Gemeente Groningen

Berenschot

LAB15

COUNTING VOTES!

Because every vote counts

PARTNERS

GEMEENTE GRONINGEN // BERENSCHOT // LAB15

ABSTRACT

The aim of this document is to provide an in-depth understanding of how the Dutch Election process can become more trusted and transparent when applying Distributed Ledger Technology (DLT).

There is evidence that the current election process in the Netherlands is insecure and prone to errors and fraud. In an attempt to improve the security and transparency of democratic elections and prevent errors and fraud from taking place, a software application based on DLT was created.

The solution was designed to count, process, and aggregate vote results and was tested in a Pilot during the referendum of the 21st of March 2018 in Groningen, The Netherlands. To give public accountability of this process, all transactions derived from the ledger were published on a public website.

Based on the results of the pilot, the solution presents itself as a promising, and more reliable, alternative for the current OSV software. More research and testing is needed however to test the projects' hypothesis that using an immutable and transparent database will increase the trust that society has in democracy.

1 INTRODUCTION

Voting is an essential act for the functioning of a democratic state. It is the collection of a nation's opinion on rights, candidates, and referendums. To avoid civil conflicts and fraud, most advanced civilizations reached the consensus that to elect citizens representatives and approve rights, citizens must be able to vote in a transparent ceremony with guaranteed vote secrecy.

Nations worldwide have tried various methods to make elections more trustworthy and transparent using both hardware (electronic voting) and software (support for counting and aggregating votes), whilst others tried to optimize the conservative approach using ballot paper and pencil. Despite these efforts the quality of elections has deteriorated considerably over time.[1]

Evidence of electoral fraud has been provided globally, both in developed as well as developing countries.[2] When consulting a list of notable elections involving accusations of direct voter fraud it returns eighty-five(!) different cases in the past 20 years only, ranging from developed nations such as the United States, Hong Kong, Austria and Hungary, to more developing nations such as Venezuela, Uganda and Kenia.[3] Although such strong accusations never occurred in The Netherlands, there are strong indications that the current process and tools used for elections held in the Netherlands are not safe.

The motivation to start this Pilot originates from questions asked in Parliament about the current insecure and error-prone election process.[4] Last year, together with cyber security expert Sijmen Ruwhof, RTL media performed an information security analysis of the counting votes process, which currently uses the OSV software.[5] This revealed critical security breaches that have existed since 2009, exposing vulnerabilities to hackers. Additionally, many incidents were reported after the House of Representatives election in March 2017, such as the 9,000 votes in Den Bosch that were not included in the overall count.[6]

Furthermore, organizing a referendum or election is very time consuming. Not just in the months preceding the event, but also both on the day itself as well as after the election, when votes are aggregated in the OSV software. It requires a considerable effort from civil servants and volunteers and depends on serious amounts of financial resources. In response to these and other events, the Electoral Council concluded that the process for counting votes for elections should be improved.[7]

1 Electoral fraud and voter turnout – Vardan Baghdasaryan et al. 2015

2 Electoral fraud and voter turnout – Vardan Baghdasaryan et al. 2015

3 https://en.wikipedia.org/wiki/List_of_controversial_elections

4 https://www.rijksoverheid.nl/documenten/kamerstukken/2017/06/13/kamerbrief-over-fouten-bij-uitslag-tweede-kamerverkiezing-15-maart-2017

5 https://www.rtlnieuws.nl/nederland/politiek/zo-werkt-het-softwaresysteem-dat-onze-stemmen-telt

6 https://www.bd.nl/s-hertogenbosch/in-totaal-zo-n-9000-stemmen-bij-verkiezingen-vergeten-in-kieskring-den-bosch~a0d16fddf/

7 https://www.kiesraad.nl/adviezen-en-publicaties/kamerstukken/2017/kamerbrief-evaluatie-tweede-kamerverkiezing-2017/kamerbrief-evaluatie-tweede-kamerverkiezing-2017/kamerbrief-evaluatie-tweede-kamerverkiezing-2017

2 OBJECTIVE

Following the Electoral Council’s recommendation, this project was launched upon the hypothesis that when votes are registered directly on an immutable and transparent database, the trust that society has in democracy will increase. Therefore, the main objective of this project is to test whether DLT (“*Distributed Ledger Technology”*) can assist in counting, processing and aggregating votes, ultimately resulting in the strengthening of our democracy by facilitating fraud- and error-free referendums and elections with greater transparency. Furthermore, it aims to make it easier for Polling Station members to correctly count and register votes.

Distributed Ledger Technology can be described as a set of technologies that provide the infrastructure for a ledger in which transactions or contracts are maintained in a decentralized manner, across different locations (nodes), eliminating the need of a central authority to keep a check against manipulation. Unlike traditional databases, distributed ledgers have no central data storage or administration functionality. All the information on it is securely and accurately stored using cryptography and can be accessed using keys and cryptographic signatures.

Once information is stored on the ledger, it becomes immutable and is governed by the rules of the network. This also implies that no single actor can modify or control data without all the others’ permission. In DLT each node verifies and processes each item, thereby generating a record of each item and creating consensus on each item’s veracity. While centralized ledgers are prone to cyber-attack, distributed ledgers are inherently harder to attack because all the distributed copies need to be attacked simultaneously for an attack to be successful. Blockchain is probably the best-known type of DLT.

If the Pilot proves the hypothesis that votes can be registered directly with the help of DLT, and thereby create more trust and transparency in society, there is an ambition to scale the solution to a more robust platform. A platform that is able to serve the purpose of counting votes nationwide in any of the various elections, including: Dutch Parliament, European Parliament, Provincial States, Local Council and Water Board elections, and of course referendums.

3 CRITERIA

Following the objective of this Pilot to validate DLT as the underlying technology for counting votes for elections, the Pilot needs to take criteria formulated by the Electoral Council into account.[8] These criteria are used to assess the Pilot’s proposed solution and viability. The list below provides an overview of these criteria and gives a description of how the Pilot takes these criteria into consideration.

[8] Commissie-Korthals Altes - STEMMEN MET VERTROUWEN - Adviescommissie inrichting verkiezingsproces

3.1 TRANSPARENCY

The Electoral Council explains the Transparency criterion as follows: "*the election process must be designed in such a way that the setup and structure is completely transparent, so that in principle everyone has insight into it. In short, this means that there should be no secrets within the election process, and that all questions should be answerable in a verifiable manner.*" Therefore, the election process must be designed in such a way that the setup and structure are completely transparent so that in principle everyone has the ability to review every step. By using DLT in this Pilot, data can be viewed by everyone, offering complete transparency of the counting votes process. With DLT, each addition of data is stored in an immutable and traceable way. Information generated by every individual Polling Station can be followed in near-real-time.

3.2 VERIFIABILITY

The Electoral Council explains the Verifiability criterion as follows: "*The election process must be objectively verifiable.*" Given that all data registered in DLT will be publicly available it appears to be one of the best tools to verify the election process for society. Because the code that verifies the validity of polling cards, calculates whether the generated control numbers match the totals of counted votes, and registers votes in an unchangeable and traceable is public, the whole process can be followed, verified and checked by everyone.

3.3 INTEGRITY

The Electoral Council explains the Integrity criterion as follows: "The election process must be conducted correctly and based on predefined rules. The outcome must not be influenceable other than by the casting of legitimate votes."

Within this Pilot the integrity of the election and vote counting process is guaranteed, because DLT records all data unchangeably and performs calculations reliably via 'Smart Contracts'. Therefore, everything will be processed according to the predefined rules. Since no identity information is used and since the voting process itself does not get affected by this system, the integrity of the election process is safeguarded.

3.4 VOTER ELIGIBILITY

The Electoral Council explains the Voter Eligibility criterion as follows: "*Only people eligible to vote may participate in the election.*" Today, people have the option to vote in a Polling Station of choice. This complicates the verification of polling card validity (and therefore voter eligibility) slightly. By registering polling card ID's, DLT can help Polling Station staff to verify whether polling card codes are valid or have already been used elsewhere.

3.5 VOTING FREEDOM

The Electoral Council explains the Voting Freedom criterion as follows: "*Each voter must be able to determine his or her vote in full freedom, free of any influence*." The current regular voting process of voting on ballot paper with red pencil remains unchanged in this Pilot, which therefore still guarantees voting freedom. The Pilot is just an auxiliary process to count the number of votes after the voting process has ended.

3.6 SECRECY OF VOTES

The Electoral Council explains the Secrecy of Votes criterion as follows: "*It must be impossible to establish a link between the identity of a voter who casts the vote, and the content of that vote. The process must be designed in such a way that it is impossible to let the voter prove for whom or on what he or she voted.*" Verifying voters' eligibility, issuing votes, and counting the votes are three separate processes. In this Pilot the three processes do not cross each other digitally, because all votes will still be issued on ballot paper with pencil as normal, guaranteeing anonymity of the voter. It is therefore also impossible to establish a link between the identity of a voter and the content of a vote.

3.7 UNIQUENESS

The Electoral Council explains the Uniqueness criterion as follows: "*Each voter may, under the Dutch electoral system, cast one vote per election, which can and must be counted exactly once.*" By registering polling card ID's, DLT can help Polling Station staff to verify whether polling card codes have already been used.

3.8 ACCESSIBILITY

The Electoral Council explains the Accessibility criterion as follows: "*Voters should, as much as possible, be given the opportunity to participate directly in the election process. If that is not possible, they must be given the possibility to participate indirectly – by authorizing another voter in the form of a power of attorney.*" This Pilot does not affect the ability of eligible voter to grant a power of attorney to another person entitled to vote.

4 REQUIREMENTS

The criteria set by the Electoral Council led to more detailed requirements that were considered on both the technical side of the solution as well as the process side during the execution of the Pilot itself. Questions that needed to be answered included how to identify Polling Station staff and voters, but also how the solution can guarantee integrity of the information whilst at the same time being completely transparent in every step of the way. The solution should therefore have a communication layer between the user interface and the blockchain to present the transparency.

This chapter will describe how the envisioned solution should work. It defines the requirements on how Polling Station staff is identified, what devices are needed, how the Chairman can navigate the solution, how voter's documents need to be registered, how the control mechanisms should function, how votes should be registered, what the final report should look like, and how users can sign-off the results.

USER ACTIVATION

In a blockchain environment, user activation is a transaction that gets registered. Transactions always occur between wallets; to give wallets to users, they first need to be identified to prevent unauthorized actors to try and issue transactions to the ledger. When users successfully identify themselves, a public and private key get created which activates their wallet. To prevent unauthorized actors from trying to access the system, identification must happen in a secure manner. For the Pilot's purpose, two-factor-authentication is considered to be sufficient. Accounts can only be activated by the Polling Station staff themselves on the device that will be used during the day using the Voting Application.

THE DEVICE

The solution requires a physical device on which polling staff can successfully manage all processes of the election day. The device should have a friendly user interface, be portable, and have an autonomous power source to operate itself throughout the election day in case of power failure. The device must have enough processing power to be able to perform complex operations and execute all tasks. The operating system of the device must be compatible with the technologies used to build the solution. Furthermore, the operating system needs to easily allow for installing a DLT node so that information can be securely distributed between the devices.

DASHBOARD

The solution should have a clear dashboard from where the Chairman of the Polling Station can navigate to the different elements of the process.

VOTER DOCUMENT REGISTRATION

The Pilot will run in parallel with the regular official process of the Polling Station. It therefore needs to precisely record the same information as the official process, including the total amount of polling cards and power of attorneys (either being a private authorization or a written authorization). As this is a Pilot, voters should have the option not to participate. Because of this, polling staff will need to have the option to (anonymously) record objections from voters to the Pilot.

Polling cards should be registered by scanning its Data Matrix code. This should create a registry that prevents a polling card from being used twice. It furthermore should create a digital control number that can be used to verify whether the total amount of votes is matching the total amount of scanned polling cards. An important requirement when creating the registry is that the Data Matrix codes themselves should not and cannot be stored. This means that the original data cannot be derived from the ledger, whilst providing a validation mechanism to avoid duplicity of records.

CONTROL NUMBER #1: PHYSICAL CARDS

As mentioned above, one control number will be generated by scanning the polling cards. This is not part of the official process of the Polling Station. The first official control number generated is based on the total number of counted physical cards. These can be either polling cards, power of attorneys, or voter passes. The combined total is called the 'Total Admitted Voters' variable and should (in the end) match the final 'Total Counted Votes' variable. The control numbers need to be registered on ledger so that they become irrevocable and immutable.

REGISTERING VOTES

Once the 'total admitted voters' are registered, the actual counting of votes starts. It is the Pilot's aim to be completely transparent and trusted in the full process. This means that it should be capable of directly registering each individual vote on ledger, so that it becomes viewable by the public and at the same time becomes irrevocable and unchangeable. The result of the counting process is the variable 'Total counted votes'.

CONTROL NUMBER #2: VOTE TYPES

The second official control number is based on the totals of the piles of the different vote types. These piles are created when votes are sorted by the Polling Station staff on vote type. Since the Pilot will take place during a referendum, there are four different vote types: (1) for, (2) against, (3) blank, and (4) invalid. These four sub totals together need to match the variable 'Total counted votes'.

REPORT

At the end of the counting process, a final report is generated. This report is the result of the counting votes process for a specific Polling Station. The report is the representation of the collected information of the physical cards, the registrations of the votes, and the vote types. When a report is digitally signed by the Polling Station staff it should allow for aggregation on three different levels: the municipality level (to be signed off by the Mayor), the central regional collection station level (to be signed off by the Mayor of where that station is based), and the national level (to be signed off by the Electoral Council). For the Pilot only the first municipality level needs to be in scope.

SIGNING OFF

The Polling Station report needs to be signed off by all the members of the Polling Station. Before the Chairman can sign off, all the Tellers need to have accepted the report first. Only then can the report become final. The Mayor of the municipality can only sign off on the aggregate result once all the Polling Station reports have been signed.

TRANSACTIONS

There are a multitude of transactions throughout the process that need to be registered and published to the public. The solution will register all transactions on ledger, making the solution completely blockchain-centric. The types of transactions are described in the previous paragraphs, but are presented here once again:

- activating account and creating key pairs;
- scanning and hashing Data Matrix codes;
- registered cards (polling cards, powers of attorney, voter passes, and objections);
- entering control number admitted voters;
- counting votes;
- entering control number total amount per vote type;
- signing off the result.

PUBLIC PAGE

With the objective of providing transparency to the referendum process, the results of all transactions performed at the Polling Stations must be displayed on a public website. The website should be capable of displaying the voter turnout during the day, the total number of admitted voters, and the number of votes divided over the different vote types. The first and the third variable should be presented in near-real-time (taking time for mining into account).

5 CONVENTIONS AND DECISIONS

In the context of this Pilot, conventions can be described as the set of technical rules, business models, terminologies, and assets to facilitate communication between the parties involved in the continued process. Decisions concern evaluating different options from a hardware, software, and technology perspective. These decisions take security, scalability, segregation of responsibilities, data integrity, and the objectives of the Pilot into account.

An overview of the different items with a high-level description is presented below. In the subsequent parts of this chapter, more elaborate details will be provided.

ITEM	**DESCRIPTION**
Roles	the different user types within the election process;
Device	machine capable of executing programming logic and connections between other devices;
Runtime	environment that applications are built on top of to provide the necessary abstraction between high level code to the underlying operating system native code execution.
Framework	collection of libraries and common interfaces used by the general programming public to speed up application development.

Tool	piece of software used to assist on project development and deployment that is not directly related to the source code.
User Interface	portion of software capable of allowing the communication and interaction between user and system;
Wallet	special credential consisting of a public key and a private key that enables an authorized user to submit transactions to the blockchain through an address;
Blockchain	set of sequential and immutable records of data transactions generated during the voting process;
Smart Contracts	groups of functions defined in an explicit context to validate the data transactions being sent to the blockchain;
Hash	digested result of a parameter passed through a mathematical function with the purpose of being irreversible.

5.1 ROLES

Every Dutch Polling Station is manned by a team, usually consisting of active members of the local community who facilitate the voting process. Their responsibility is to make sure that the voting and counting process is carried out correctly and is not violated upon. They monitor the processes throughout the day, count and register the total number of votes at the end of the day, and create the final report for their Polling Station. People involved with the referendum have predefined roles.

- The Chairman is a role assigned to the user responsible and accountable for the voting- and counting process within the assigned Polling Station.
- The Teller is a role assigned to the user responsible for helping the Chairman on several tasks, including, amongst others, validating voter's credentials, handing out ballot papers, and counting votes.
- The Mayor is a role assigned to the user responsible for signing the aggregate result of all the Polling Stations in their municipality.
- In this application, the roles are tied to the user's key pair, which is based on their personal email and a password, to allow authentication and authorisation of requests made to the system.

5.2 DEVICE

The device selected for the Pilot is the Microsoft Surface Book 2 i7, because of its strong processing power, memory, storage, and battery autonomy. Bundled with the Windows 10 Professional operating system, the device has an embedded solution easily recognized by users. This device is equipped with enough graphical processing power to handle the user interface tasks proposed in the Pilot. Furthermore, it also has a camera on the backside that allows for the scanning of polling cards.

The decision to choose the Microsoft Surface Book 2 i7 was also based on the ease of deploying DLT nodes, allowing for (near-)real-time sharing of the election information between the Polling Stations.

5.3 RUNTIME, FRAMEWORKS AND TOOLS

For the Pilot there were a few runtimes and frameworks to be taken into consideration. They provide common API's (Application Programming Interface) and baseline infrastructural implementations that abstract common data transformation and transmission procedures. Bundled with tools, they provide the development environment for the Pilot.

The criteria used to determine the development environment were as follows:

- General adoption and broad public availability of information.[9]
- Security: Runtime and frameworks should provide security standards compliance.
- Performance: Runtime and frameworks should provide good performance benchmarks.[10]
- Portability: Runtime should be cross-platform, that is, be able to run on top of different operating systems (i.e. Windows, Linux, macOS).
- Runtime must have blockchain compatible frameworks (i.e. web3js).
- Language targeting the runtime should be easy to use and have general adoption.[11]

To fulfil all those criteria the **Node.js runtime** was selected. It contains a wide selection of frameworks that supports it, it is cross platform, has a high benchmark score, can be targeted with the very popular language JavaScript, and contains many frameworks and libraries to support the development on top of it.

The selected framework to deal with web requests (internet communication) was **ExpressJS**, due to its lightweight nature, wide adoption, security best practices support[12] and active community support and development[13].

The selected library for development against blockchain was **web3js**. Web3js abstracts most of low-level algorithms required to transact with Ethereum blockchain fabric (see chapter 5.5), therefore eliminating the need for custom implementation. Since web3js is compatible out of the box with **Node.js**, it also contributed to the case of choosing it as the runtime environment. Other complimentary frameworks and libraries were used but are out of scope for this document. Please refer to the source code for more information.[14]

To facilitate the development, the tools selected were:

- NPM: A popular and easy-to-use package manager for Node.js applications. It hosts all the necessary libraries and frameworks required by the application.
- Truffle: Provides smart contract compilation, unit testing and deployment capabilities.
- Visual Studio Code: Lightweight and powerful IDE (Integrated Development Environment) with numerous extensions, including Solidity programming language (Ethereum) support.

9 https://insights.stackoverflow.com/survey/2017
10 https://www.techempower.com/benchmarks/
11 https://insights.stackoverflow.com/survey/2017
12 http://expressjs.com/en/advanced/best-practice-security.html
13 https://github.com/expressjs/expressjs.com
14 https://github.com/Dev-LAB15/counting-votes

5.4 USER INTERFACE

To interact with a blockchain-based system, a user interface should be provided. The objective of this software solution was to provide an end-to-end experience instead of only subsystems for blockchain interaction and counting of the votes.

User interfaces can also be powered by frameworks that facilitate development and secure communication with the blockchain subsystems. Without frameworks, user interface development is error-prone, time-consuming and more susceptible to security breaches that have already been taken into consideration on mature frameworks.

The following criteria were considered to determine the user interface development:

- Security: The framework should have built-in security capabilities;
- Performance: The framework should have a small footprint, that is: contain little to no third-party package bundling or dependencies (i. e. jQuery); small size for fast download by a web browser; virtual DOM[15] programming model;
- Compatibility: The framework should be compatible with a wide range of web browsers[16];
- Documentation: The framework should have a very thorough and complete documentation;
- Adoption: Adoption should be broad, which also helps determining the maturity, ease of use, and number of alternate sources of information;[17]

The user interface framework that was judged the best candidate following the criteria above was **Vue.js**[18]. It is a full featured with no third-party dependencies framework, which helps reduce the surface for security vulnerabilities.

Vue.js is fully componentizable, which helps with code reuse. Additionally, due to its stripped-down nature, it is a fully pluggable framework, allowing the developer to choose the components of choice for the different types of action, like communicating with a web server (i.e. axios[19]).

Finally, Vue.js has proven high-performance capabilities[20], which completes the analysis and considerations for the selected user interface framework.

15 https://en.wikipedia.org/wiki/Document_Object_Model

16 https://www.npmjs.com/package/vue

17 https://insights.stackoverflow.com/survey/2017

18 https://vuejs.org/

19 https://github.com/axios/axios

20 https://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html

5.5 BLOCKCHAIN

Blockchain is an emergent DLT (Distributed Ledger Technology) with some key features to set it apart from other DLT technologies. The combined features below make it unique and ideal to the use case.

5.5.1 IMMUTABILITY

One of the greatest challenges in information technology is to guarantee that data has not been tampered with by an internal or external actor. Records in common ledgers or standard database systems can be tampered with and, in many occasions, leave no trace of such tampering. These database systems rely on several surrounding security and auditing measures to prevent such type of malicious actions and are often overcome by hackers or people with privileged access.

To protect the tampering of data, blockchain implementations are designed with immutability as its core principle. Its name already hints this concept: a chain of blocks linked by hash pointers that use the previous block's hash as a parameter to the next block's hash. This means that any change on a block will invalidate any subsequent block, as the resulting hash will change for every block after the altered block.

The decentralized nature of blockchain makes the resistance to tampering even stronger. Even if an attacker could manage to rewrite the entire history of a chain of blocks, it would need to do so to all parties. Without a single point of failure, it becomes increasingly hard to manipulate the chain history. Other DLT implementations have limited or reduced decentralization which increases the attack surface provided by this characteristic.

Therefore, this feature of blockchain was a key element to fulfil the data **integrity** criteria for this Pilot. No other identified technology had a better case on this matter.

5.5.2 TRANSPARENCY

Not every blockchain network fabric provides the same level of transparency; even the same type of blockchain network fabric can have different types of application deployed to it, which implement different policies on the level of information it wants to disclose. The important aspect of a DLT appropriate for this use case is that it must provide the infrastructure necessary for complete transparency. Blockchain is a DLT implementation which by nature already distributes the whole of information transacted to it to every participant on the network; information is completely distributed, therefore allowing the highest degree of transparency possible. It is up to the stakeholders to build hiding mechanisms on top of it to control the disclosure of information (i.e. encryption).

Since one of the criteria defined for this Pilot is **transparency**, the choice of blockchain among other DLT implementation was obvious. It was necessary to ensure that the software implementation on top of it would disclose all necessary information to the entire network, unencrypted. Some DLT implementations could jeopardize this claim as they allow more control over the flow of data (i.e. Corda[21])

[21] https://docs.corda.net/

in such a manner that some parties may not even realize there is data flowing through some specific nodes on the network.

The **accessibility** criteria helped the definition of the specific blockchain implementation. Ethereum remains by far as the most popular blockchain platform[22], therefore containing the highest number of resources to allow public participation on the election process, by monitoring the network or validating the transactions.

Even though the Ethereum network setup is private (for cost effectiveness reasons), participation is open. A private blockchain network can be open to public participation and therefore increasing the level of **verifiability**.

5.5.3 SECURITY

Exposure and transparency also comes with a twist: security. The software solution to run such an important process in the elections should guarantee that information sent to the blockchain is coming from legitimate sources. Although everything that happens on a DLT ledger is traceable, some level of permissioned access should be safeguarded. An example of this is that only authorized Polling Station chairmen can cast vote counts to the blockchain on behalf of a Polling Station. Also, no rogue Polling Station should be able to interfere in a way to mislead the aggregated results.

This is yet another important aspect of Ethereum that distinguishes it from other blockchain implementations. It allows transactions to run under the governance of smart contracts, which are programable pieces of software (further explained in section 5.7) with a specified set of rules that can safeguard legitimacy of the counting process. All smart contracts are executed within Ethereum Virtual Machine (EVM), so its rules are also immutable.

Within the smart contracts it is possible to define the authorized actors (Mayor, Chairmen, Tellers) through digital wallets (explained in section 5.6). The wallets allow for the identification and signing of the transactions on the Ethereum blockchain using safe cryptographic keys that only these actors have access to. With this we can further guarantee the fulfilment of the **integrity** criteria.

Finally, a blockchain network is as secure as the number of participants. For the Pilot, we had five Polling Stations which were also blockchain nodes (running on tablets). We also had four nodes on the cloud[23], totalling nine verification nodes. The main public Ethereum network averages on the order of 20 thousand nodes simultaneously online[24], which has been an effective defence against 51% attack[25] to date. While the Pilot was vulnerable to such an attack due to the small number of participating nodes, a full-scale envisioned implementation of the network will count with around 10 thousand Polling Stations[26] acting as validating nodes, plus any citizen willing to participate and safeguard the network. The Polling Stations alone, by acting as nodes, are already strong enough to almost match Ethereum's public

22 https://cointelegraph.com/news/top-10-companies-of-the-blockchain-industry-in-2017

23 https://en.wikipedia.org/wiki/Cloud_computing

24 https://www.ethernodes.org/network/1

25 https://www.investopedia.com/terms/1/51-attack.asp

26 https://www.kiesraad.nl/verkiezingen/europees-parlement/stembureaus

security strength. This makes it very unlikely and expensive to perform a 51% attack. With population participation that number can increase significantly, which further fulfils the **accessibility** criteria, adding to the level of security.

5.6 DIGITAL WALLET

The Ethereum Wallet is a gateway to decentralized applications on the Ethereum blockchain. It allows a user to hold and secure Ether and other crypto-assets built on Ethereum, as well as write, deploy, and use smart contracts. For context of this Pilot, the digital wallet is a reliable resource generated from the user's key pair, which is generated from the user's personal email and a password. It allows for the authentication of a user's transactions on the blockchain during the voting session.

5.7 SMART CONTRACT

Smart Contracts are programs that allow for the execution of predefined tasks. These programs are distributed and decentralized, as they are stored and executed on a blockchain network. If the predefined rules are met, the program gets automatically executed. This allows for decentralized automation. Because they are stored on a blockchain, they cannot be tampered with, and all nodes can verify the inputs, the code, and the output.

The smart contracts for this application were developed using Solidity 0.4.19, a programming language that inherits influences from C++, Python, and Javascript, enabling automatic routines to be executed on the Ethereum Virtual Machine (EVM). Therefore, the core solution of the application is defined in a set of smart contracts explained in Section 6.1.

5.8 HASH

In computer science, a hash is a mathematical function that converts known data into irreversible digests (hash codes). Within this Pilot, the polling cards, voter passes, and power of attorneys need to be hashed to guarantee anonymity of the voters. To enforce that security, the secure data is bundled with a so-called salt (random data used as additional input to a one-way function that "hashes"[27]), combined with Secure Hash Algorithm version 3 (SHA-3)[28].

Secure Hash Algorithms are cryptographic hash functions published by the National Institute of Standards and Technology (NIST). They are mathematical algorithms that map data of arbitrary size to a bit string of a fixed size (a hash) in a one-way function, making it infeasible to invert the digest.

The SHA-3, formerly called Keccak, uses the sponge construction, in which data is consumed, absorbed, and then squeezed out. It is designed to hash unlimited chunks of data. This enables the application to hash all information in a secure manner to the Ethereum Virtual Machine (EVM).

27 https://en.wikipedia.org/wiki/Salt_(cryptography)

28 https://keccak.team/files/Keccak-implementation-3.2.pdf

6 ARCHITECTURE

The main objective of this Pilot is to test whether blockchain can assist in the process of counting votes, to facilitate fraud- and error-free elections with greater transparency. To address the main objective in a good architectural design, specific elements needed to be considered.

From a user perspective, the solution should provide the Polling Station staff with the best application possible in terms of intuitive design. From a security perspective the solution should guarantee that data is registered in a trusted and safe manner, making it impossible to tamper with. From a societal perspective, the solution should be able to communicate results via a publicly available website in a completely transparent way. These elements resulted in a break-down of the project into four different modules:

- the frontend (ps-ui),
- the backend (ps-backend),
- the blockchain (ps-blockchain),
- the public dashboard (ps-landing-page).

For clarification, the prefix "ps" stands for Polling Station, while the suffixes mean the subdivision of the project, to be described on the following sections.

Modules should be interpreted as parts of a system that aggregates methods in a predefined context, which allows for better code organization. Methods can be described as procedures that are triggered by human or machine interventions on a software process, like the scanning of a polling card. They are part of an algorithm capable of executing routines, providing optional results based on given parameters. Additionally, parameters can be described as predefined variables that may change method behaviour in an algorithm.

The communication between the Frontend and Backend is done through Transmission Control Protocol (TCP), one of the Internet's main protocols. It derives from Internet Protocol (IP), allowing reliable transfer of data between connected devices. The Backend communicates with the Blockchain through WebSockets, an advanced technology that allows an interactive communication session to listen for the Blockchain transaction confirmations. The Ethereum Virtual Machines communicate through a peer-to-peer protocol, a distributed application architecture that partitions workloads between devices.

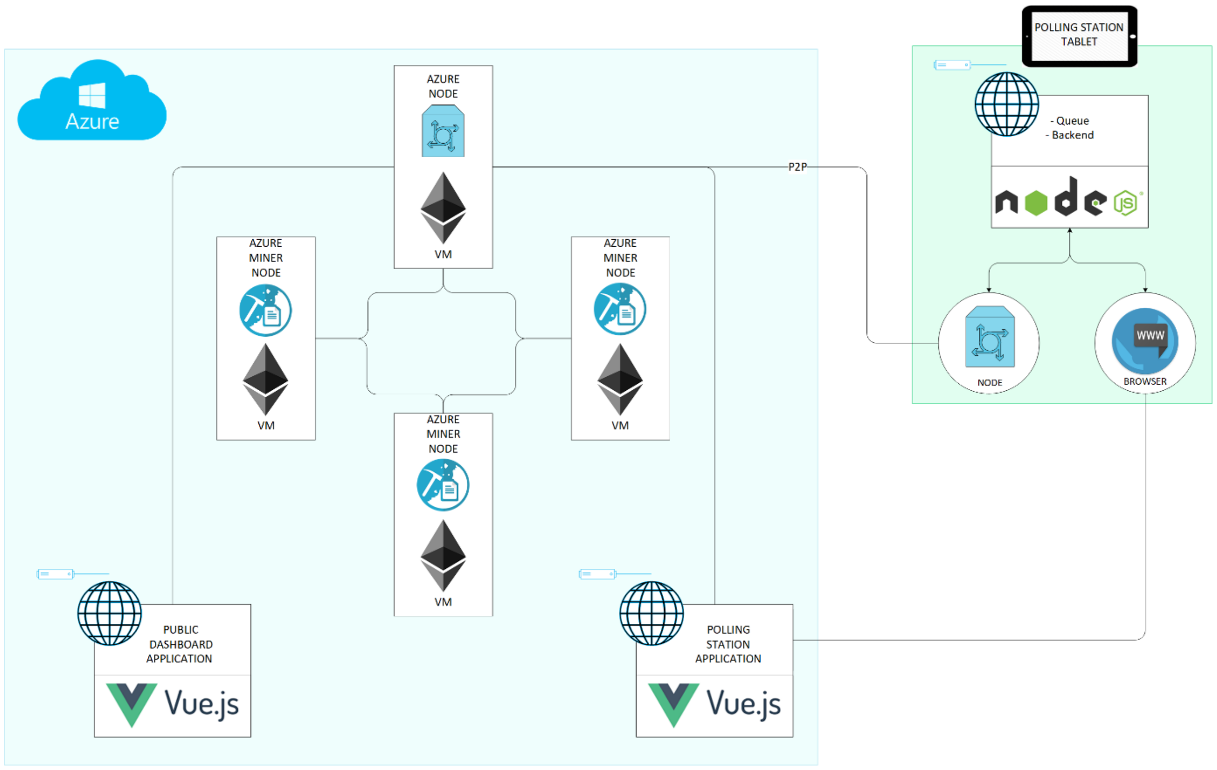


Image 1. Architectural Overview

6.1 PS-BLOCKCHAIN

For this application, all data is validated based on the implemented business logic, and stored in an Ethereum Enterprise network ledger, providing transparency and security.

On the Ethereum Virtual Machine, data archiving and validation is done by using Smart Contract methods. These method calls are possible using signed transactions. When a transaction is confirmed, an event is triggered to the systems that are watching the ledger.

To build Smart Contracts, a proper structure must be defined. To do so, the module structure follows the Technical Specification of the Truffle Framework[29], which states that a contract project must contain a *contracts* folder and the *migrations* scripts. This *contracts* folder contains the contracts written in Solidity 0.4.19, while the *migrations* folder contains the logic to compile the contracts and send it to the Ethereum Virtual Machine in a specific deployment transaction.

During the contract compilation Abstract Binary Interfaces (ABI's) are generated. These can be described as structured documents, used by applications to call smart contract methods on the Ethereum Virtual Machine.

For this Pilot, five smart contracts were created to process and handle the data:

- router.sol: maps the public addresses to the other smart contracts;
- useractivation.sol: handles the transaction authorization based on roles;

29 http://truffleframework.com/

- permissions.sol: abstract contract that validates user roles and permissions;
- pollingstation.sol: handles the logic for the Polling Station counting process;
- municipality.sol: aggregates the results of the elections for the municipality.

6.1.1 ROUTER.SOL

When a smart contract is deployed to the Ethereum Virtual Machine, a public address is returned that refers to the published contract instance. That address will change every time the contract gets re-deployed. During tests this created configurations problems, which is why a contract was created to map the different contract addresses. That contract is called router.sol.

This contract was created to provide an easy configuration loading mechanism for the **Backend** to reach out to the contract addresses for the **useractivation.sol**, **municipality.sol** and **pollingstation.sol**. There is no human intervention on this contract during the application deployment. The relationship between the Polling Station and the **Backend** is stated as simply as associating a machine name to a **pollingstation.sol** contract instance address.

The **pollingstation.sol** contract is directly related to a Polling Station. At the same time, the **municipality.sol** and the **useractivation.sol** contracts can aggregate various Polling Stations in one concentrator. The methods of this contract are described below:

- ***router#setAddress(pollingStationId as string, contractAddress as address)***
  This method relates a pollingstation.sol instance address to a machine name. This contract method needs the EVM owner keys to register Polling Stations using Remix[30], since there is no implementation on the **ps-backend** project to register Polling Stations.
- ***router#getPollingStationAddress(pollingStationId as string)***
  This method shows the public address of the **pollingstation.sol** contract of the specific Polling Station identifier. It is also used by the *ps-backend@config.service#initializeConfig promise*.
- ***router#getMunicipalityAddress(pollingStationId as string)***
  This method shows the public address of the **municipality.sol** contract of the specific Polling Station identifier. It is also used by the *ps-backend@config.service#initializeConfig promise*.
- ***router#getUacAddress(pollingStationId as string)***
  This method shows the public address of the **useractivation.sol** contract of the specific Polling Station identifier. It is also used by the *ps-backend@config.service#initializeConfig promise*.

6.1.2 USERACTIVATION.SOL

There are multiple secure processes in which transactions are generated that need to be authorised and stored on the ledger, including: registering voter documents, registering total number of collected voter documents, counted votes, and signing off on the results. These transactions are supposed to be generated by authorised users, avoiding fraudulent operations. In that sense, the contract

30 https://remix.ethereum.org/

useractivation.sol represents the security of the system, guaranteeing that only authorized users can generate transactions.

This contract is used to setup the users' allowed emails and to query user roles from and to the API. The approved email addresses are stored on the private mappings *useractivation.emailAddresses* and *useractivation.roles*. The mapping *useractivation.usedAddresses* starts with all the added addresses mapping to a false value. The methods of this contract are described below:

- ***useractivation#addEmail(email as string, assignedUserRole as uint8)***
  Adds a user and assigns the role. The roles can be: 1) Owner – temporary wallet used to deploy the contracts only, 2) Mayor, 3) Chairman or 4) Teller. Whenever a user needs to reset his credentials or wallets this method needs to be called again, so that the user can execute the account activation procedure on the Polling Station they are supposed to work.
- ***useractivation#getRoleId(email as string)***
  This method queries the mapping *useractivation.usedAddresses* to display the user role.
- ***useractivation#getUsedEmail(email as string)***
  This method is used to validate if the user already activated a proper wallet to use on the EVM. As a lost wallet cannot be recovered, care is required. However, the **pollingstation.sol** and the **municipality.sol** contract data are safe. If the user by mistake forgets their credentials they need to be reset by use of the method *useractivation#addEmail(email as string, assignedUserRole as uint8)*.
- ***useractivation#setUsedEmail(email as string)***
  This method is called by the backend services when the user makes their first access, guaranteeing that the user is activated just once.

6.1.3 PERMISSIONS.SOL

To enforce the security within the application, an abstract contract was designed to handle authorization rules for the registered users. An abstract contract is a special type of contract that describes common methods, events, and behaviours, to be called by derived contracts. In this application,, **permissions.sol** is an abstract contract, from which the **municipality.sol** and the **pollingstation.sol** are derived. This contract describes behaviours and events to validate the following business logic:

- define user role upon first access for the designed Polling Station or municipality;
- activate the user for system usage;
- verification of the user role for every request.

The **permissions.sol** contract also fires events when the user is activated and when the user tries to make an unauthorized procedure. These events are recorded on the Ethereum Virtual Machine.

6.1.4 POLLINGSTATION.SOL

All events executed on the application in the Polling Station are recorded on the ledger. The **pollingstation.sol** contract therefore forms the core of this project. It is responsible for validating the identity of the Polling Station staff and verifies documents (polling cards, power of attorneys, and voter passes) issued by the municipality. The contract furthermore queries the individual vote count and aggregates the results. Finally, the contract manages the sign off process and submits the final result.

VALIDATING IDENTITY

Every team member has a wallet associated to their Polling Station that is generated by the **useractivation.sol** contract. The pollingstation.sol validates the identity by verifying if its address is mapped to an authorized role. Consequently, a Chairman or Teller from a different Polling Station cannot interfere with the procedures of another Polling Station.

POLLING CARD REGISTRATION

The voter registration starts when the Chairman scans a polling card. After scanning, a hash is generated to guarantee the anonymity of the voter, as described in Section 5.8. A transaction is created to store that hash in the ledger, sharing the information with related Polling Stations to avoid duplicity of the vote. The same happens with the power of attorneys and voter passes. In case voters do not want to participate in the Pilot, the system also gives the possibility to register this objection. To avoid duplicity of the vote, a person can only use a polling card once in a single Polling Station. The registration consists in hashing the code of the polling card by the ps-backend services, sending it to the ledger through the pollingstation.sol contract. After the polling card is registered, the pollingstation.sol contract sends the hashed code to the municipality.sol contract to be stored in the ledger, forbidding future use of the same polling card, but still ensuring anonymity via the application of a salt to the irreversible hash.

CONTROL NUMBER #1: PHYSICAL CARDS

At the end of the day, when the Polling Station has closed its doors, the counting process starts. The first step is to count and register the total number of the polling cards, power of attorneys, objections, and voter passes. This sum is called: Total Admitted Voters. The registration creates a transaction on the ledger, with the purpose of matching the sum against the total amount of counted votes later in the process.

REGISTERING VOTES

The second step is registering the votes. Due to the nature of the referendum, voters had three options: accepting it, rejecting it, or voting blank. To process these votes a fourth option needed to be added in case votes were defined as invalid. Every vote is registered by the Chairman, signed with his keypair, and sent to the ledger as individual transactions. Once the transaction is confirmed, each vote type is stored in temporary variables pre-defined on the pollingstation.sol contract. Those variables are then used in the next step as verification procedure.

CONTROL NUMBER #2: VOTE TYPES

The third step happens when the Chairman enters the totals of the different vote types. The contract will match those with the temporary stored variables. When all the variables match, the Chairman can continue the process; otherwise, votes need to be counted again. The reason for this procedure is to avoid mistakes during the counting process. For this to happen, a transaction is needed to record the number of collected votes, then the verification method is called from the pollingstation.sol contract after the transaction is confirmed.

REPORT

After the verification is done, all vote types are displayed by a read-only call from the pollingstation.sol#getReport contract method. The result of the method call displays two columns of data, as known as collected votes per type and counted votes per type. Also a summary of collected voter documents (an aggregated sum of all the collected voter document types variables) is available to be used by the ps-backend services. The number of admitted voters and the sum of votes are meant to be matched, otherwise the Chairman needs to write an explanation for the deviation.

SIGNING OFF

After all the counting and validation steps are done the final procedure is to sign off the results, by authenticating the Polling Station staff credentials against the system. The final screen of the flow will require each Polling Station Teller to click on his email address, write his password, and thereby issuing their last transaction to the ledger. That transaction will call the pollingstation.sol#signoff method, discarding the user public address, forbidding them to do any further procedure on the system. At the end, the Chairman must do the same, using his password to sign off the final results, which are then reported to the municipality.sol contract, to be displayed on the public page. When this transaction is confirmed, the Polling Station is closed, therefore no other transactions can be issued to the ledger by the Polling Station staff.

6.1.5 MUNICIPALITY.SOL

The final goal of the solution is to summarize the number of the collected votes to display the results on the public page, giving publicity of the referendum to the population. Therefore, an aggregation structure to summarise all collected data on the municipality level contract needed to be developed.

For that reason, this scope was implemented on a smart contract called MUNICIPALITY.sol, with global events, methods, and parameters accessible by the Polling Stations when generating transactions.

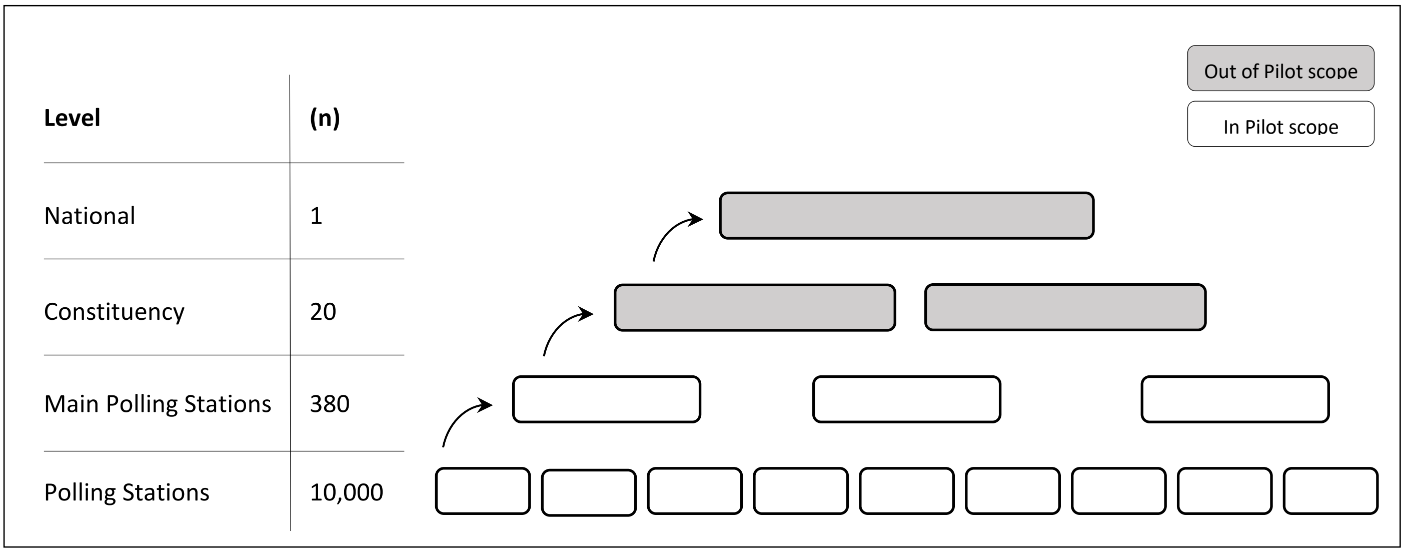
Among several characteristics of the **municipality.sol** contract, some play the most important roles in the project, as enumerated below:

- ❖ Aggregation of the results per Polling Station;
- ❖ Aggregation of the total sum of results of all the Polling Stations;

- Aggregation of the results per vote type;
- Aggregation of the total collected documents, such as polling cards, voter passes, power of attorneys, and objections;
- Trigger the end of voting session per Polling Station;
- Sign off the results by the Major;

These features are accomplished by a reference to the municipality smart contract from the polling station smart contract. So, every casted vote in a polling station also casts a vote in the municipality smart contract and therefore aggregating the results automatically.

When all polling stations finish casting their votes, the aggregated results will be promptly be available for the general population via the public page.



Level	**(n)**
National	1
Constituency	20
Main Polling Stations	380
Polling Stations	10,000

6.2 PS-BACKEND

Built on top of Web3JS, the most widely adopted API implementation of Ethereum’s Generic JSON RPC interface for Ethereum, the Backend Services are a set of interfaces between the Polling Station devices and the Ethereum blockchain nodes. These interfaces implement functions to write transactions based on the smart contract methods. The base communication is done through ExpressJS controllers. Entry points for secure HTTP requests (HTTPS) are accomplished via front-end services.

For this application, three controllers were implemented:

- Scan-controller: validate the documents provided by the voters;
- Verification-controller: register voter documents and match the total of admitted voters against total registered votes;
- Counting-controller: Provides the mechanism to cast the votes to the blockchain ledger and other auxiliary methods to count the votes.

As a means to an end, the controllers provide a seamless and secure layer between the user interface and the blockchain ledger.

6.2.1 SCAN-CONTROLLER

As stated, voter verification is required before voting. For that reason, scan-controller was created to validate the documents provided by the voters in exchange for a ballot paper. The scan-controller reads the Data Matrix code of the Polling Card, adds a salt, and then hashes it using the Keccak algorithm (its output format is natively supported by Ethereum) and ultimately stores it in the ledger. As a result, we know whether the card has already been used before letting the voter proceed to the private voting cabin. To properly recognize duplicate records, a private salt is used. This salt is destroyed after all scanning has been completed on all Polling Stations at the end of the day. All Data Matrix codes are hashed, which makes them irreversible and impossible to determine the original codes, whilst providing a validation mechanism to avoid duplicity of records (double-vote).

6.2.2 VERIFICATION-CONTROLLER

The verification process consists of two steps. The first step is to register voter documents (polling cards, power of attorneys, voter passes, and objections). The second step is to match the total of admitted voters against total registered votes.

The verification-controller is responsible for collecting the registered data to validate the amount of collected polling cards, voter passes, and power of attorneys, matching these sums with the total number of counted votes. The first step requires the entering of the total amount of collected voter documents on the Polling Station. The second step is validating whether the amount of collected documents matches the total amount of counted votes. The final step requires the Chairman to input the totals of the four different vote types, validating if they match the registered number of voters. If the validation is successful, nothing more needs to be done. If the validation is unsuccessful the system will request a recount of the votes. If there is any deviation, an explanation is requested from the Chairman.

6.2.3 COUNTING-CONTROLLER

To help on the counting process, a proper controller needed to be implemented. The solution collects each vote type and generates an individual transaction that represents it. After all the votes have been accounted for and the physical voting session has ended, the actual counting of the votes should begin. This controller is responsible for registering the actual votes called out by the Tellers and registered by the Chairman. Every single vote is registered as an individual transaction on the blockchain via the PollingStation.sol smart contract.

Using the chairman’s private key and blockchain address, the voting transactions are cast to the blockchain ledger. Guaranteeing the legitimacy of the transactions.

6.3 PS-FRONTEND

The front-end services are the main user interface tools that enable the Polling Station staff to interact with the ledger through the Backend Services via Secure Sockets Layer using Asynchronous Javascript and Xml (AJAX) requests.

SIGNING IN AND OFF

This interaction is transparent to the user, being presented in forms, panels, alerts, and confirmations. All critical interactions (signing in and signing off) with the front-end services will require the authentication of each user, requiring them to provide their credentials using two-factor authentication with a randomly generated code sent to the user's registered email address to verify the member's identity. The identities of the users are uniquely derived from their credentials that derives a public address on the blockchain and ensures only authorized members can change the data of these contracts. All users therefore must have their email addresses recorded on the ledger. On the first access attempt, the system sends an activation code that is used to create a password. With the email and password, the system generates a blockchain wallet that is needed to issue transactions.

SCANNING

The process for scanning starts when the Chairman chooses the *scan* option on the dashboard. The device camera is activated to capture the polling card data matrix image. The frontend then sends the code to the backend services to be hashed. Once the hash is done, a transaction is created and sent to the ledger. During the *scan* process, the screen allows the Chairman to verify if the polling card has not been used before. Other possible options are available on the right bottom corner of the screen, allowing the Chairman to register a private power of attorney, written power of attorney, voter pass, and objection. The private power of attorney is supposed to be scanned since it also has a data matrix code, whilst the voter pass and the objection options just record random numbers based on the current timestamp as a seed[31]. This guarantees the lack of collision on the resulting hash and therefore statistically eliminating the risk of incorrectly invalidating data matrix codes of other voters whilst maintaining a single point and logic for voter registration.

VERIFICATION

At the end of the day, the Chairman chooses the option *verification* from the dashboard menu. The system will then request the number of polling cards, voter passes, and power of attorneys. Once the numbers are confirmed, the application will be redirected to the counting screen.

COUNTING

The counting screen was designed with a swipe functionality to facilitate the counting votes process. That decision was made to avoid confirmations, reduce the risks of errors during the counting process, and give the Chairman feedback on the counting operation. There is a card in the centre which can be

[31] https://en.wikipedia.org/wiki/Random_seed

dragged to one of the four corners of the screen. When the card is dropped, a message is displayed as feedback of the operation. When all the ballots are counted, the Chairman chooses to continue to the next screen.

MATCHING

In the next screen, the Chairman is asked to enter the totals of the four different vote types. After clicking *verify,* the Chairman is presented with the result of the match between the total amount of counted votes and the totals of the four different vote types. The screen will give a visual clue that tells the Chairman if the numbers match. When they match, the system will allow him to continue to sign the results off. If they do not match, the system will request a recount based on the possible failures. The results only reveal if there is a match or not, it does not reveal the actual numbers to avoid bias on the recount process.

If there are no errors on the counting, the *sign off* screen will require that all the Tellers sign the results off. If there is any deviation between the counted votes and the number of registered voters, a text field will be shown to request an explanation for the deviation. Once all Tellers signed off the results the Chairman will also need to sign the results. The process is secured via password verification. Finally, the results are submitted to the ledger, and a confirmation screen is displayed.

7 CONCLUSION

As stated in the introduction, voting is an essential act for the functioning of a democratic state. To safeguard trusted and transparent elections, nations worldwide have tried various methods using both hardware (electronic voting) and software (support for counting and aggregating votes). Despite these efforts, elections are still vulnerable for errors and fraud.

To make elections less vulnerable and more trusted by society, this Pilot has taken the software approach to improve the way how votes are counted, processed, and aggregated. This Pilot was based upon the hypothesis that when votes are registered directly on an immutable and transparent database, the trust that society has in democracy will increase. To test this hypothesis, first, it needed to be validated that DLT can assist in counting, processing, and aggregating votes.

From a technical perspective it can be concluded that DLT, and more precisely blockchain, can be of great value when counting, processing, and aggregating votes. The technology offers complete transparency and, given that all data is registered in an immutable and traceable way, it is one of the best technologies for society to verify what is happening during elections. From an integrity point of view blockchain guarantees that the process is executed according to predefined rules. Blockchain records all data unchangeably and performs calculations reliably via 'Smart Contracts'.

The Polling Station staff is enabled by the solution to count and process votes in a trusted and transparent way, which also creates the opportunity to digitally sign off their result, instead of doing this with pen on a physical document that needs to be transported to the main Polling Station of the municipality. The solution furthermore enables the digital sign-off for aggregated results by the appointed mayors. This makes the manual process of aggregation redundant and minimizes the changes for errors or fraud. As a result, the solution therefore presents itself as a promising, and more reliable, alternative for the current OSV counting software.

Reliability, however, is also based on the level of security. A blockchain network is only as secure as the number of participants. The Pilot ran on four nodes on the cloud and one node on each of the five tablets, making it vulnerable to an attack. A full-scale envisioned implementation however, with ten thousand Polling Stations acting as participating nodes, does have the level of security needed to count, process and aggregate votes using blockchain.

It is too early to conclude that using blockchain as the underlying technology to count, process and aggregate votes will increase the trust society has in democracy. The mechanisms of blockchain have proven to be valuable, but to validate the impact on society more research is needed. This research can only be carried out when testing the solution on a larger scale with more municipalities and in different types of elections.