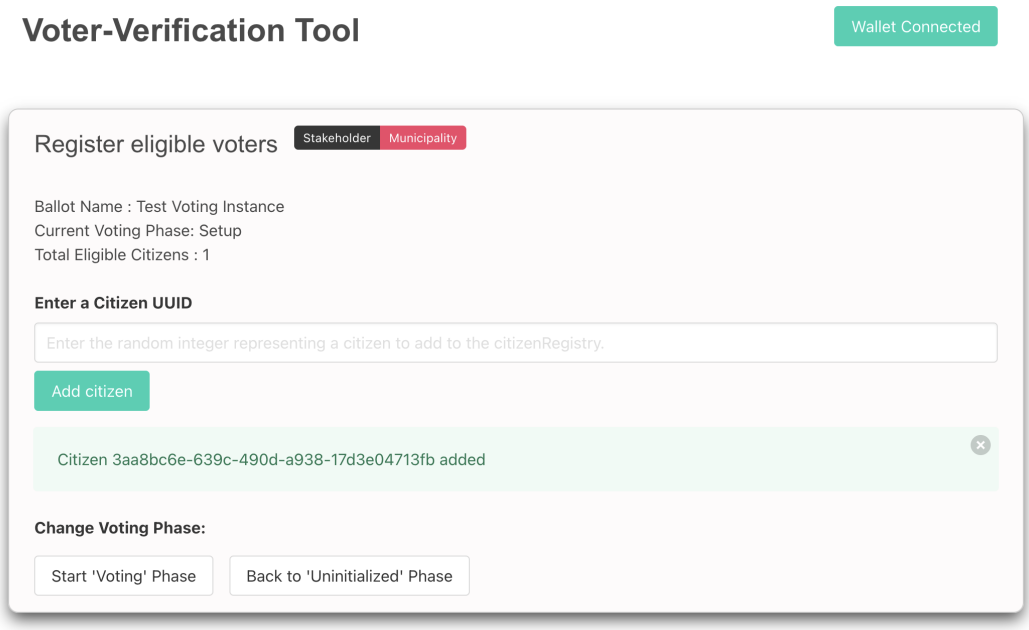

Figure 5.1: Register Eligible Voters UI Layer

5.3 User Interface

The technologies used for the UI are javascript with the Next.js framework for building react applications, which connects to the smart contract via web3. It includes a QR code scanner library called html5-qrscanner[28]. The UI consists of four layers for the three major stakeholders, each corresponding to one or multiple voting phases. Every layer interacts with the smart contract, each in its own way. Similar to the smart contract functions, the functionalities of the different layers are only accessible if the voting instance is in the correct phase. The following list further explains each UI layer in the same order as a voting instance would take place.

5.3.1 Setup Layer

The first layer is solely used by the municipality authorities, and the voting phase begins as Uninitialized. On the Web-app the authorities must first connect to the blockchain via the MetaMask wallet, the address used to instantiate the voting instance is the only address that will have access to all the 'authority-only' functions. Once connected, the setup layer enables the corresponding authorities to instantiate a new voting instance by initializing it with a name and a date. Once the SC confirms this request, the second SC phase, the Setup, begins. At this point, the authorities add all random integers to the eligible voters' list in the smart contract as seen in Figure 5.1. Once complete, the voting instance is placed into the 'Voting' phase by a button click.

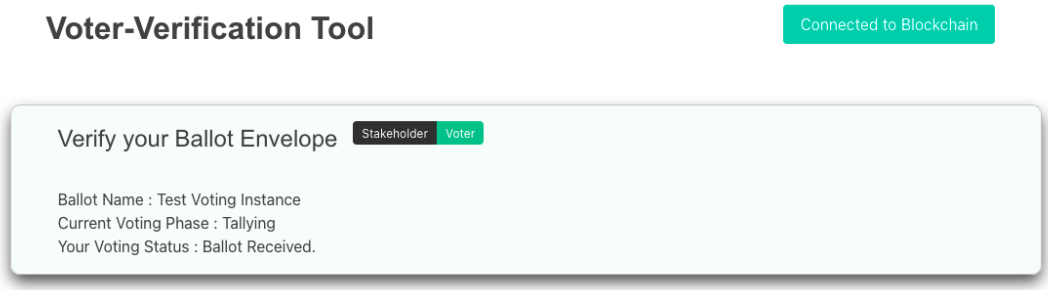

Figure 5.2: Voter's Device UI Layer

5.3.2 Voter's device Layer

This layer is a progressive web app (PWA) built for the voters to load onto their smartphones. In practice, each voter could download the app and scan the QR code on their VSC, this prototype however is web-based. During the Voting Phase, the app scans the QR code and retrieves the UUID stored on the user's phone. They will not receive confirmation that the UUID scanned is on the blockchain's list of UUIDs. During the Tallying phase, the voter can return to the app and verify that the tallying office successfully received their VE, as shown in Figure 5.2. The voter's device layer doesn't need to be connected to the blockchain via a wallet since the functionalities only include reading data. It is sufficient to access the deployed SC on Infura via the application binary interface (ABI), achieved by a simple button click. The voter neither has access to any other data except the information of their specific random integer nor do they have access to any post functions to manipulate the SC.

5.3.3 Tallying Layer

The third layer is a PWA, which is used in the tallying office on the tallying day. Firstly, once tallying has begun, the corresponding authorities change the SC Voting Phase to 'Tallying'. Next, they scan each envelope's QR code with the app, as demonstrated in Figure 5.3, which triggers two smart contract actions. Firstly, a read action verifies that the received envelope corresponds to an eligible voter who hasn't placed their vote yet, by checking whether the random integer exists in the eligible voters' list and whether a ballot with that random integer has already been received. And secondly, a write action is executed to change the 'voted' status of that random integer to 'received' to ensure that each random integer can only be used once to place a ballot. Once the tallying phase is finished, the authorities push the voting instance into the Results Phase. Once the final results are published, and the paper ballots are destroyed, the authorities can end the voting instance with a button click. This action temporarily puts the voting instance into the 'Destruction' phase, triggering the erasure of the entire list of eligible voters. Once this is finished, it is automatically put into the Uninitialized phase, and a new voting instance can begin.

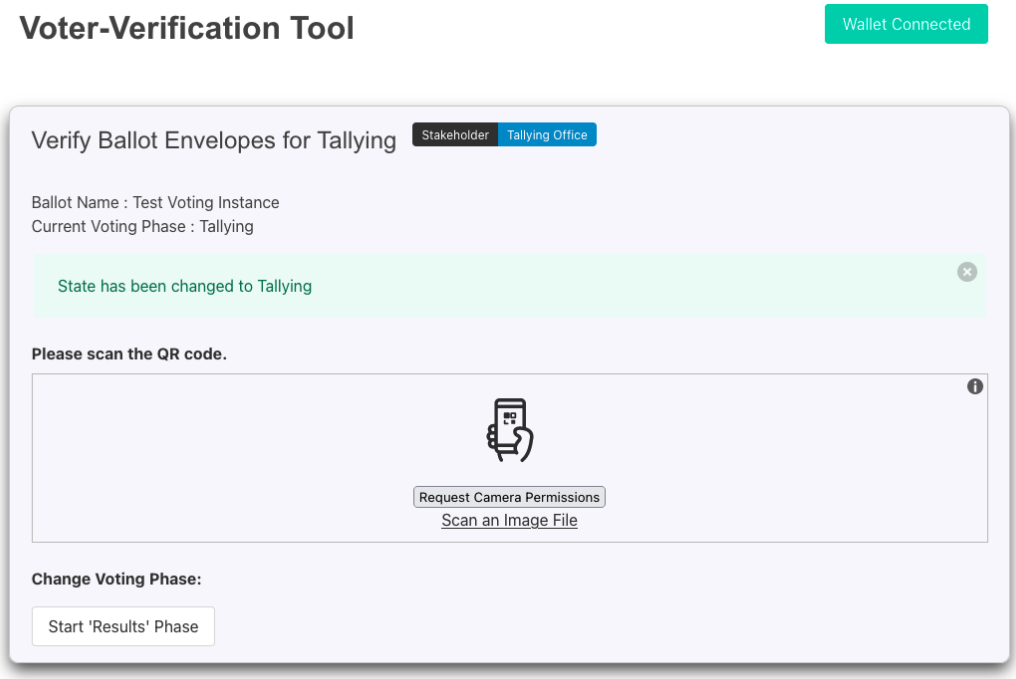

Figure 5.3: Tallying Office UI Layer

5.3.4 Results Layer

Once the SPVS Tallying and Verification phases are complete, it is time to publish the results. On top of all the regular voting proceedings, the public now has access to further information on the Results UI. While the Smart Contract remains in the Results phase, it displays all the UUIDs used in the voting instance to achieve Eligibility Verifiability. An example with some fictional results is shown in Figure 5.4. Once a specific period is over, all the data is destructed, and the Results page is empty until the next voting cycle begins.

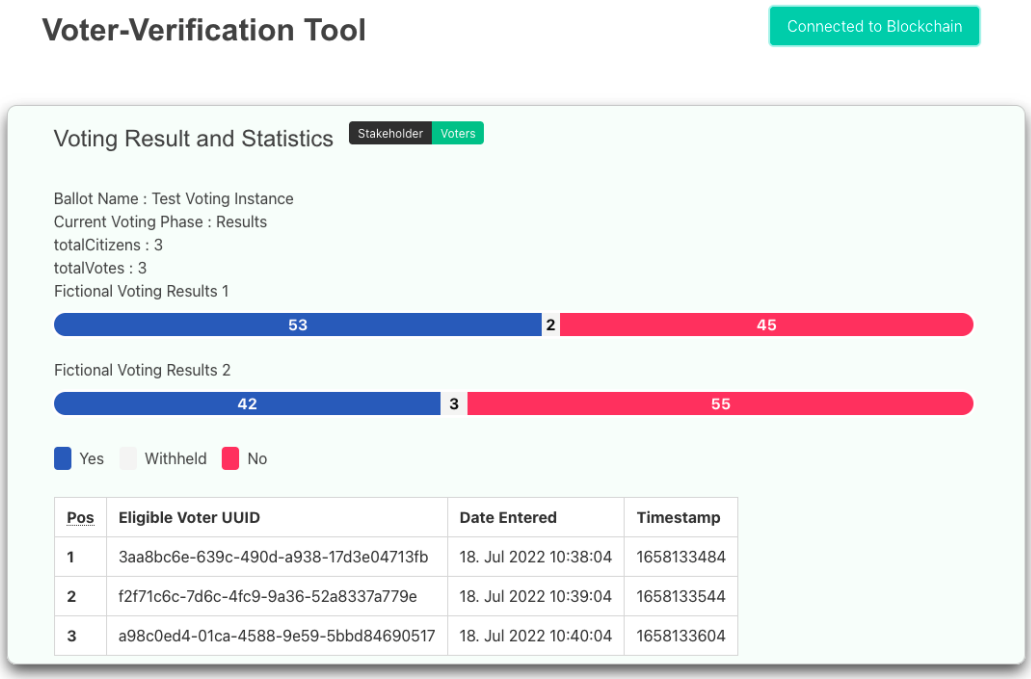

Figure 5.4: Result UI Layer

Chapter 6

Evaluation

6.1 Privacy

As established above, the current SPVS ensures Ballot Privacy and Receipt-freeness (based on a few assumptions). Adding the verification prototype to the SPVS doesn't diminish these privacy standards in a significant way, but it doesn't improve them either. The newly introduced system remains equally receipt-free as the current SPVS; there is no method for any voter to produce proof of how they voted. Whether that person follows the voting protocol or not. Even if an attacker knows the voter's UUID to check whether it's contained in the published list of ids after the vote, it is still no proof of how the person voted. Therefore there is no incentive for an attacker to blackmail a voter into voting a certain way or into voting at all since voting envelopes can also be sent empty. Ballot Privacy, on the other hand, is up for discussion. Fact is, there is a possibility that an attacker can figure out the UUID of a particular voter and then check whether that person cast their vote or not. However, the attacker cannot determine how any voter casts their vote since the actual ballot is always completely separated from the random id and the smart contract. An attacker has two means of discovering the id of a specific voter: either they manipulate the ES in the Setup phase while printing the QR codes on the VSCs, or they intercept and open an envelope undetectably during the Delivery-Storage phase.

6.2 Verifiability

The sole verifiability standard the Swiss postal-voting system demonstrates is the Cast-as-Intended (CaI) Verifiability due to its paper-based nature. The newly designed prototype perfectly maintains this standard since it remains a paper ballot. Further steps in the End-to-End Verifiability, namely Recorded-as-Cast (RaC) and Tallied-as-Recorded (TaR), aren't achieved because of their difference from Eligibility Verifiability. RaC ensures that the tallying authorities record the ballot cast by the voter correctly. To achieve RaC, the actual ballot choice needs to be verified (e.g., whether the voter voted 'yes' or 'no' for an

initiative). McMurtry, for example, achieves this by sending the ballot decision digitally and analogically by a paper ballot. The tallying office can then cross-check the vote on paper with the electronic vote to verify that the decision hasn't been manipulated [27]. However, the prototype introduced in this thesis only marks whether the VE is received and not whether the ballot decision is then recorded correctly. The tallying authority scans the QR code on the VSC, and once verified, the VSC and the voter's identity are separated from the ballot choice. If a voting envelope is intercepted in the Casting or Storage phase and the decision is somehow changed, this prototype system wouldn't be able to detect that attack. It only verifies that the ballots received at the tallying offices are legit envelopes corresponding to an eligible voter who hasn't voted yet. In order for a system to achieve RaC, this ballot manipulation would need to be detectable.

When considering the other measurement of verifiability, Individual and Universal Verifiability, the same explanation as above applies. The prototype adds neither of those definitions to the current SPVS. The only things being tracked are the VSCs, not the ballot choice itself. Individual verifiability isn't given through the User layer or the public blockchain since the voter can only verify whether their ballot was received and not whether their choice was recorded/tallied correctly. Similarly, with Universal Verifiability, the public blockchain in this prototype isn't enough to prove voter choices' accuracy.

This prototype introduces an additional layer of verifiability: Eligibility Verifiability (EV). Thanks to the blockchain, the public can verify that an eligible voter has cast each vote. As Bernhard and Warinschi [4] state, to ensure EV in an election, eligibility information (e.g., a list of all eligible voters) must be published to verify that an eligible voter has cast every voting envelope. When the random integers, each corresponding to an eligible voter, are added to the blockchain in the setup phase, the addition is timestamped. Once the voting results have been published, these random ids are published along with that timestamp. This information provides the public with proof that only envelopes with an authentic id have been tallied.

Eligibility Verifiability would also be given with fewer steps. Assuming a voter couldn't scan any QR code and therefore wouldn't know which random id belonged to them and all the other steps remained the same. EV would still be given through the publication of the timestamped integers on the blockchain. Inspired by the alternative approaches used in related work, such as the continuous auditing in the Prêt 'a Voter, this additional step serves as further proof for an individual voter. Adding this layer doesn't improve the verifiability standards themselves, however, as discussed in the paragraphs below, it does help mitigate some threats.

6.3 Software Independence and Accountability

This prototype adds an additional piece of software and doesn't change any of the existing software in use; hence software independence still isn't given. However, the prototype itself is software-independent, i.e., if the current SPVS were software-independent, it would remain that way if this prototype were added on top. The prototype doesn't have any direct contact with the actual ballots cast and the final tally; hence if there is an

undetected error or change in the system, it doesn't influence the outcome of the final result. Accountability is maintained, and its relevancy is slightly reduced due to the prototype's decentralized nature since less trust has to be placed on the authorities.

6.4 Threat Analysis

Based on the threat analysis made by Killer and Stiller, each threat is reiterated to evaluate how the prototype influences the threat [23]. A summary of the evaluation is shown in Figure 6.1.

- TE1: This threat is unchanged.
- TE2: No changes since this happens before the prototype is set into place.
- TE3: This step is slightly securer since the number of eligible voters must match the number of UUIDs sent to the ES. Any addition or removal of entries in either the snapshot or the UUID list will be detected. However, the snapshot of the ER is still susceptible to tampering if solely the voter data is changed (e.g., change name and address of eligible voters to non-eligible voters) or if both lists are tampered with equally.
- TE4: Adding the prototype completely mitigates this threat assuming the attacker doesn't have access to the list of UUIDs of the current voting instance. Since every envelope is scanned for validity when tallying, any forged envelopes with QR codes of UUIDs that aren't in the smart contract will be detected.
- TE5: The prototype doesn't mitigate physical theft, it still relies on the voters to notify a missing envelope.
- TE6 and TE7: The prototype doesn't mitigate any threats in the delivery process. As long as one envelope per valid UUID arrives at the tallying office, no re-routing or theft of envelopes that the voters haven't received is recognized.
- TE8: Since voters have the option to track their VSC to verify whether the tallying office received their sent envelope, the prototype can slightly alleviate this threat concerning the destruction of envelopes. However, it depends on how many voters actively track their envelope and notify the authorities if the tallying office never receives their sent envelope. This threat is not detected if an attacker opens and modifies the envelopes.
- TE9: Similarly to TE8, this threat is slightly mitigated concerning the destruction of the VEs.
- TE10: Forged VEs will be detected every time (assuming the attackers don't have access to the UUIDs). Concerning the destruction of VEs, as above, it depends on the number of voters that track their ballots and then notify authorities when they don't get the 'received' verification on their app.

- TE11: The destruction in Storage is also undetected unless voters track and notify their missing ballots via the prototype app. However, modification of the ballots isn't detected at all.

- TE12, TE13 TE14: After scanning the QR code and separating the VSC from the ballots, the process is the same as in the SPVS. The prototype doesn't mitigate any threats after this step, so the manipulation of tallying, the final tally, or premature destruction are all still present.

In some mitigations above, it is assumed that the attackers do not have access to the list of UUIDs, but what happens if the attackers intercept the list in the SETUP phase? This event doesn't add any additional threats to the current SPVS; however, it can weaken the threat mitigations of the prototype. In this case, the attackers can forge envelopes (which requires a lot of knowledge [23]) and add the UUIDs onto the VSC. It makes forging the envelopes even harder since there is the added step of intercepting them in the setup phase, however, it still can be done. There are only two ways in which attackers can gain access to the UUIDs of a particular voting instance: they intercept the UUID list when it is being sent to the ES in the setup phase, or they steal envelopes in the Delivery - Storage phase. Since the voter device app doesn't confirm the validity of the scanned UUID, trying to query the blockchain with random UUIDs doesn't work either. The authorities in the tallying office scan these forged envelopes, and the fraud is detected if any eligible voters corresponding to the stolen UUIDs also cast their ballot. Once a UUID is encountered twice, the authorities are alerted. So the chance of detection depends on how many envelopes they forged and what percentage of voters cast their vote. As a small side note to detecting forged envelopes, if an eligible voter scans the QR code on the app but doesn't cast their ballot, the voter can notify the authorities if the app shows a 'received' confirmation. In the SPVS without the prototype, forging isn't mitigated at all, so no new threats are introduced this way.

6.5 Requirement Analysis

In order to evaluate the prototype, the following section recapitulates the requirements set in the design process and analyze how well each one is fulfilled.

- **Maintain the current privacy and verifiability standards**
  The current privacy standards (fulfilled Ballot Privacy and the lack Coercion-Resistance) are maintained since no further data about the voters is leaked by adding the prototype (based on a few assumptions).

- **Improve either privacy or the verifiability standards or both**
  This requirement is met concerning verifiability; the prototype adds Eligibility Verifiability (EV). With the additional eligibility checks added by the prototype, which each voter has the option to verify with their ballot envelope, each voter can assume every other ballot cast is eligible. The privacy standards aren't improved with the prototype.

Phase	TE	Description	Evaluation
A	TE1	Delay production of physical artifacts	○
A	TE2	ER master records	○
A	TE3	ER data snapshot	◉
A	TE4	Forge physical aartifacts	●
A	TE5	Steal assembled VEs before dispatch	○
B	TE6	Re-route VEs	○
B	TE7	Steal VEs from voter letterboxes	○
C	TE8	Steal VEs from municipal letterboxes	◉
C	TE9	Re-route VEs	◉
C	TE10	Cast stolen or forged VEs	●
D	TE11	Access stored VEs	◉
E	TE12	Manipulate tallying	○
E	TE13	Manipulate final tally	○
F	TE14	Initiate premature destruction	○

Figure 6.1: Threat Event Evaluation [23]

- **Prevent as many current threats as possible**
  The prototype mitigates quite a few of the threat events analyzed by Killer and Stiller. Most of the mitigation is focused on preventing the forging of envelopes, and once the ballot has been cast, the detection of the destruction of envelopes is also increased. The prototype is also successful in not adding any additional threat events.

- **Ease of use for all stakeholders involved.**
  For the voters, the SPVS with the prototype is as simple as it is without it since any steps for the voters are optional. And even the optional step is easy to use and, therefore, easy to be trusted for an average voter since it includes technologies that are already very omnipresent in our society (downloading an app and scanning a QR code). Also, the additional steps are straightforward for the municipalities, any other involved authorities, and external suppliers.

6.6 Additional Effort

Since this prototype is an added layer to the current SPVS, it will make the voting process a bit more complicated and expensive. In order to manipulate the Ethereum blockchain by interacting with the smart contract, a gas fee must be paid. This SC requires gas for registering each UUID onto the blockchain, changing the phase of the voting instance, marking each UUID used to vote, and then deleting all the UUIDs. Therefore there is a write action for every eligible voter in Switzerland (on average around 5.5 million people), a negligible 6 write actions for the voting phases. The average voter turnout in Switzerland in 2019 was around 45%; therefore, approximately 2.25 million UUID markings for each

vote cast, and another 5.5 million deletion actions [18]. The gas prices are estimated when running the Hardhad gas reporter plugin on each of these functions.

Each write action has a specific gas price already converted to the current price of Swiss Franks CHF. The one-time costs independent from the number of UUIDs added for each voting instance are the instantiation which averages at 1.44 CHF, and each phase change costs around .60 CHF. This is a negligible cost of 3.35 CHF for each voting instance. This test suite adds 132 UUIDs; therefore, the price for adding one id is around 3.2 CHF. For 5.5 million eligible voters, this averages to approximately 17.6 million CHF. Marking the ids used for voting in the tallying phase costs around 1.44 per citizen who voted, usually around 2.25 million people, and therefore sums up to 3.24 million CHF. The last step is the destruction of all the UUIDs on the blockchain, which is approximately 0.63 CHF per eligible voter, which sums up to 3.465 million CHF. In total, each voting instance would cost around 24.3 CHF million additionally.

Chapter 7

Discussion

Voting is the key to accessing politics as an eligible citizen in Switzerland; as a direct democracy, sovereignty lies with the people. The public carries a lot of political power [13]. The current Swiss postal voting system heavily relies on the trust of the public places since there is little to no verifiability of the voting outcomes given to the public. While the system achieves a moderately high standard of privacy, the many threats analyzed and the lack of verifiability opens the system up to many attackers. According to Jonker et al., there seems to be a trend in the voting field, where system designers first lay their focus on privacy and then later on verifiability. This is evident as well in Switzerland; the SVPS has a heavy emphasis on privacy. However, there is a significant lack of verifiability. Therefore, much of the current research on the Swiss voting field attempts to improve its verifiability standards. This paper serves as a proof of concept in achieving higher verifiability in the SVPS while maintaining the current privacy standards.

7.1 Privacy Discussions

While it is clear that the proposed SPVS with the prototype remains as receipt-free as the current SPVS, it does not introduce coercion-resistance. The newly introduced threat must be discussed to deduce whether the new system upholds its ballot privacy standards. As mentioned in the evaluation, the only additional information an attacker could access is whether or not a particular voter casts their vote. I.e., not how this person cast their vote, solely if the tallying office received their ballot. By the definition of ballot privacy, it reveals more of the voters than exclusively their vote if the outcome is unanimous. While their voting choice will remain equally private as before, one can argue that the information on whether someone casts their vote or not might be valuable for the public image of the voter. For example, a politician might be seen as a lesser candidate if it comes to light that they never cast their vote. This notion might weaken the trust the public places in a voting system. On the other hand, several arguments demonstrate that this possibility is negligible. This attack on the integrity of voters solely depends on the attacker getting access to the UUIDs of the corresponding voter, which is theoretically possible, as explained in the evaluation and further discussed in the next paragraph.

Since the ES preparing the EVs must assign and print the QR code ids randomly, and they are prohibited from keeping track of the matches, it should be impossible for anyone to gather the knowledge in this step. The chance of an attacker unnoticeably intercepting VEs in the Delivery-Storage phase is higher, and this attack is highly unscalable. Suppose an attacker is targeting a specific person. In that case, the only real possibility of finding the right envelope is once it is placed in the voter's mailbox before the voter empties it. The chance of intercepting a single specific VE in the postal office or the storage phase is improbable. Observing the letterbox of a victim 24/7 to intercept the VE without being detected is possible but not scalable. The current SPVS ballot privacy is based on a few assumptions, which are a much higher threat to the integrity of its voters. For example, if an attacker manages to access the VEs in the Storage phase, they can open all envelopes and publish which eligible voters voted and how they voted. Based on the assumption that the two beforementioned possibilities are negligible, the newly introduced SPVS maintains its ballot privacy.

7.2 Verifiability and Threat Discussions

Even though Eligibility Verifiability is often neglected in related work, it is a crucial part of voting systems. It ensures the integrity of the voting results in two ways; it only allows eligible voters to cast their vote and assures each eligible voter only casts their vote once. The introduced prototype manages to introduce EV to the current SPVS. Since many of the threats in the SPVS identified by Killer and Stiller are mitigated when introducing EV, adding this prototype significantly improves the security standards of the SPVS [23]. Some threat events carry multiple aspects. Often, they include manipulation, forgery, and/or the destruction of the VEs. This prototype manages to completely mitigate forgery in every phase of the voting process since the tallying office verifies the QR code of each VE before tallying. If an attacker completely forges the random ids, it is detected. If they get access to some ids from eligible voters and use them in forged envelopes, it is also very likely to be discovered. Especially if this attack is scaled up, which is the only way to influence the voting outcome. As explained in the evaluation, manipulating the ballot choice of VEs in any phase between Casting and Tallying has an equal chance of remaining undetected as in the current SPVS without the prototype. The Destruction of the VEs, however, is a mixed case. If VEs are destructed after Casting, there is a chance that it is detected if many people use the app to track their VEs and then report any instances of fraud. While this is an added step to mitigate this threat in the SPVS, it isn't as strong as the mitigation of forgery.

Unlike other related work, like McMurtry et al., this prototype remains a two-way paper-ballot system and automatically maintains CaI Verifiability. This is a crucial part of the system's simplicity and the trust placed on the system by the public. However, this prototype doesn't manage to achieve the verifiability standards McMurtry et al. introduce in their paper on the SPVS. The key difference is the information that is being tracked. Like Gjoesteen et al., this prototype doesn't track the actual ballot choice, only the outer envelope. It is difficult to argue which proposal is 'better' since a voting system has to be judged on many pillars. However, when solely focusing on verifiability, this prototype is outshined by many other proposals in related work [27, 26].

Under the new verifiability standards, the the impact of the blockchain of the protype must be discussed. While RPV has many benefits, adding a blockchain to voting systems does have its advantages. The key difference to previous suggestions proves to be the decentralized nature of the blockchain. This aspect alleviates a lot of the trust placed in authorities by installing a tamper-proof audit trail. An approach to keeping complex technology simple and trustworthy in the eyes of the public is introducing alternative methods for the voters. As related work has been done with short strings, continuous auditing, and so on, this prototype provides an additional layer for the voters. Enabling each voter to scan their QR code with a provided app on their smartphone and then allowing them to track that envelope places some power back into the voters' hands. They don't solely have to rely on complex technologies, and they can see some proof for themselves if they choose to.

7.3 Limitations

This prototype is a proof of concept that increasing the verifiability standards is possible while maintaining privacy in the SPVS. It does have many limitations though. On the one hand, it is a single prototype that hasn't been scaled to the 5.5 million users it would hold. If it were introduced to Switzerland, every of the 2148 municipalities would need to create their own voting iterations. Such scaling could bring forth some unforeseeable problems. While the cost estimation for 5.5 million users is too high for the prototype ever to be used in its current state, the estimate itself is very vague. Since it heavily depends on the time of day, day of the week, and many other factors concerning the Ether and gas prices. Adding a cost of 24.3 million CHF to each voting cycle which costs 7.5 million Swiss Franks, isn't worth the improvement. On top of that 24.3 million CHF, a lot of administrational tasks would need to be fulfilled. Election officials would need to be schooled on using the prototype, a task force must inform the citizens to retain their trust in voting systems, and computer scientists would need to be hired to maintain the prototype. As this paper's main focus was to provide a proof-of-concept to retain and increase its voting standards, investing time into making it a cost-efficient prototype is a task for future work.

Proving that remote postal voting systems can be improved in their verifiability while remaining privacy shows that RPVS are a strong contender in the voting field. RPVS has many benefits: the remote aspect increases voter turnout and enables home-bound citizens to vote, it retains the public's trust since it is a straightforward, easily-understandable system, and a paper audit trail is difficult to infiltrate with high-scale attacks. While there are some excellent internet voting proposals in related work, political voting remains a high-coercion environment where internet voting is more vulnerable to high-scale attacks. While there is a worldwide trend of transferring tasks to the internet, political voting is a special case. The difference between other important internet transactions, for example, significant financial payments through online banking, is the lack of verifiability for the users. When anyone makes a financial transaction through the internet, all parties involved can easily verify that their transaction was executed exactly how they intended it to be. However, in voting systems, there mustn't be any connection between the voter's identity

and the vote cast due to the requirement of the voter's privacy [32]. While this prototype introduces EV, it still lacks any other type of verifiability besides Cast-as-Intended. The prototype does not verify whether the ballot decision cast by the voter is accurately portrayed in the voting results.

Chapter 8

Summary and Conclusions

8.1 Summary

This paper comprises an analysis of existing voting schemes, varying from in-person poll site systems to electronic voting trials. These systems are evaluated according to the foundations of voting security: verifiability, privacy, software independence, and accountability. The focus lies on the use case of the Swiss remote postal voting scheme. Its weak points are analyzed, and a prototype is designed to improve the current SPVS as simply as possible.

The different security layers of privacy and verifiability in voting schemes are discussed to provide foundational knowledge. These layers are used to analyze the current Swiss postal voting system, which consists of seven phases, from Setup to Destruction. This system focuses on the voters' privacy and relies on the people's trust due to its lack of verifiability. The basics of blockchain solutions are introduced along with the foundation of voting systems. Namely, the Ethereum blockchain and the Smart Contract enable publishing code to interact with the blockchain. Related work was summarised into four sections: in-person vs. remote voting and paper-based vs. electronic voting. With a focus on remote paper-based voting, related work was evaluated to design a prototype improving the current security standards.

An analysis of the current security threats of the SPVS revealed that manipulation of the eligible voter's list (either in the citizen registry or as voting envelopes) remains among the most significant threats. Similar to previous papers, the prototype's design focused on keeping track of the voting envelopes to mitigate this threat. A simple blockchain solution was designed to alleviate as many threats as possible and achieve a higher verifiability standard. It is to be used alongside the seven SPVS phases. In the Setup phase, the municipality generates as many random ids as there are eligible voters. These ids are posted on the blockchain and are sent to the ES to be printed on the VSC of each voting envelope as QR codes. In the casting phase, each voter can use an app to scan this QR code to verify later whether the tallying office received their ballot. In the tallying phase, the officials scan the QR code to validate the authenticity of the envelope and mark each random id as received. The VSC is then separated from the actual vote to be tallied. In

the last step, part of the blockchain is made public to achieve Eligibility Verifiability, and then before the next voting cycle, all the digital data is destructed.

This prototype was implemented via Smart Contract in Solidity on the Ethereum blockchain with three interaction layers for the different stakeholders. It was then evaluated and discussed upon the set requirements and its usability. It serves as a proof-of-concept to infuse the current SPVS with Eligibility Verifiability and as mitigation to some SPVS threats, primarily focused on forgery.

8.2 Conclusions

This paper provides a step towards higher verifiability in remote postal voting systems without compromising its current privacy standards. It combines the privacy standards of postal voting and adds Eligibility Verifiabliaty through the decentralized nature of blockchain. It also enables the voters to verify whether the tallying office has received their ballot to break down the new technology's complexity. This paper alleviates some of the trust placed in the electoral officials by providing a decentralized tamper-proof audit trail on the blockchain. While the current cost of the prototype limits its usability in the current SPVS, this approach demonstrates possibilities for improving verifiability without diminishing its privacy or ease of use for the voters. It proves that remote postal voting is still a strong contender for politically binding elections worldwide.

Further studies can focus on keeping paper-based voting as a foundation and building more verifiability on top of them. Specifically improving this prototype's efficiency and scalability, or in a broader context introducing verifiability in general in remote paper-based voting schemes. However, keeping in mind that the political environment in which elections are held as a whole, rather than just the effectiveness of the voting process, shapes the foundation of public trust. Future work can focus on voting environments as a whole, investigate how to maximize voter turnout, and how to lessen the need for the public trust in the political climate by improving the voting systems themselves.

Bibliography

[1] Adida, B. Helios: Web-based Open-Audit voting. In *17th USENIX Security Symposium (USENIX Security 08)* (San Jose, CA, July 2008), USENIX Association. https://www.usenix.org/conference/17th-usenix-security-symposium/helios-web-based-open-audit-voting.

[2] Amundsen, B. No more online voting in norway. *Sciencenorway.no* (September 2019). Retrieved April 15, 2022, from https://sciencenorway.no/election-politics-technology/no-more-online-voting-in-norway/1562253.

[3] Barben, D. Moutier-abstimmung: Folgt das nächste desaster? *Der Bund* (2021). Retrieved March 12, 2022, from https://www.derbund.ch/brisantes-papier-weckt-unbehagen-677279356631.

[4] Bernhard, D., and Warinschi, B. Cryptographic voting - a gentle introduction. In *FOSAD* (2013).

[5] Bernhard, M., Benaloh, J., Halderman, J. A., Rivest, R. L., Ryan, P. Y. A., Stark, P. B., Teague, V., Vora, P. L., and Wallach, D. S. Public evidence from secret ballots, 2017.

[6] Center, P. R. The voting experience in 2020. *Pew Research Center* (2020). Retrieved March 22, 2022, from https://www.pewresearch.org/politics/2020/11/20/the-voting-experience-in-2020/.

[7] Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., and Vora, P. Scantegrity: End-to-end voter-verifiable optical- scan voting. *IEEE Security and Privacy 6*, 3 (2008), 40–46.

[8] Chevallier-Mames, B., Fouque, P.-A., Pointcheval, D., Stern, J., and Traoré, J. On some incompatible properties of voting schemes. In *Lecture Notes in Computer Science* (05 2010), vol. 6000, pp. 191–199.

[9] Choren, M. How secure are your universally unique identifiers (uuids)? - why uuids should not be used as security capabilities. *VERSPRITE* (03 2020). Retrieved June 16, 2022, from https://versprite.com/blog/universally-unique-identifiers/.

[10] Cortier, V. Formal verification of e-voting: solutions and challenges. *ACM SIGLOG News 2* (Jan. 2015), 25–34.

[11] Cuvelier, Ã., Pereira, O., and Peters, T. Election verifiability or ballot privacy: Do we need to choose? In *Lecture Notes in Computer Science* (2013), Springer Berlin Heidelberg, pp. 481–498.

[12] Di Pierro, M. What is the blockchain? *Computing in Science Engineering 19*, 5 (2017), 92–95.

[13] Eidgenossenschaft, S. Direct democracy. *Schweizerische Eidgenossenschaft* (07 2021). Retrieved June 26, 2022, from `https://www.eda.admin.ch/aboutswitzerland/en/home/politik-geschichte/politisches-system/direkte-demokratie.html`.

[14] Fichter, A. "passwort: Wahlen" - der technische hintergrund und das glossar zur recherche. *Republik* (2020). Retrieved March 03, 2022, from `https://www.republik.ch/2020/09/25/passwort-wahlen-der-technische-hinter-grund-und-das-glossar-zur-recherche`.

[15] Formation, . M. â. A. C. Install metamask for your browser. *MetaMask*. Retrieved July 20, 2022, from `https://metamask.io/download/`.

[16] Foundation, O. Download. Retrieved June 06, 2022, from `https://nodejs.org/en/download/`.

[17] Fujioka, A., Okamoto, T., and Ohta, K. A practical secret voting scheme for large scale elections. In *AUSCRYPT* (1992).

[18] IFES. Electionguide - democracy assistance and election news. *IFES - International Foundation for Electoral Systems* (12 2019). Retrieved June 24, 2022, from `https://www.electionguide.org/countries/id/207/`.

[19] Inc, I. Infura documentation. *INFURA* (2022). Retrieved June 26, 2022, from `https://infura.io`.

[20] Indergand, I. Verifiability in the swiss remote postal voting system. Master's thesis, University of Zurich - Department of Informatics (IFI), 2021.

[21] Jonker, H., Mauw, S., and Pang, J. Privacy and verifiability in voting systems: Methods, developments and trends. *Computer Science Review 10* (2013), 1–30.

[22] Killer, C., Rodrigues, B., Scheid, E. J., Franco, M., and Stiller, B. *From Centralized to Decentralized Remote Electronic Voting.* to Be Published.

[23] Killer, C., and Stiller, B. *The Swiss Postal Voting Process and Its System and Security Analysis.* 09 2019, pp. 134–149.

[24] Killer, C., Thorbecke, L., Rodrigues, B., Scheid, E., Franco, M., and Stiller, B. Proverum: A hybrid public verifiability and decentralized identity management, 2020.

[25] Kremer, S., Ryan, M., and Smyth, B. Election verifiability in electronic voting protocols. In *Computer Security – ESORICS 2010* (2010), Springer Berlin Heidelberg, pp. 389–404.

[26] Kristian Gjøsteen, Clémentine Gritti, K. N. M. Ballot logistics: Tracking paper-base ballots using cryptography. *E-Vote-ID* (2020).

[27] McMurtry, E., Boyen, X., Culnane, C., Gjøsteen, K., Haines, T., and Teague, V. Towards verifiable remote voting with paper assurance, 2021.

[28] Minhaz. Html5-qrcode, 2015-2022. Retrieved June 01, 2022, from `https://github.com/mebjas/html5-qrcode`.

[29] Miragliotta, N., Laing, M., and Thornton-Smith, P. *A Review of convenience voting in the state of Victoria*. Electoral Regulation Research Network, Oct. 2018.

[30] Neff, C. A. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2001), CCS '01, Association for Computing Machinery, pp. 116–125.

[31] O'Sullivan, D. How the world's most frequent voters handle postal ballots. *SWI swissinfo.ch* (october 2020). Retrieved April 15, 2022, from `https://www.swissinfo.ch/eng/how-the-world-s-most-frequent-voters-handle-postal-ballots/46070666`.

[32] Peter Wolf, Rushdi Nackerdien, D. T. Introducing electronic voting: Essential considerations. *IDEA - International Institute for Democracy and Electoral Assistance* (12 2011), 36. Retrieved June 30, 2022, from `https://www.corteidh.or.cr/tablas/28047.pdf`.

[33] Ryan, P. Y. A., Bismark, D., Heather, J., Schneider, S., and Xia, Z. Prêt à voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security 4*, 4 (2009), 662–673.

[34] Schumacher, S., and Conaughton, A. From voter registration to mail-in ballots, how do countries around the world run their elections? *Pew Research Center* (2020). Retrieved March 22, 2022, from `https://www.pewresearch.org/fact-tank/2020/10/30/from-voter-registration-to-mail-in-ballots-how-do-countries-around-the-world-run-their-elections/`.

[35] SEZC, . S. C. L. Request testnet link. *Chainlink*. Retrieved June 20, 2022, from `https://faucets.chain.link/`.

[36] Specter, M., and Halderman, J. A. Security analysis of the democracy live online voting system. In *30th USENIX Security Symposium (USENIX Security 21)* (Aug. 2021), USENIX Association, pp. 3077–3092. `https://www.usenix.org/conference/usenixsecurity21/presentation/specter-security`.

[37] Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., and Halderman, J. A. Security analysis of the estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2014), CCS '14, Association for Computing Machinery, pp. 703–715.

[38] Swinford, S. Postal voting fraud is 'easy', electoral commissioner says. *Daily Telegraph* (april 2015). Retrieved March 20, 2022, from https://www.telegraph.co.uk/news/uknews/law-and-order/11560017/Postal-voting-fraud-is-easy-electoral-commissioner-says.html.

[39] Tagesschau. Der günstige preis der direkten demokratie. *srf.ch* (05 2016). Retrieved June 28, 2022, from https://www.srf.ch/news/schweiz/der-guenstige-preis-der-direkten-demokratie.

[40] Tiwari, S. An introduction to qr code technology. In *2016 International Conference on Information Technology (ICIT)* (2016), pp. 39–44.

[41] tutorialspoint. Solidity - contracts. Retrieved June 06, 2022, from https://www.tutorialspoint.com/solidity/solidity_contracts.html.

[42] Wackerow, P. Gas and fees. *Ethereum Blockchain* (06 2022). Retrieved June 26, 2022, from https://ethereum.org/en/developers/docs/gas/.

[43] Wackerow, P. Intro to ethereum. *Ethereum Blockchain* (06 2022). Retrieved June 26, 2022, from https://ethereum.org/en/developers/docs/intro-to-ethereum/.

Abbreviations

ABI	Application Binary Interface
CaI	Cast-as-Intended
DL	Distributed Ledger
EO	Election Office
ER	Electoral Register
ES	External Supplier
ETH	Ether
EV	Eligibility Verifiability
EVM	Ethereum Virtual Machine
E2EV	End-to-end Verifiability
FC	Federal Chancellery
IV	Individual Verifiability
PB	Paper Ballot
PBE	Paper Ballot Envelope
PVPF	Postal Voting Process Flow
PWA	Progressive Web App
RaC	Recorded-as-Cast
RFID	Radio Frequency Identification
RPV	Remote Postal Voting
SC	Smart Contract
SP	Swiss Post
SPVS	Swiss Postal Voting System
TaR	Tallied-as-Recorded
TE	Threat Event
UI	User Interface
UUID	Universally Unique Identifier
UV	Universal Verifiability
VE	Voting Envelope
VSC	Voting Signature Card
WBB	Web Bulletin Board

List of Figures

Listings

Appendix A

Installation Guidelines

This prototype uses Solidity, Truffle, Bulma version 0.9.4, Html15-qrcode version 2.2.1, next.js version 12.1.6, react.js 18.2.0, solc version 0.8.15, and web3.js version 1.7.3 on runs the macOS operating system on Firefox.

1. **Setup**
   (a) Node.js is a prerequisite to run this system; if it isn't installed, download the installer from the nodejs.org website [16].
2. **Clone Repositories**
   (a) Clone the Frontend GitHub repository by running
   `$ git clone https://github.com/LexHeggli/verify-voter-app`
   (b) Clone the Smart Contract GitHub repository by running
   `$ git clone https://github.com/LexHeggli/verify-voter-sc`
   (c) navigate to the root of the SC project with `$ cd verify-voters-sc`
3. **Run Smart Contract**
   (a) Install all dependencies using `$ npm install`
   (b) navigate to the root of the project with `$ cd verify-voters-sc`
   (c) Create a .env file in the root of the project. Two variables will be saved in this file, a private key from the wallet provider and the Infura Website.
   (d) Set up an Infura account on the website [19] and sign into the account. Create a new project and navigate into the project settings. Change the Endpoint to **Rinkeby** and copy the second URL, the URL starting with `wss://rinkeby.infura.io/ws/...`. In the verify-voters-sc, open the .env file and add `INFURA_API_URL=` and then directly add the URL.
   (e) Install Metamask as a browser extension if it isn't already installed. Create a Metamask account [15] and select the account to be used for this project. Select the three dots on the side to see the setting and click on Account details.

In order to export the private key, enter the MetaMask password and copy the key. In the verify-voters-sc open the .env file and add `PRIVATE_KEY_1=` and then directly add the password. An example of an env file is shown in 3e.

```
1 PRIVATE_KEY_1=
    cf9034b2ae12fg4c59fdedt74434c234e2654dbr54e1d40c3m387fdbdk7c17e6

2 INFURA_API_URL=wss://rinkeby.infura.io/ws/v3/
    c6652e4d32834d95bfhdf8f4d1fhcc0r
3
```

Listing A.1: Smart Contract env file

(f) In the root of the project run `truffle migrate --network rinkeby`. Once that deployment goes through, go under `2_verify_voter_migration.js` in the deployment output and copy the contract address.

(g) In order to test the Smart Contract on Ethereum, some test Ether is needed. On the Chainlink website [35], click on 'Connect Wallet' and follow the steps to connect chainlink to the MetaMask account. Next, ensure the '0.1 test ETH' checkbox is checked, fill out the captcha, and send the request. Once the request is complete, there should be 0.1 test Ether on the MetaMask account.

4. **Start Frontend Application**

(a) Navigate to the root of the project with `$ cd verify-voters-app`

(b) Install all dependencies using `$ npm install`

(c) In the blockchain folder, open `verify.js`, in this file, replace the current URL located in the WebsocketProvider in the second line with the Infura URL from step 3 d from above.

(d) In the same file, paste the contract address from Step 3 f into the `contract_address` variable.

(e) Run `$ npm run dev` to run the application and to receive the localhost link.

(f) Open a browser with MetaMask installed and run the localhost link.