Digital Voting Pass

One step towards the digitalization of the entire voting process

W.K. Meijer
D. Middendorp
J.M. Raes
R. Tubbing

SAMSUNG
Authorization
Voting pass succesfully spent, please hand out 1 voting ballot
Mr. Middendorp has right to:
1
Voting pass
TRANSACTION HISTORY
NEXT VOTER
EUROPESE UNIE
KONINKRIJK DER NEDERLANDEN
PASPOORT

TU Delft

Digital Voting Pass

One step towards the digitalization of the entire voting process

by

W.K. Meijer
D. Middendorp
J.M. Raes
R. Tubbing

To obtain the degree of Bachelor of Science
at the Delft University of Technology.

Students:	W.K. Meijer	
	D. Middendorp	
	J.M. Raes	
	R. Tubbing	
Project duration:	April 24, 2017 – July 3, 2017	
Supervisor:	Dr. J.A. Pouwelse	TU Delft
Client:	S. Hadri	Milvum
Project Coordinator:	O.W. Visser	TU Delft

Acknowledgement

We would first like to thank our project coach Dr. Johan Pouwelse of the department of Distributed Systems at Delft University of Technology. The meetings with Dr. Pouwelse always inspired us to keep exploring the blockchain possibilities and bring the quality of our thesis to a next level. He also was glad to communicate our results to the outside world and brought us in contact with several experts, which was valuable for us.

Secondly, we are grateful for the support and office space that was arranged by our client, the company Milvum. We also would like to acknowledge the municipality of Emmen and the ICTU foundation, who took the time to listen to us and give expert feedback on the proposed solution. Furthermore, we thank the Ministry of the Interior and Kingdom Relations for the set of specimen travel documents they provided.

At last, we want to acknowledge the authors of all the source code we have used from several open source projects. Thank you all for making this research possible.

Summary

The Dutch electoral voting system is designed in a way that does not fit in the current era of the digital transformation. Almost every procedure during the elections is done manually and change in policy is very difficult as it requires a change in legislation.

This thesis explores a more trustworthy, transparent and less expensive way of verifying the suffrage of a citizen. In order to accomplish this, a literature review is presented about the election process in other countries and new possibilities with state of the art technology are reviewed. Based on these investigations, a solution that would make the current voting pass superfluous is presented.

The presented solution in this thesis is built with blockchain and machine readable travel documents. Considering blockchain, this technology is a decentralized ledger of transactions, where every transaction is immutable and built following predefined rules. These transactions are visible and verifiable for anyone connected to the decentralized network. Due to this decentralized design, the network is resilient against DDoS attacks and has no single point of failure.

Machine readable travel documents are used for the authentication. Dutch travel documents (passports, identity cards and driver's licences) are designed according to the ICAO machine readable travel document standard, which also requires the presence of a biometric chip. This means that every citizen already has access to a PKI (public key infrastructure) key pair, which can be used to sign data.

The presented solution combines these technologies and is first of its kind to link a machine readable travel document to a blockchain. These new possibilities can help solving the identity problem for all kinds of organizations.

Contents

1 Introduction

This thesis is the result of a bachelor project under supervision of the department of Distributed Systems at Delft University of Technology. The project is done for a client company called Milvum, helping their vision of creating a fully digitalized voting solution. The goal of this project is to design a solution, supported by modern technology, which increases the trustworthiness of a part of the voting process and decreases the manual labor. In order to narrow the scope, the main focus is on the digitization of the voting pass, the document which is needed at the polling station to prove that the citizen is eligible to vote. To confirm the proposed solution, a proof of concept based on a fully open source ecosystem is built. We propose a system that makes use of blockchain technology as a tamper-proof database to ensure integrity, combined with the ePassport chip from Dutch travel document to correctly identify voters. The source code of the different parts can be publicly found on Github[1].

To get a better insight in the Dutch voting process, first some background information will be provided. In the Netherlands, there have been different types of voting processes over the past decades. From mechanical machines in the late ’60s and electronic machines in the late ’70s to digital voting computers since the early ’90s. These voting computers were non-transparent and therefore evoked a lot of resistance by skeptical parties [1]. This resistance and skepticism is understandable in this era of cyber-warfare. Due to the high pressure of skepticism and distrust, the parliament decided to return to the pencil and paper variant in 2009 [2].

With the return from digital to paper voting a lot of manual work and extra costs came along. Paper voting requires more effort from volunteers at the polling station, who need to tally the votes by hand till the early morning. As stated in an interview with a Dutch municipality in chapter 5, there is a strong call for any technology that can help making these elections easier to organize from a practical point of view. Next to the increase in labor that comes with the return to paper voting, this manual approach has two additional negative side effects. In the first place, during the last parliament elections written out by the Dutch government in 2017, several thousands of votes were lost or not processed correctly. In one municipality 7 600 votes went missing due to human error while entering the results [3]. The second effect concerns the paperwork which is required for letting Dutch people abroad vote. During the last elections the papers that were sent abroad did not arrive on time, making it impossible to vote for Dutch citizens abroad [4]. The issues mentioned above show that manual work will not always ensure a trustworthy election. According to a statement by the political government office, the disruptions in the last elections did not affect the final outcome [5]. However, the investigations are not finished at the time of writing. Besides, it cannot be guaranteed that such issues will not occur again in the future.

The types of mistakes stated above relate mainly to carelessness and are therefore not really an issue with digital solutions. However, digital voting solutions have their own risks, mainly because they are proved to be more vulnerable to fraud. The Dutch “We do not trust voting computers” foundation[2] did a security analysis of the voting computers used by 90% of the Dutch municipalities until 2008. Their results were quite embarrassing [1]. It showed for example that the physical lock system, which should protect the system from unauthorized usage, is a complete fallacy. Every key used across the country is identical and everyone is able to order those keys online in a webshop. Furthermore, the software used on those machines appeared to be even

[1]`https://github.com/digital-voting-pass`
[2]`http://wijvertrouwenstemcomputersniet.nl/English`

worse: the research proved that it is possible to modify the software of the voting computer by unscrewing it and replacing a memory component. The researchers managed to install a chess game on the voting computer and after that they succeeded in installing a "fraud" version of the voting system application. This fraudulent version was able to cast random extra votes to a fraudulent party, without anything suspicious happening.

1.1. Research statement

The purpose of this research is to investigate the possibilities and importance of technological advancements in elections. The aim is to prove that there are digital solutions which enhance the entire process while improving the level of trustworthiness of voting. To support this statement, the following research question is defined.

How can the paper voting pass in the Dutch voting system be replaced by a secure, trustworthy, and easy-to-use digital system?

This research question is split up into the following sub-questions:

- How does the current Dutch (paper) voting process work?
- What are existing digital voting systems and how do they work?
- How to keep track of the identity of the voter?
- How to ensure that an individual cannot vote more than once?
- How to ensure the process cannot be manipulated by external parties?
- How to allow proxy-voting?
- How can we make the process trustworthy to skeptics?

All of these questions are answered in the research report which can be found in Appendix C.

1.1.1. Methodology

The analysis of the research question will be based on a functional prototype of a digital voting pass. This prototype is built using existing libraries and systems to create a standalone voting system which is compatible with the cryptography used in the biometric chip of Dutch travel documents. Using the knowledge gained during the research phase, requirements for the system were created using the MoSCoW model, these can be found in Appendix D.

The proposed system consists of four parts:

Polling station app
An Android application to verify and redeem the suffrage on the blockchain using a machine readable travel document.

Voter helper app
An Android application to transfer the suffrage to someone else using two machine readable travel documents. Due to different priorities, this app is not realized within this project.

Blockchain
A blockchain implementation which is compatible with the elliptic curve in Dutch travel documents issued after March 9, 2014.

Blockchain utility
A simple command line utility to organize an election by importing a CSV-file containing all the public keys of the citizens eligible to vote.

For this project, an agile approach was chosen. We worked with sprints of two weeks, releasing a working prototype after each sprint. To test and develop the connection with the ePassport chip, a specimen set of a passport and an identity card is provided by the Ministry of the Interior and Kingdom Relations, which are fully compatible with Dutch travel documents issued after March 9, 2014.

1.2. Thesis structure

The chapters of this thesis are structured in a way that corresponds largely with the actual process in chronological order. In chapter 2, the problem that this research is working on is described and analyzed. The first sprint where the passport interaction was implemented and the blockchain was modified, is described in chapter 3. The second sprint where the implemented technologies are connected can be found in chapter 4. The last sprint, creating the full working prototype, is discussed in chapter 5. To measure the scalability and performance, multiple tests are performed of which the results can be found in chapter 6. chapter 7 describes how the code quality is maintained. To get a broader view of the project, the social impact is taken into account in chapter 8. Finally, this thesis concludes the research in chapter 9.

2
Problem statement

The introduction in chapter 1 gives an overview of the current voting system (and its fallacies) the Dutch government is struggling with. This chapter will describe the issues that need to be addressed before the actual solution can be implemented. It also proposes several directions for a solution based on the original problem description as provided by Milvum (Appendix A).

Due to new insights, found in the research phase, the proposed solution in this thesis differs from this description on two parts. First, the requirement for a security check of the smartphone was deemed very impractical and is not necessary because this thesis only focuses on digitalizing the voting pass. Besides this, the government can provide secure phones for polling stations. Furthermore the Python requirement for the blockchain was deemed not viable as it severely limits the choice of blockchain frameworks.

This chapter concludes with a overview of the system proposed in this thesis.

2.1. Carelessness and fraud

In the Netherlands, recent decades have seen the introduction and farewell of multiple paper and digital voting systems. They all had their pros and cons in terms of trustworthiness. As for most systems, they were not 100% airtight, which is after all extremely hard to achieve. However, great care should be taken in order to discover whether or not flaws in a voting system will affect the final outcome and if a specific system is acceptable in terms of ease of use.

In order to minimize fraud possibilities, it is important to improve the level of trustworthiness. The system should be attack-resilient and highly fault tolerant. In order to make the system fault-tolerant the system should be as simple as possible and great care should be taken while developing the system, chapter 7 elaborates on this. For better attack-resilience of the system, blockchain technology is used. This technology has become more prominent in the past years.

Satoshi Nakamoto built the first practical application of blockchain technology, a peer-to-peer digital cash system called Bitcoin. Since then, there have been a lot of experiments performed using this technology [6]. A blockchain is a decentralized ledger of transactions, where every transaction is immutable and built following predefined rules. These transactions are visible and verifiable for anyone connected to the decentralized network. This makes it well suited for applications where trust is a central subject. As of today, implementations with blockchain are studied in a wide range of fields, from electric vehicle charging stations to medical records and currencies [7–9].

2.2. Ease of use and efficiency

In the current system there are three element which could use improvement in the area of ease of use and efficiency. In the first place, it is the task of the Dutch municipalities to organize the elections. This task consists out of setting up the polling stations and making sure that every citizen eligible to vote, receives their voting pass. The logistical operation of delivering 13 million voting passes to every citizen eligible to vote is a great challenge. On one hand this distribution is highly dependent on the current postal infrastructure and on the other hand it is a costly and time-consuming process. Secondly, on election day the voter needs to bring two items to the polling station: a voting pass and an identity document. These documents are checked at the polling station by an official who is assigned by the municipality. The officials are trained to verify the authenticity of the voting pass by looking at several authenticity marks [10]. There is however a big flaw in this authentication process at the polling station. Anyone with access to a printer capable of creating those authenticity marks can reproduce the voting pass and vote more than once at different polling stations within their municipality.

Thirdly, after the polling station is closed, the ballot papers and voting passes are counted manually. One of the most cumbersome parts of this step is when the amount of ballots and amount of voting passes do not match. If this is the case, all the counting needs to be done over and over again until the numbers match. If the voting passes would not have to be counted at the end of the day the job of volunteers at the polling station will be more efficient and the time needed to determine the final outcome will be shorter.

The voting pass is the only a way to prove that an identity is eligible to vote. This data needs to be connected to the identity itself in order to hand out a ballot paper. There should however be a system for this which is more efficient and easier to use than the current system. A useful element for such a system could be the ePassport chip, with which Dutch travel documents are equipped since August 2006 [11]. These chips are designed according to the ICAO (International Civil Aviation Organization) standard of machine readable travel documents [12] and provides every citizen with technological advanced tools to prove their identity. This chip is compatible with widely adopted NFC (Near field communication) standards which comes with the benefit that it can be read by an Android smartphone with NFC.

One of these tools in this ePassport chip to ensure an identity, is the presence of an elliptic curve PKI (public key infrastructure) mechanism, which consists of a private and public key stored onto the chip. The elliptic curve that is used in ePassports from 2014 is brainpoolP320r1 [13]. The private key is generated inside the travel document and cannot be read from the outside [14]. This ensures that it is, with the current technology, impossible to make an exact clone of the ePassport chip. Contrary to the private key, the public key can be read directly from the chip. Furthermore, a signing function is present in the chip, which can be used to sign data with the embedded private key. This signature proofs that this travel document was used at the moment of signing. Building a new system based on this chip that would substitute the current voting pass, will contribute to a more efficient and easy to use system.

2.3. Willingness to vote

The turn-out rate of elections is sometimes not as high as politicians would like to see it. Therefore, it is important that the proposed system will motivate more people to vote. This motivation can be improved by changing the overall image of the elections using technology. The generation Z, with birth years ranging from the mid-1990s to early 2000s has been exposed to an unprecedented amount of technology in their upbringing [15]. This generation is currently reaching the voting eligible age. In a decade where it is possible to order a pizza by talking to artificial intelligence [16], it is hard to explain an old fashioned way of voting to a new generation. Visiting a polling station several blocks away to handle some paperwork does not comply with the workflow this generation is used to in the commercial world. The current system does not correspond with the current digital era and therefore a digitalization of this system would be more suitable for the new generation of voters. However, there does not seem to be any research available to confirm this effect on the willingness to vote.

2.4. Design criteria

This thesis focuses on the applicability of blockchain technology for the election process, more specifically on digitalization of the voting pass. The system should consist of an open and transparent process which cannot be altered by other parties. The fundamental idea is that the proposed solution should be more trustworthy, efficient and easier to use than the current one.

In order to achieve such a system, it should first and foremost be ensured that the system cannot be influenced by unauthorized people. Besides the requirements related to confidentiality, the voting process should be experienced as a smoother process that the current one and the new process must comply with or exceed the current standards in terms of appearance and safety.

2.5. Proposed solution

In Appendix C the research process performed to find a solution to the given problem is described. A system that is outlined in Figure 2.1 followed from this research phase. Initially, a person is registered for the digital voting pass system, if he chooses. If registered, this person will not receive the paper voting pass as normal. Instead, the government will send 1 digital voting token to the voter's blockchain address. On the day of the vote, just as normal, the voter goes to one of the polling stations in his area, taking only his passport with him. At the polling station, the worker checks the voters document's authenticity as normal, besides this the worker also scans the document using the app created during this project. During this scanning the blockchain is checked to see if the voter owns any voting tokens on the blockchain. If any voting tokens are found, a transaction is created and broadcasted sending these voting tokens back to the government. This causes the voting pass to be spent and the voter receives a ballot paper from the polling station official, using this he casts his vote just like before the digital voting pass system.

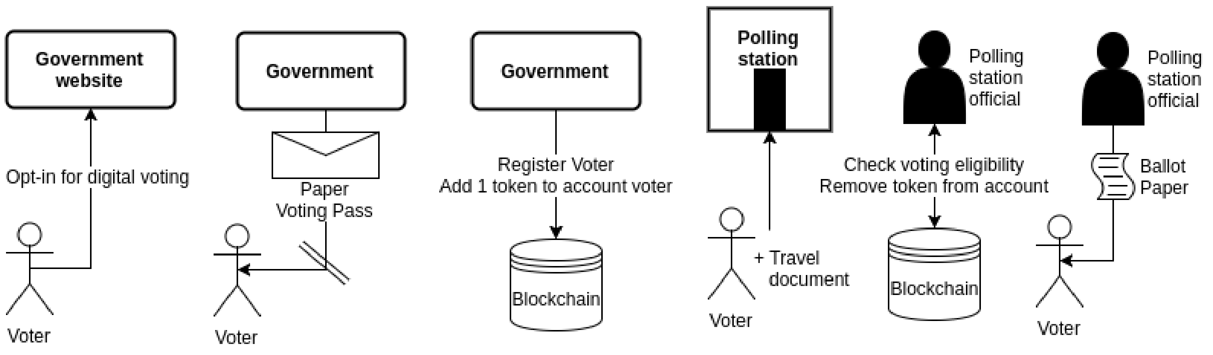


Figure 2.1: Schematic rendering of the proposed system

3 Blockchain implementation and ePassport connection

This chapter mainly describes the first sprint in this project. This sprint had a duration of only one and a half weeks instead of two. Two parts of the system were covered. Firstly, the creation of the app that can interact with the travel document. Secondly, the modifications to the blockchain to make it compatible with the travel document. The focus of this sprint was to make clear if the proposed solution would be viable. Getting a blockchain to work with same elliptic curve as travel documents from 2014 was a concern and had to be implemented in this first sprint, otherwise another (less preferred) blockchain implementation had to be chosen to ensure the viability of the project.

3.1. Blockchain implementation

A blockchain is essentially a distributed, decentralized database. Blockchains are transparent and incorruptible ledgers.[1] There are lots of blockchain implementations available. Each having their own strengths and weaknesses. In order to make a selection, many different projects were looked at and analyzed. An overview was created listing the important features and flaws of those implementations. These features and flaws were evaluated with the usecase of a digital voting pass in mind. This process is documented in the research report (Appendix C). Out of these options, MultiChain was selected as preferential choice and is discussed in the next section. The alternatives were kept as back-up options and are discussed further in subsection 3.1.2.

3.1.1. MultiChain

MultiChain[2] is a blockchain which is based on Bitcoin core, but the developers of MultiChain have introduced additional features. It contains support for asset transactions. This means that assets can be created on the blockchain that act just like the main currency and can be send between addresses. In this prototype assets are used to represent different elections and municipalities, since elections are often split between municipalities in the Netherlands. Besides this, inhabitants of one municipality cannot cast their vote in a second municipality. Having a voting pass balance for a particular asset ensures that a voter can only cast a vote in the corresponding municipality and thus for the election that he or she is allowed to cast a vote.

Furthermore, MultiChain includes a permission system on address level. Sending and receiving transactions are important permissions since every voter has an address which has one or more voting tokens. These represent if a voter is eligible to vote. Before an address can receive a token, it must have receive permissions. This way the central authority is in control who receives voting rights. Furthermore, an address is only able to send voting tokens, or claim its voting right, when it has send permissions.

The mining permission is valuable, since a third party could perform a majority attack (51% attack) [3] only if it can mine new blocks. Since only addresses with mining permissions can create new blocks and thus expand the chain, a monopoly attack is not possible.

[1]https://blockgeeks.com/guides/what-is-blockchain-technology/
[2]http://www.multichain.com/
[3]https://en.bitcoin.it/wiki/Majority_attack

Modifying the signing algorithm

MultiChain and most of the other blockchain technologies use the secp256k1 elliptic curve to verify and sign transactions. Since travel documents use the brainpoolP320r1 elliptic curve, a library that implements the brainpoolP320r1 elliptic curve is required. CryptoPP[4], a library of cryptographic schemes, has implemented this curve and has been added to MultiChain, as part of this project, to support brainpoolP320r1. From now on the modified version of MultiChain will be called ePassportChain.

The initial step to implement brainpoolP320r1 in ePassportChain was to find where the private and public keys are stored. After some searching in the code it was discovered that the private and public key are stored in the classes `CKey` and `CPubKey` respectively. These classes have been modified so MultiChain uses brainpoolP320r1 instead of secp256k1.

Originally, MultiChain stored the keys in the variable `vch`. In ePassportChain the variable `vch` contains the raw key (in both classes), while `privKey` and `pubKey` are the representation, in CryptoPP, of a private and public key respectively. The variable `vch` is still present in the implementation of ePassportChain since some functions require the raw keys as input or output. To be specific, in the class `CKey` the function `GetPrivKey` has as output a raw key (the type `CPrivKey` is a vector of unsigned chars with a secure allocator), while `SetPrivKey` and `Load` require as input a `CPrivKey`. In the class `CPubKey`, the functions `Serialize` and `Unserialize` have raw keys as their in- and output.

Because the signing function of a travel document is created for authentication of the document, there is no need for it to sign large byte arrays, therefore it can only sign 8 bytes of data. However, the hash of a transaction is a SHA256 hash, which is 32 bytes long. To tackle this problem, the hash is split up in four different parts and are signed individually. In Figure 3.1 a visualization of the signing by the passport is given, where the 32 byte block is the hash of a transaction.

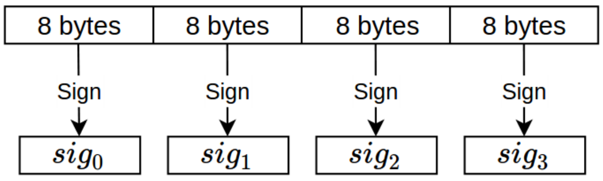


Figure 3.1: Signing a transaction with a travel document

As a consequence, ePassportChain must verify a transaction signed by a travel document, so the `Verify` function must be altered in such a way that it accepts four signatures. Initially, this caused some trouble since MultiChain implemented a maximum signature length of 255 bytes while one signature from a travel document is always 80 bytes long ($4 * 80 > 255$). The 255 byte limit has been altered to support signatures of up to 320 bytes. This makes is possible to store the signatures together in a byte array, as is visualized in Figure 3.2. So for example, the signature of the second part of a hash, $\texttt{sig}_1$, is stored at index 80 until 160 of the byte array containing all signatures. The verify function now verifies each $\texttt{sig}_i$ and if one of the signatures is incorrect the entire transaction is rejected.

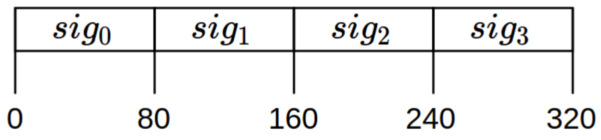


Figure 3.2: Signature storage

The new problem is that mined blocks create one signature of the hash, while the verification expects four signatures. To address this issue, two options were considered:

- Choose the appropriate verify function based on whether a block or transaction is verified
- Sign the mined blocks in the same way as transactions, using four signatures

The second option was chosen, since it is easier to implement and due to the struggles with MultiChain. The block-hash, like transactions, is split up in four parts and signed individually. These signatures are then stored as in Figure 3.2 so the verify function accepts these signatures.

[4]`https://www.cryptopp.com/`

Improving the signing algorithm

During the research phase of this project, it did not seem safe to sign the hash of a transaction in parts. In order to improve the level of security another method was considered. This problem of limited signing and encryption functions is not an unknown one and is also relevant in other fields of cryptography, for example the 3DES encryption is only able to encrypt blocks of 8 bytes.

The first signing strategy, as discussed above, is known as ECB (Electronic Code Book), named after the analog lookup books for encrypting and decrypting texts [17]. With this method, every single block of 8 bytes is encrypted and decrypted separately. Because 3DES is deterministic, which means that encrypting the same data with the same key results in the same cyphertext, blocks which contain the same data will also have the same cyphertext. Due to this, it becomes easy to find patterns in the encrypted data. Furthermore, this also makes it possible for an attacker to combine different blocks and create a complete new signed/encrypted message by using blocks from other encrypted messages. This is not that big of a problem, because the only thing that will be signed in this prototype is a SHA256 hash. A part of a hash does not represent a particular part of the transaction, so this will not be vulnerable to replay attacks [18].

The second other method is called CBC (Cipher Block Chaining). This method consists of using previous blocks to sign or encrypt the upcoming blocks. This makes it more difficult to see patterns in the data which seems to be a clever idea, because it isn't even possible to determine the signature of a block if the signature of the previous block is unknown. Unfortunately, this method uses the previous signature to perform the binary XOR operation on the following block. The signatures created by a Dutch travel document are signed using ECDSA. Due to the possible leak of the private key, these signature are not deterministic which means that signing a block with the same travel document multiple times, results in a different signature every time. So encrypting an decrypting the next block using XOR will not work.

Because of this limitation of the non-deterministic signing function, CBC cannot be implemented in the algorithm. The original plan to implement the ECB algorithm had been followed.

Public key to blockchain address

To ensure that a machine readable travel document can act as a wallet, the public key of the document needs to be translated to an address on the blockchain. MultiChain uses an address system similar to the one used in Bitcoin [19]. The small differences consist of the addition of an identifier generated by the blockchain instance itself. This value is hashed into the final address. Due to this, it is impossible to use an address which is intended for a different blockchain [20].

The steps leading to a valid address are well documented in the developer reference of MultiChain. This ensured a smooth integration of this process which led to a method implementation to translate the public key of a passport to a valid address on the blockchain.

Distributing voting tokens

For every election, there needs a new batch of voting tokens to be sent to the citizens. For this first prototype, a simple Python command line script was built. This script follows the steps below:

1. Load CSV containing public keys
2. Convert public keys to valid ePassportChain addresses
3. Create new assets, with the number of coins corresponding to the amount of addresses
4. Grant send and receive permissions to the generated addresses
5. Distribute the assets to the generated addresses

This script can be found in the `e-passport-chain-util` repository[5] on GitHub.

3.1.2. Alternatives

Due to minor struggles and relatively slow progress initially during the implementation of the brainpoolP320r1 elliptic curve in MultiChain, other blockchain solutions were also considered during this sprint. In the following section they are discussed in more detail.

[5] `https://github.com/digital-voting-pass/e-passport-chain-util`

The first considered blockchain alternative is Dragonchain. Dragonchain is a blockchain platform which is actively developed by the Walt Disney company. It is written in Python which makes the code very accessible. The structure of a transaction relies on an embedded public key in PEM format. This means that the curve parameters are shipped with every transaction. Due to this, it does not matter which Elliptic Curve is used for the signing of a transaction. It was quite straightforward to come up with a proof of concept by modifying the unit test to a brainpoolP320r1 curve.

Unfortunately, Dragonchain is more some sort of blockchain platform, which only has support for raw transactions and Python based smart contracts. There does not seem to be any implementation of a wallet or any other way to store any value in the blockchain, just pure signed transactions.

Due to this, and the lack of documentation, this platform does not seem to be suitable for the purpose of implementing a digital voting pass.

Another possible blockchain is Openchain, which is not setup in a decentralized way. Because of this, there can only be one validation node, as discussed in Appendix C. The code is written in C# which makes it quite accessible. The implementation uses the Bouncycastle library[6] for signing and validation of transaction signatures. Due to this, it was relatively easy to transform the codebase to brainpoolP320r1. This transformation consisted out of changing the curve properties which are sent to Bouncycastle.

Different experiments turned out that is is fairly easy to issue a new assets as an admin and distribute it to an address. It was also quite simple to transform the public key of a passport to a valid Openchain address.

Further investigation of this solution was stopped because significant process was booked with the first-choice blockchain solution (Multichain).

3.2. Coupling android app and passport

For both the polling station app and the voter helper app a connection with the passport needs to be made. This connection needs to accomplish two things:

- Read the Active Authentication (AA) public key (located in data-group 15 [21])
- Sign a blockchain transaction using the private key located in the passport

The AA public key serves as an identifier for the account of the voter on the blockchain. For every interaction (e.g. sending tokens, retrieving current balance) with the account of a voter, its public key is needed. To ensure that a transaction is performed with the authorization of the voter, the transaction needs to be signed with the voters' travel document.

The signing is done by making use of the AA protocol. This protocol serves as a verification of the uniqueness of the travel document. It accepts an 8-byte input and signs this with the private key of the travel document located in a non-readable part of the chip. The travel document returns a byte array containing the signature. It can be verified that this byte array was signed using the travel document belonging to a certain public key, confirming the correct person claimed their suffrage.

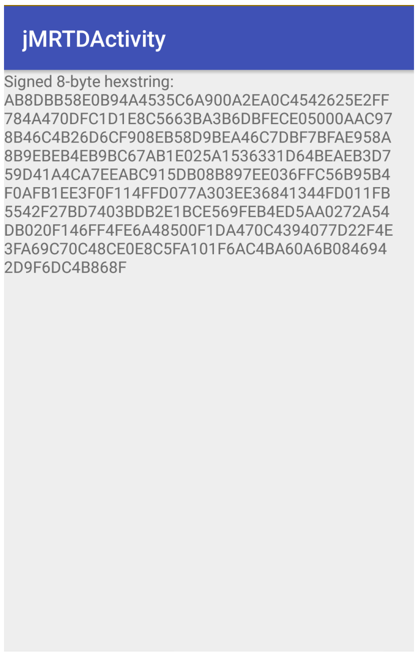
For creating the connection with the passport the JMRTD library[7] is used. JMRTD is an open-source java library that implements the MRTD (Machine Readable Travel Documents) standards as defined by the International Civil Aviation Organization (ICAO). The use of this library makes the connection in an Android app much easier since there is no need for a custom built interpretation layer between two programming languages.

In the Google Play Store a few apps can be found that make a connection with the passport and retrieve information from it. These apps give a clear example of a working concept. A well-known app is ReadID[8] which also makes use of the JMRTD library, this app unfortunately is not open source. As a starting point the epassportreader app by github user Glamdring[9] was used, which has a very basic implementation of the JMRTD library.

The activity that reads the data from the passport waits until a NFC chip is detected, if this NFC chip belongs to a passport it tries to make a connection.

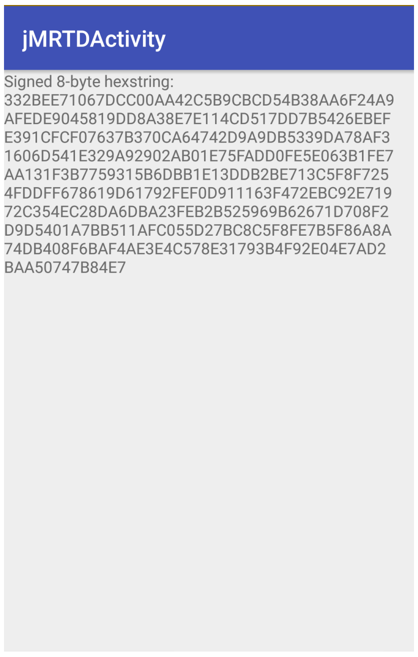
For this connection three pieces of information are needed: the document number, date of birth and the expiry data of the document. This is needed for the Basic Access Control (BAC), which ensures a person trying to read the passport also has physical access to it. This information can be read from the Machine Readable

[6]https://www.bouncycastle.org/
[7]http://jmrtd.org/
[8]https://www.readid.com/
[9]https://github.com/Glamdring/epassport-reader



```
Signed 8-byte hexstring:
AB8DBB58E0B94A4535C6A900A2EA0C4542625E2FF
784A470DFC1D1E8C5663BA3B6DBFECE05000AAC97
8B46C4B26D6CF908EB58D9BEA46C7DBF7BFAE958A
8B9EBEB4EB9BC67AB1E025A1536331D64BEAEB3D7
59D41A4CA7EEABC915DB08B897EE036FFC56B95B4
F0AFB1EE3F0F114FFD077A303EE36841344FD011FB
5542F27BD7403BDB2E1BCE569FEB4ED5AA0272A54
DB020F146FF4FE6A48500F1DA470C4394077D22F4E
3FA69C70C48CE0E8C5FA101F6AC4BA60A6B084694
2D9F6DC4B868F
```

(a) Output 1



```
Signed 8-byte hexstring:
332BEE71067DCC00AA42C5B9CBCD54B38AA6F24A9
AFEDE9045819DD8A38E7E114CD517DD7B5426EBEF
E391CFCF07637B370CA64742D9A9DB5339DA78AF3
1606D541E329A92902AB01E75FADD0FE5E063B1FE7
AA131F3B7759315B6DBB1E13DDB2BE713C5F8F725
4FDDFF678619D61792FEF0D911163F472EBC92E719
72C354EC28DA6DBA23FEB2B525969B62671D708F2
D9D5401A7BB511AFC055D27BC8C5F8FE7B5F86A8A
74DB408F6BAF4AE3E4C578E31793B4F92E04E7AD2
BAA50747B84E7
```

(b) Output 2

Figure 3.3: Results of signing the hex string '0a1b3c4d5e6faabb' with a passport

Zone on the passport by using optical character recognition (OCR) as explained in section 3.3 or can be filled in manually when OCR fails or if OCR is not preferred by the user. Once the BAC succeeds, a connection is established and the contents of the chip are unlocked and can be read.

The app mainly uses the following two functions:

- `getAAPublicKey()` - which returns the public key located in datagroup 15
- `signData()` - which accepts an array of 8 bytes as input and returns a byte array signed by the passport

These two functions are essential for signing transactions and getting information from the blockchain associated with the voter. In Figure 3.3 the resulting byte array, represented as a hex string, is shown. Since the signing algorithm of the passport is non-deterministic, the same input will yield different outputs every time. The implementation of this functionality is described in chapter 4 and chapter 5.

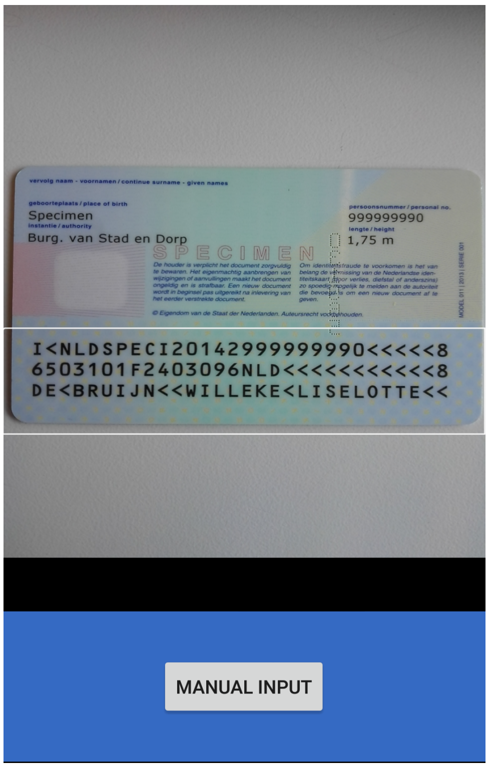
3.3. Optical character recognition

In order to communicate with an ePassport from an Android device, some data needs to be read optically from the document and supplied digitally for authentication.

There are a few options to choose from when it comes to performing OCR on an Android device. Two popular libraries that were considered are the Mobile Vision API [22], provided by Google, and Tesseract [23], which is also maintained mostly by Google. Tesseract has a history of being an accurate OCR solution, and it has an Android implementation called Tesseract Tools [24].

We chose to implement Tesseract and not Android Vision because first of all, looking at the vision example project, there was little support as issues were not being addressed. Next to this, after some testing it seemed that the Android Vision library is much less accurate than Tesseract. Also, Mobile Vision has no support for putting constraints on the detected data. Some people online seemed to switch to Tesseract after not reaching their goal with the Vision API. Therefore it was decided to go with Tesseract straight away, and more specifically, with tess-two [10], a fork of Tesseract Tools which is expanded and maintained. Because of the API 21+ requirement, (see requirement 8 in Appendix D), only supporting the improved "android.hardware.camera2" camera API is required, these classes were introduced by Google in API 21 [25]. To make implementation easier and not reinvent the wheel, a sample app made by Google was used to get the code related to using the camera2 API [26].

[10] `https://github.com/rmtheis/tess-two`


vervolg naam - voornamen / continue surname - given names

geboorteplaats / place of birth
Specimen
instantie / authority
Burg. van Stad en Dorp

persoonsnummer / personal no.
999999990
lengte / height
1,75 m

SPECIMEN

De houder is verplicht het document zorgvuldig te bewaren. Het eigenmachtig aanbrengen van wijzigingen of aanvullingen maakt het document ongeldig en is strafbaar. Een nieuw document wordt in beginsel pas uitgereikt na inlevering van het eerder verstrekte document.

Om identiteitsfraude te voorkomen is het van belang de vermissing van de Nederlandse identiteitskaart (door verlies, diefstal of anderszins) zo spoedig mogelijk te melden aan de autoriteit die bevoegd is om een nieuw document af te geven.

© Eigendom van de Staat der Nederlanden. Auteursrecht voorbehouden.

MODEL 011 | 2013 | SERIE 001

```
I<NLDSPECI20142999999990<<<<<8
6503101F2403096NLD<<<<<<<<<<<8
DE<BRUIJN<<WILLEKE<LISELOTTE<<
```

Figure 3.4: Scanning of MRZ using the OCR in app, early version

During the development of the OCR feature, it was discovered that the speed of detection and the ability to run multiple scanning threads simultaneously is greatly dependent on the kind of phone used. For example, the Samsung Galaxy S7 and Google Pixel that were used for testing had no difficulty running at least four threads and is able to scan a passport or ID card almost instantly. However, scanning on a LG G4 and a Galaxy S3, being older phones, was more difficult. Those phones started visually lagging when multiple scanning threads were started. Of course optimization of the OCR feature is still possible, which could improve detection speed and accuracy.

Surprisingly, the accuracy of the OCR seemed to be greatly dependent on the used document. One passport's MRZ might be scanned instantly while scanning another might require a lot of effort. This may be because of a different contrast on the older passports or because of the particular order of letters in the MRZ. In particular, characters like 0, *O* and *D* or 8 and *B* are often wrongly interpreted. It is possible to write a piece of code that swaps known wrongly-interpreted characters, but this would involve checking many combinations as the list of possibly wrongly interpreted characters grows. In another attempt to increase the accuracy, some image processing was done. Attempts included changing the contrast, filtering colors, and converting to monochrome bitmap. This did not seem to have any significant, if any at all, effect on accuracy. This result was partly expected since Tesseract does image processing by itself using the Leptonica library [27].

Tesseract uses training data when doing its OCR scanning. This file contains information about the shapes of the letters used. This means that improved accuracy and speed can be achieved using a trained data file fine tuned to the ePassport's MRZ. Online [28] a specialized trained data file for MRZ was be found. Unfortunately this file caused crashes when performing the scan, likely due to a mismatch in versions the file was created for, and the version used in the application. This was encouraging to do some more research into creating test data for Tesseract, which led to a way to convert a TrueType-Font file to a trained data file for Tesseract [29].

To ensure that the information obtained from the OCR scan is correct, the MRZ provides several check digits [30–32]. A method was implemented to check these digits to know when the MRZ was interpreted correctly.

4 Connecting with ePassportChain

During the second sprint a clickable demo of the app's UX (User experience) and a technical demo consisting of the connection between the blockchain and the app was developed. Furthermore, testing was set-up for the various parts of the application. This chapter gives an overview of the work that was done and the problems that were encountered.

4.1. Retrieving the digital voting pass

The polling station app needs to check the balance of votes spendable by the owner of the travel document. In order to know this balance the transactions need to be inspected to calculate the final balance. Clearly a connection to the blockchain is necessary. There are four ways of connecting to the blockchain. In this section, the advantages and disadvantages of these options will be taken into consideration.

The first option is to use a JSON RPC interface. This option is the easiest solution, because the app does not need to know anything about the blockchain. Downloading all the blocks will not be necessary and all the logic happens at the server, only the signing of the transaction occurs in the app. Due to this, the system relies on one working web service, which is a single point of failure.

The standard RPC service which is shipped with MultiChain is also not designed to act in this way. For example, it relies on its own wallet which is attached to the node. So, before this RPC service can be used, MultiChain has to be modified and some authentication rules have to implemented.

The second option is to implement a so called "light wallet". In this setup, the wallet functionality is embedded into the app and the app is directly connected to the nodes in the network. The download of the entire blockchain can be a time and data consuming process. To fix this, light wallets only rely on the headers of every block with only the transactions related to the wallet included. The headers of a block can be used to verify the transactions in a block, without the need of the details of every transaction.

The main problem with this approach is that the headers are not enough to calculate the balance of voting tokens of every citizen. So, an interface needs to be implemented which responds to requests for balances associated with any given address. This is problematic as these requests cost time. Also these requests would need to be answered by nodes, meaning the functionality needs to be build in the node, because introducing a central server to answer these requests would also introduce a central point of failure.

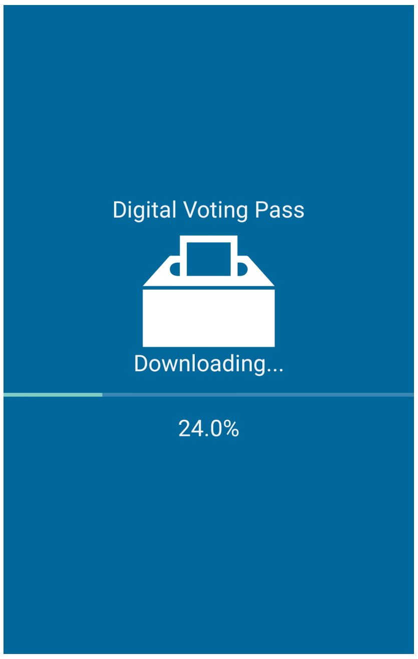
A third option is implementing a full wallet in the polling station app. In contrast to the light wallet, a full wallet downloads the full blockchain with all blocks and transactions. Because all transactions in this case are stored locally, balances of voters can be easily calculated on the spot within a short time frame. However, initial synchronization will be slow as all this data needs to be downloaded. Storage space requirements will also be higher than in the other proposed solutions.

To have the best of both worlds (light wallet and JSON RPC), the following approach might work. If the app doesn't have a local blockchain at all, but is connected to a node. Then it is still able to broadcast a transaction to the network. A rejected transaction is sent back to the sender if it is invalid.

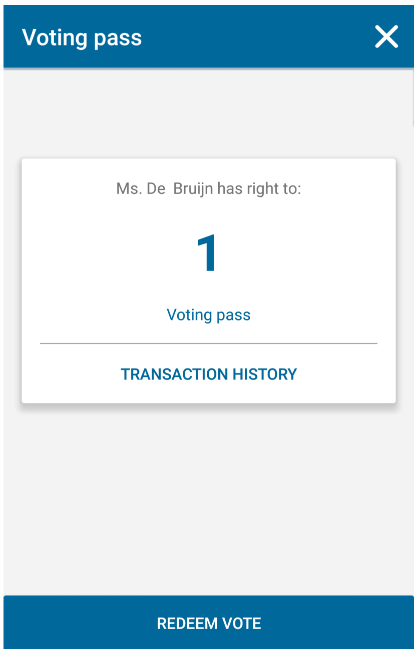
It is possible to cast three transactions. The first transaction tries to claim three voting tokens, if this fails the second transaction is broadcast. This transactions claims two voting tokens. When this transaction is rejected the last transaction is broadcast which claims one voting token. This way there is no need for a central server to return the balance of users.

The last proposed solution might seem to be the most obvious choice and was initially chosen for the development of the wallet in the application. However, during the implementation, several new insights forced us to take a step back and rethink the proposed solutions.

Transactions on the blockchain are based on so called UTXO's (unspent transaction outputs), so every regular transaction needs at least one UTXO in order to spend an asset. These UTXO's can only be queried from the blockchain, which makes it impossible to build a transaction without any knowledge of the blocks. Because of these new insights, the decision about the type of wallet used in the application had to be changed to the full wallet implementation. Which means that the application need to synchronize with the blockchain at every start, as showed in Figure 4.1a. This connection is implemented in the polling station app using a modification of the bitcoinj library[1]. The performance of this approach will be discussed in more detail in chapter 6.



(a) Downloading of blockchain on device



(b) Retrieved voting pass balance

Figure 4.1: App acts as a full node, syncing the full blockchain

4.2. User experience design

As stated in Appendix B, the client offered a budget to use an external party to create a UX design for the application. At the end of the first sprint, a professional UX designer was hired at Milvum. To make the purpose of the first application clear to the designer, a wireframe was sketched on paper. During the second sprint, there were several meetings set up with this professional.

The main requirements for the UX design are that the user should be in control, it should be clear what is happening and the app needs to feel trustworthy. The first two requirements can be accomplished by providing clear descriptive text and images indicating what is happening, or what will happen when the user performs an action. This can for example be seen in the bright green or red colors depending on whether redeeming the voting pass succeeded or failed. The latter requirement is a bit more abstract. To accomplish this the same colors as the Dutch government are used to give forth a sense of authority and security. Furthermore trust can simply be created by having a UX that makes the system looks finished and well thought through. This implies

[1]`https://bitcoinj.github.io`

that other parts of the system have been developed with similar care. With these requirements in mind the UX designer started on his first prototype.

The first prototype was based on a sketch drawn with pencil and paper, to make clear which screens were needed. This emerged to a basic colorized workflow, visible in Figure E.1 (Appendix E).

While discussing this design, there still were some sub-optimal things which resulted in new insights. There was for example no way to select the current elections for example. Also, the amount of ballot papers that should be handed was not very obvious and the process of scanning the MRZ isn't really a process which should be visible in a process bar. Furthermore, the design was not fully compliant to the Android Material Design [33] style guide as we preferred. The style guide states that the App Bar should be used for branding, navigation, search, and actions, not for notifications. The feedback above resulted in a second iteration where these issues are improved. This result is visible in Figure E.2.

Besides design choices made by the designer, we made some of our own. One of these is about the status bar, we decided to make it transparent according to material design guidelines. We added a top-padding to the action bar and enabled the transparent status bar. This way the color of the action bar will shine trough the status bar giving it the same kind of color but somewhat darker. In the camera view the transparent status bar causes the camera feed to shine through he status bar, which is enhances the visual appeal.

4.3. App permissions

The polling station app needs several permissions in order to perform its basic functions. These include:

- Camera permissions to scan the MRZ
- NFC permissions to communicate with the document
- Internet permissions to communicate with the blockchain
- Storage permissions to store and read both the blockchain data and trained data for the OCR scanning library

Internet and NFC permission are not considered dangerous permissions by Android [34], so these permissions are granted when installing the app and we can use them straight away. The others, storage access and camera access, are considered dangerous permissions. This means that starting from API level 23 these permissions needs to be granted at runtime when the app requires them. For the camera permission, this occurs when the OCR scanner is opened after the user clicks the button to scan a document. The need for camera permissions in this case is quite clear to the user. For storage permissions the situation is different however, these permissions are needed right when the app is launched in order to store the downloaded blockchain data. The need for this permission is not very clear to the user and may be confusing when the app asks to write to the storage right after launch. For this reason an error dialog was included in the app. When the app is denied one of the dangerous permissions, it will display a dialog explaining why the permission is needed.

4.4. Blockchain permissions analysis

MultiChain has implemented a permission system, which enables managing permissions at address level. These permissions are thus also available in ePassportChain. The following permissions are currently available [35]:

- `connect` – to connect to other nodes and see the blockchain's contents
- `send` – to send funds, i.e. sign transactions
- `receive` – to receive funds, i.e. appear in the outputs of transactions
- `issue` – to issue assets, i.e. sign transactions which create new native assets
- `create` – to create streams, i.e. sign transactions which create new streams
- `mine` – to mine blocks, i.e. to sign the metadata of coinbase transactions
- `activate` – to change connect, send and receive permissions for other users, i.e. sign transactions which change those permissions

- `admin` – to change all permissions for other users, including issue, mine, activate and admin

As will be explained in section 4.5, anyone is able to connect in order to create a transparent network. This poses a minimal risk to the network since normal nodes can only see what is happening in the blockchain. They cannot alter the blockchain in any way.

If an address has the `send` permission, it is allowed to send funds (i.e. a voting token) to another address. In order to redeem a voting pass, an address needs to send a transaction, so this permission is required for all voters. Still the parameter `anyone-can-send` is set to false. This is because the government needs to be able to revoke a person's voting right, for example in the case that this person dies after the voting passes were sent, but before the election day.

However, it is recommended that multisignature transactions are enabled in future releases, as will be discussed in section 9.1. Multisignature transactions in an one of two setup, where the government is one of the two key owners, allows the government to spend the voting tokens when a person's voting right needs to be taken away.

When an an asset or stream is created, a transaction is send to the blockchain [36]. This means that rogue nodes can flood the network by, for example, creating many assets. To prevent this only government nodes should have permissions to `issue` and `create`. However, the government nodes can be hacked and hackers can flood the network. This can be prevented by revoking these rights of these nodes after the vote tokens are distributed. A disadvantage of this solutions is that it is not possible to reuse the blockchain for a future election.

The mining right allows a node to generate a block. If there is monopoly of rogue nodes with mining permissions it is possible that fraud is committed. Of no one has the mining right no blocks will be generated and no transactions will be verified. So, there needs to be at least one mining node and this node should be owned by the government. Again, this mining node can be hacked, which is why it is desired to have several mining nodes owned by the government. This way the non-hacked nodes can reject a mined block from a rogue node.

4.5. Blockchain parameters

With ePassportChain it is possible to alter parameters of the chain. The parameters must be set before the initial run of a chain, after the initial run it is not possible to alter the parameters.

In this section the most important parameters, values of these parameters and a justification of these values will be discussed. Since there are more than 50 parameters, a complete list of the parameters, values and explanations can be found in Appendix F.

4.5.1. Permissions

The two parameters that will be discussed in this section are `anyone-can-connect` and `anyone-can-mine`, which are permission parameters. These parameters control if any node can connect to the blockchain and if any node can mine a block. Parameter `anyone-can-connect` is set to `true`, so any person can see what is going on in the blockchain and increase the trust of a person. The second parameter, `anyone-can-mine`, is set to false. This is done so only government nodes can mine a block. This ensures that a majority attack is not possible because only nodes with permissions, which are run by a trusted party, can generate new blocks.

4.5.2. Blocks

A transaction is confirmed when it is contained in a block, so in order to verify an transaction swiftly, a block should also be generated swiftly. ePassportChain has two parameters which can influence the block time: `target-block-time` and `pow-minimum-bits`. The parameter `target-block-time` which is the target average time in seconds between two consecutive blocks. The other parameter, `pow-minimum-bits`, is the proof-of-work difficulty which must be a value between 1 and 32. The first being the most easy and the latter being the most difficult.

If it is assumed that there are 13 million people who cast their vote on a single day [37], and it is possible to cast a vote from 6am to 9pm, a calculation shows that there are approximately 240 transactions per second ($13\,000\,000/15/60/60 \approx 240$). This is the average , but it is desired to handle at least double the average to account for peak moments. Furthermore, a transaction is about 200 bytes and blocks are at most one megabyte. Then, about 5 000 transactions fit in a block. From this it is possible to calculate the required target block time: $5\,000/480 \approx 10.4$. The `target-block-time` is set to ten seconds since only natural numbers are allowed.

To make sure that a miner is indeed able to mine a block in ten seconds, the parameters `pow-minimum-bits` must be low enough. After some experimentation it is concluded that a proof-of-work difficulty of four is suitable to mine blocks in ten seconds.

5 Redeeming the suffrage

The last iteration of the development process focuses on finishing the full prototype. This prototype shows the ability to redeem the voting token from the blockchain by signing transactions using a travel document and a mobile phone. Furthermore, this iteration was used to evaluate the functionality of the prototype by receiving expert feedback.

5.1. ePassportChain compatibility with travel document

The world of cryptography is a tangle of different formats and systems. As stated in chapter 3, the entire blockchain depends on an elliptic curve which must be compatible with the travel document. Even with the right curve, there are other specifications that need to be satisfied, e.g. the amount of storage for the public key and signature. A public key and a signature are basically stored as two very large numbers, which can be encoded in several ways. These numbers can for example be stored in a hexadecimal string or in a serialization structure like DER [38].

Because of the compact size, the hex encoded string method is widely used over blockchain technologies, which is also the case in ePassportChain. Public keys in this format can be compressed or uncompressed. Compressed keys are used to keep the transactions small. For the convenience of this research, only uncompressed keys are used.

However, during several experiments it is found that the public key of the travel document is stored in DER format. This format is based on the ASN.1 serialization structure and contains other information about the public key, such as the curve parameters. The last 80 bytes of this serialization represent the public key, these 80 bytes consists of two elliptic curve points of 40 bytes each. In the initial prototype of ePassportChain, the keys are encoded as an uncompressed hex encoded string. The initial prototype is thus not compatible with a travel document. Fortunately, this could easily be changed since CryptoPP is capable of dealing with elliptic curve points. After these changes ePassportChain expects an 81 bytes long public key, the first byte is 0x04 and the last 80 bytes contain the two elliptic curve points. The first byte informs ePassportChain that this is an uncompressed key. Although solely uncompressed keys are currently used, future implementations can benefit from this.

Due to the fact that these are all bytewise operations to verify the signature of a public key in ePassportChain, it is really hard to support multiple elliptic curves. A different curve also results in a different key length. For this reason, this research is limited to the last Dutch travel document standard, issued from March 9th, 2014.

5.1.1. Building an ePassportChain transaction

The app uses the Bitcoinj[1] library to keep track of the blockchain. This library is originally developed for the Bitcoin cryptocurrency, however MultiChain is a fork of Bitcoin and because of this, it is quite easily adoptable in the project.

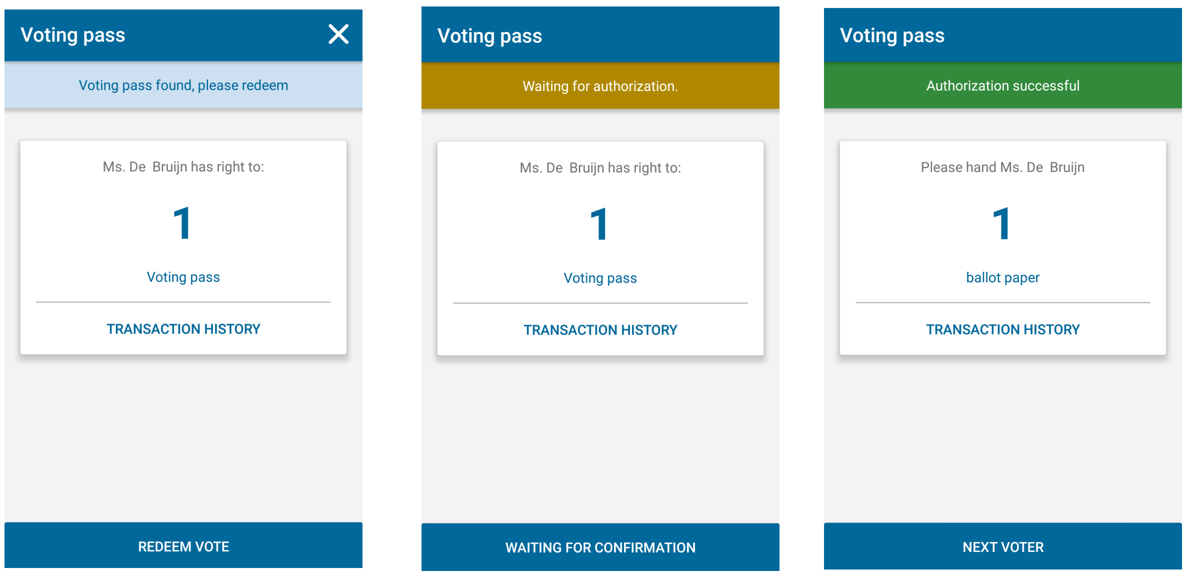
Because of the big modifications on the blockchain as shown in chapter 3, Bitcoin has to be modified to make it work with ePassportChain. One of these modifications was altering the signing method and making it

[1] `https://bitcoinj.github.io/`

talk to the passport. In order to achieve this, the `LocalTransactionSigner` class was extended to `Passport-TransactionSigner`, which also takes an extra argument which is the connection to the machine readable travel document. The rest of the transaction was built in the traditional way by adding the previous transaction and specifying new outputs. After the transaction was built and signed, it was broadcast over the network (the one connected peer in a testing environment).

Unfortunately, this did not work as expected. The transaction got rejected immediately after broadcasting. Some investigations into this matter resulted in an error `could not connect inputs`. The transactions in ePassportChain are build according to the Bitcoin standard[2], which initially is hard to understand, especially if there are modifications to the signature and the way the transactions are verified. After some debugging, it did not seem to be working using the transaction builder in Bitcoinj. However, to make sure that this was not a flaw in the ePassportChain, some cross-checking was performed using a transaction built in ePassportChain CLI. This CLI is able to completely build a transaction, sign it and not broadcast it to the network (this is possible with the RPC call `createrawtransaction`).

In order to check if ePassportChain worked correctly, the transaction, built with the CLI, was encoded as a byte array and hardcoded in the testing environment which broadcasts it to the network. This worked as expected and the transaction became visible in the block explorer instantaneously. Based on these findings, it turned out that the ePassportChain implementation did work well and the fault was in the usage of the BitcoinJ library.



(a) Transaction created (b) Transaction broadcast (c) Transaction accepted

Figure 5.1: The several states the app is in while redeeming a digital voting pass

To gain complete understanding of the way a transaction is formed, a transaction was built using pencil and paper according to an online step by step guide[3]. This enabled us to keep control over the transaction at byte level. After a couple of little corrections, the correct translation to the MultiChain format was made and it was accepted by the network. This made it possible to generate a transaction in the same format from code. To keep track of the code quality and respect the BitcoinJ structure, a decision was made to extend the standard `Transaction` class to a `PassportTransaction` class. This class is able to build an entire transaction on its own by providing an unspent transaction output and a passport connection. The transaction is built to send the entire amount of voting tokens back to the government wallet and sign the transaction using the provided passport connection. The broadcast of the transaction happens in the regular way using Bitcoinj. Figure 5.1 displays the several states the app is in during the creation and broadcasting of a transaction.

[2]https://en.bitcoin.it/wiki/Transaction
[3]https://bitcoin.stackexchange.com/a/5241

5.2. System overview

This section gives an overview of the system in its form as a full prototype. In Figure E.3 (Appendix E) screenshots of the final prototype are shown. This gives an idea of how volunteers of a polling station would experience the system.

Figure 5.2 shows a sequence diagram of the polling station application. It shows the normal flow through the activities on an election day. It starts with starting the app initially, synchronizes the blockchain and then moves on to the main activity. Here, the main loop starts. The main loop consists of scanning a documents MRZ first. Then, communicating with the document through NFC. After the NFC communication is complete the `Blockchain` object is queried for the amount of voting passes the document has access to, this happens in the `ResultActivity`. In the `ResultActivity` the transaction history can be requested, though this is not required and so not part of the main application loop. Suffrage is redeemed by pressing a button which causes the transaction to be broadcast. When the transaction is confirmed by the network, a message is received, then the polling station volunteer presses *next voter* and the main loop starts over. For readability, the `ManualInputActivity` is left out in this diagram. If manual input were used to enter info of the MRZ zone, then `ManualInputActivity` would replace `CameraActivity` and `TesseractOCR`, leaving the rest of the sequence intact.

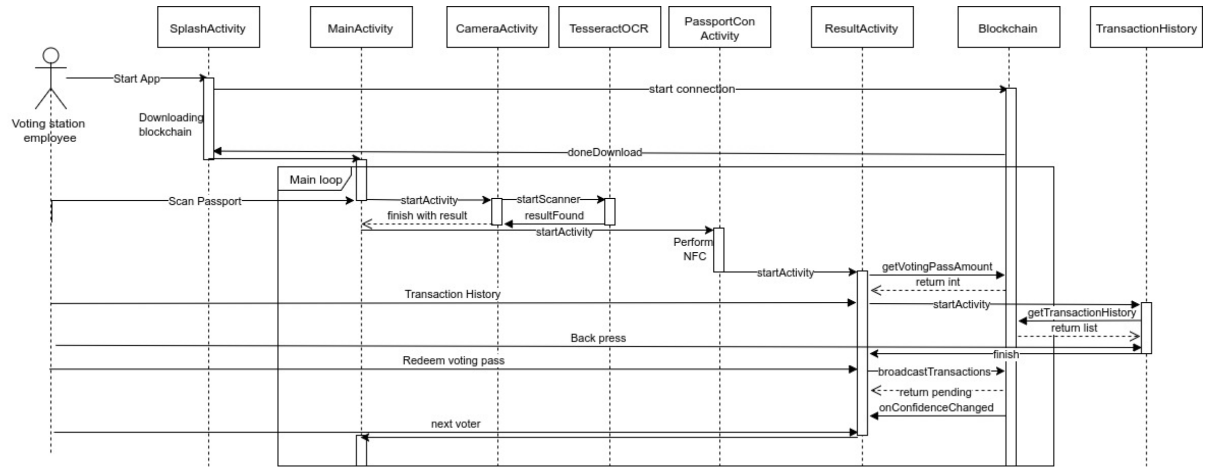


Figure 5.2: Sequence diagram of application

5.3. Expert feedback

Five municipalities in The Netherlands have the intention to develop a voting app for calling local polls for various issues in their town, e.g. whether a roundabout should be created or not. They want to hold these polls to the same conditions as the nationwide elections. The vision for the long term is that the system for these local polls can also be used for nationwide elections. Since these municipalities are already in the process of developing a digital voting system, it is interesting to get their opinion about this proposed solution.

We were brought into contact with the town of Emmen, located in the north east of The Netherlands, who are one of the five municipalities looking into digital voting. They were happy to receive us to talk about our proposed solution.

5.3.1. Why digital voting

From the point of view of this project it is very interesting to know why a municipality wants to implement digital voting. They stated some very clear reasons:

- Easier participation in elections for citizens
- Increased accuracy for counting votes, current system is very fault-prone
- Faster counting of votes
- Much lower costs

- Easier organization, less logistical issues

These improvements are important for local polls. They need to be very easy to participate in, because people will probably give less priority to these polls in comparison to regular elections. Furthermore these polls will be held with a much higher frequency and going to the polling station every few weeks is just not viable.

The municipalities want to slowly scale the application in order to serve more people and achieve more general use for elections. They are in contact with a few computer experts and will continue to keep experts involved in the process. With this combination of a slow build-up and the backing of experts there is time for the solutions to gain the trust of the general public. Our proposed solution fits in the idea of slowly building towards a digital voting system.

5.3.2. Demo of the proposed system

We explained the goals, scope and setup of the project. Because not everyone was familiar with the technology used, a simplified explanation of blockchain was given. This was expanded to the concept of public/private key-pairs in combination with Dutch travel documents. After this theoretical presentation, a demo of the app was presented going through every step a polling station volunteer would also go through.

5.3.3. Feedback

The representatives of the municipality had a positive response towards the presented solution. They see the system as a nice first step towards a full digital voting procedure. Furthermore they were very interested in other applications of the passport connection to the blockchain, as this could solve some of the problems with digital identity. They also had some interesting points which should be addressed before the system is put to use in the real world.

It should be possible to keep the blockchain in sync with databases of the government to invalidate the digital voting pass of deceased people, currently this is done by keeping lists of uniquely identifying numbers of invalidated voting passes.

Currently the ballot papers and voting passes are destroyed three months after the election. This happens in order to have less discussions about the outcome of the elections a long time after they have taken place. Destroying an open system like a blockchain is not possible, since everyone has the ability to copy the blockchain, therefore the election outcome will be open for debate for a longer period of time.

In order to keep voting accessible for everyone, they want to keep the option for paper voting open. People who have serious objections to digital voting for whatever reason would still be able to vote on paper.

From a transparency point of view, a digital solution would be a major improvement. Currently in The Netherlands it is possible to attend the counting of the votes. However, during the last election in Emmen, as stated by one of the representatives of the municipality, there was only an extremely small amount of people who actually made use of that right. A digital solution would make this much easier.

It is great to get a confirmation that the goal of the project, taking a small step towards digital voting, is achieved by the proposed system. From the response of the municipality it becomes clear that compatibility with the current voting system is very important. They want to keep the paper option open and even though one could argue that it is not needed to take into account everyone's preferences, this compatibility is essential for the slow adoption of a a full digital voting system. The proposed system of this thesis allows for compatibility with the current paper voting system.

Finding a good solution for keeping the blockchain up to date with the governmental databases is something that will be a complex problem as this connection, going from centralized to decentralized, will be a weak point in the system. Close contact with system experts from the government will be required in order to find a good solution for this connection.

Not being able to throw away the blockchain will probably not be a major issue. With the increased accuracy of counting the amount of voting passes there will be less discussion about the election outcome. Therefore it is no problem if the blockchain is available a long time after election, it even increases transparency which has a positive effect on trust in the voting system.

6 Performance analysis and evaluation

During the last parliamentary election in the Netherlands there were around 13 million voters [37]. If the current analog voting passes are replaced by this digital solution, this system has to cope with 13 million transactions on a single day. In order to evaluate the scalability of our system, multiple performance tests were executed in order to analyze the systems' performance on a larger scale. Furthermore, the security of the system is discussed.

Four performance tests have been executed in order to simulate the entire process; ranging from the creation of an election to voters claiming their right to vote. Each test will be discussed in a separate section and has a fixed layout. First, an elaboration of the setup is given, which discusses how a test is performed. After this, the results are discussed and evaluated. All tests were executed on a single machine which is a MSI GP62 2QE Leopard Pro. This machine has an Intel® Core™ i7 5950HQ, 8GB of RAM and SSD storage.

6.1. Verifying transaction

This performance test will try to measure the difference in time between verifying a single transaction with MultiChain and ePassportChain. To verify a transaction, a hash of the transaction, signature and a public key must be provided. Before trying to verify a signature using a public key, the public key itself must be checked for validity. In this test, the time to verify a public key and a signature is measured, because both are tested for correctness every time a transaction has to be verified.

In this test, two new chains were created, one for ePassportChain and one for MultiChain. 1 000 public keys of both chains were created and stored in a CSV file. For each chain, a Python script has been used to send 1 000 transactions to the 1 000 public keys. This script communicates with the a node via RPC. All transactions were sent to a single node, which verifies all transactions. This node would then return the elapsed time, in milliseconds, it took to verify a single transaction.

Two test runs have been completed and the results are shown in Table 6.1. This table clearly shows that MultiChain performs significantly better than ePassportChain. The main difference comes from the fact that

	MultiChain	**ePassportChain**
Run 1	0.0655 ms	17.1 ms
Run 2	0.0721 ms	17.3 ms
Average	0.0688 ms	17.2 ms

Table 6.1: Elapsed time of verifying a transaction

secp256k1 is 30% faster than most elliptic curves [39]. In contrary with other elliptic curves, secp256k1 signatures can be efficiently computed since secp256k1 generates signatures in a non-random way. Next to this, ePassportChain uses the CryptoPP library which is not optimized for the brainpoolP320r1 elliptic curve. Furthermore, MultiChain has an optimized implementation of the secp256k1 elliptic curve from which it benefits significantly. The last difference comes from the fact that ePassportChain must verify a signature four times instead of one, as explained in subsection 3.1.1.

In order to calculate the impact of verifying only one signature in ePassportChain, another test has been executed. This test measures the time it took to verify one part of a hash, and thus verifies one signature. Using the same setup as discussed in the previous section, 1 000 transactions were sent to a single node. This test showed that verifying one fourth of a hash on average took 2.85 milliseconds. From this data it is possible to calculate the hash verify time when it is possible to sign the whole 32-byte hash in one go:

$$17.2 - 2.85 * 3 = 8.65 \text{ ms}$$

This means that in the future, when a travel document is able to sign more than 32 bytes, performance can be increased. Verifying a signature can be almost twice as fast, if CryptoPP is used.

6.2. Creating elections

Before a voter can claim his voting right, the voter must have at least one voting token on ePassportChain. The voter receives its token from the government, since the government knows who is eligible to vote. This means that the government first has to distribute voting tokens to all individuals who are eligible to vote. As stated before, there are about 13 million persons eligible to vote which means that the government has to distribute about 13 million voting tokens. In this section, the creation of an election is simulated and the performance of ePassportChain is analyzed. In addition, the size of of the chain and the time it takes to create an election are measured in this test.

In order to create an election, a Python script was written. This script first creates an asset on the chain which represents an election. Then it loads public keys from a CSV file and converts these public keys to ePassportChain addresses. Subsequently, every address is granted the appropriate permissions and the script sends one voting token to each address. The script sends the transactions to a single node which processes all the transactions. All of these steps are done via RPC, specifically the following calls are used: `issue`, `grant` and `sendasset`.

For this test 400 assets have been created and for each asset a CSV file containing 1 000 public keys has been used, thus creating 1 000 addresses where voting token can be send to. This means that in total 400 000 transactions are generated and processed in this test. For each asset, the elapsed time to generate and process the 1 000 transactions is measured. Furthermore, the size of the chain is measured at four different moments, after 100 000, 200 000, 300 000 and 400 000 transactions.

Figure 6.1 visualizes the time it took to complete each test run. From this figure it is clear that the time it takes to process transactions stays constant when the chain grows.

The average processing time of 1 000 transaction was determined to 37.3 seconds. With this information it is possible to calculate how long it should take to create an election for 13 million people:

$$\frac{13\,000\,000}{1\,000} * 37.3 = 484\,900 \text{ seconds} = 5.61 \text{ days}$$

This means that a government must at least create an election six days before the actual elections if the government runs similar hardware. Better performance can be expected when an election is created on superior hardware.

In Table 6.2 the size of the chain after 100 000. 200 000, 300 000 and 400 000 transactions can be found. Using this data, it is possible to calculate how fast the chain grows in size, which is determined to be around 78.5 MB per 100 000 transactions. From this, it is possible to calculate the chain size after 13 million transactions:

$$\frac{13\,000\,000}{100\,000} * 78.5 = 10.2 \text{ GB}$$

Ten gigabyte is quite big but can be downloaded by a mobile device with via WiFi connection on a single day. Furthermore, the mobile device which runs the polling station app, needs to have at least 21.4 GB (twice as many since 13 million voters also create transactions) of storage to be functional.

6.3. Simulating the voting process

The following test simulates people claiming their voting right at the polling station. Specifically, it measures the time it takes to verify multiple transaction by a node. In a real world scenario, voters will claim their right

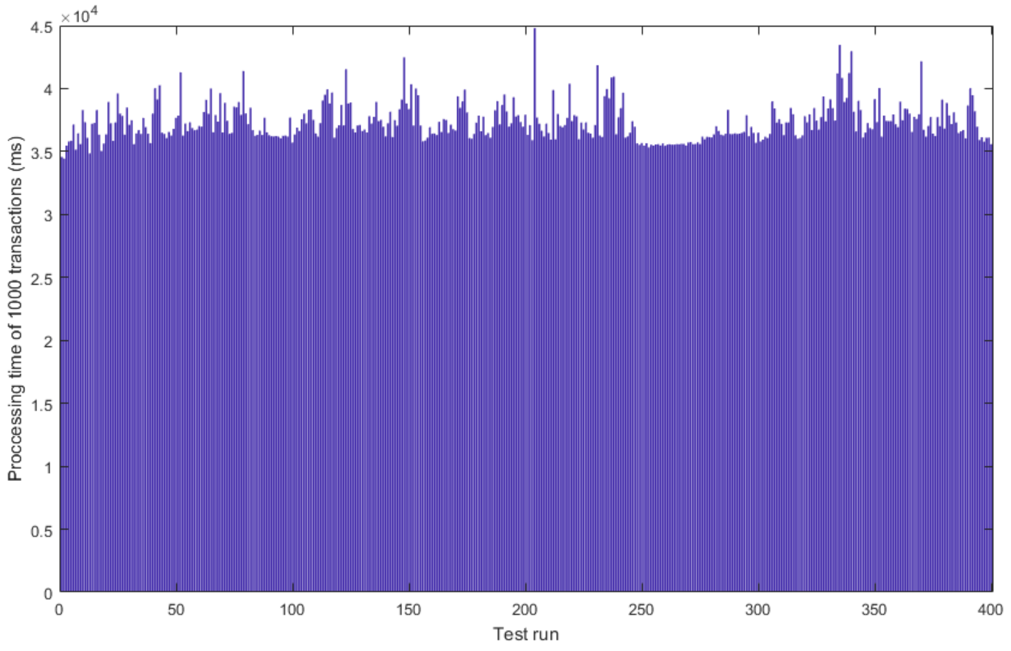


Figure 6.1: Simulation run of the creation of an election on ePassportChain

Number of transactions	Size of blocks (MB)
100 000	95.0
200 000	173
300 000	252
400 000	331

Table 6.2: Impact of transactions on the size of ePassportChain

from almost 9 000 polling stations [40] and thus 8 000 transactions could be send concurrently. From the results of these tests, it should be clear whether or not it is feasible to create one ePassportChain for 13 million voters.

This test is different than previous tests as described in section 6.2. The test from section 6.2 builds a transaction on the node itself and thus signs and verifies the transaction on the node. In the test described in this section, a node only verifies a transaction. This test creates 1 000 new addresses and sends each one a voting token. Subsequently, it creates 1 000 transactions (one for each of the 1 000 addresses), signs these transactions and stores the signed transactions. Finally, the signed transactions are loaded and sent to the node which solely verifies a transaction and measures the time it takes to verify all 1 000 transactions. Similarly, in a real world scenario a transaction is build on a mobile device and sent to a node which in turn verifies the transaction. To ensure a reliable average this test was run 100 times.

In Figure 6.2, a bar graph of the results is shown. From this data it is possible to calculate the time it takes to verify 1000 transactions, which is determined to be 23.5 seconds on average. The following calculation shows how long it should take to verify 13 million transactions:

$$\frac{13\,000\,000}{1\,000} * 23.5 = 305\,500 \text{ seconds} = 84.9 \text{ hours}$$

Obviously, 84.9 hours is too much for an election. Besides this, voting rights can only be redeemed when a transaction is confirmed by the network and thus is contained in a block. When the network is under heavy load, the number of unconfirmed transactions will grow. This will result in a greater waiting time for voters.

A possible solution to this problem could be to create separate ePassportChains for different regions. This way the load can be divided and the waiting time of a voter is reduced. The size of a region should depend on

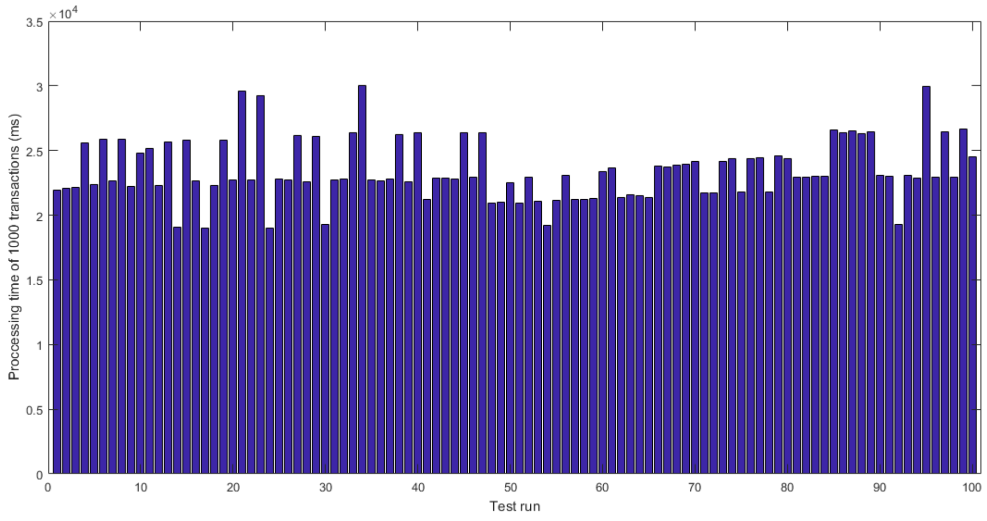


Figure 6.2: Bar graph of a simulation run of sending transactions on ePassportChain

the number of transactions ePassportChain can process in 10 seconds (the target block time) and the number of polling stations. These numbers should roughly be equal so the number of unconfirmed transactions does not grow when a new block is mined. The maximum number of transactions that can be processed in 10 seconds can be calculated as follows:

$$\frac{\text{amount of transactions}}{\text{average verify time}} * \text{block time} = \frac{1\,000}{23.5} * 10 = 425 \text{ transactions/10 seconds}$$

This means that approximately 425 transactions can be processed in 10 seconds. It is recommend that a single ePassportChain is used by around 425 polling stations. However, it is possible to have, for example, twice as many polling stations, but this will double the waiting time of voters.

6.4. App performance

When the polling station app is initializing, it downloads the chain (or new blocks). In this test the time it takes to initialize the app is measured. It will do this by measuring the time it takes to initialize the app when ePassportChain contains 0, 1 000, 2 000, ..., 8 000 transactions. From this data it is possible to calculate how long it should take to initialize the app after an election has been created.

The app is run on a OnePlus One mobile device, which connects to a single node and downloads the chain from this node. Initially, a new ePassportChain is created containing zero transactions. After the chain is created the following steps are taken:

1. Clear the data and cache of the app
2. Send 1 000 transactions to ePassportChain
3. Start the app and measure the time it takes to initialize the app
4. Repeat until there are 8 000 transactions in ePassportChain

The result of this test can be found in Table 6.3 and a graph of this data is shown in Figure 6.3. Figure 6.3 shows that the time it takes to initialize the app is not linear but exponential. Even if we assume that the elapsed time grows linear with 350 seconds per 1 000 transactions and we create separate chains as discussed in section 6.3, it clearly performs poor. The time it would take to initialize the app when the chain contains 1 million transactions can be calculated as follows:

$$350 * 1\,000 \approx 4 \text{ days}$$

This is unacceptable for major elections such as the parliamentary elections. Better performance is definitely possible but due to limited time in this project, this was not implemented.

Number of transactions	Time (s)
0	5.43
1 000	36.1
2 000	87.6
3 000	158
4 000	277
5 000	399
6 000	590
7 000	796
8 000	1 063
9 000	1 407
10 000	1 788

Table 6.3: Elapsed time for the app to initialize

6.5. Security level

The application is built upon two key elements: the blockchain and the ePassport. These key elements are the most important factors of the security aspect of this project. The following section will cover the possible flaws in these parts of the implementation.

6.5.1. Blockchain

The ePassportChain is set up in a decentralized way, which means that everyone is able to take part in the network. In order to prevent a 51% attack on the network there are rules to which every node has to comply. One of these rules is the authorization to mine new blocks, which is limited to the government node. If anyone tries to alter the parameters and set up its own ePassportChain, it will not match and the polling station therefore will not be able to connect to the new node. The genesis block contains these parameters and is part of the blockchain.

Another attack would consist of intercepting the private key from the government. If this happens the attacker would be able to provide its own voting tokens and mine blocks by itself. Such an attack would be visible by anyone with access to the blockchain, especially the government itself will see unauthorized transactions happen. Even though the attack will be detected, preventing it is preferable. In order to prevent the leaking of the private key, the government should consider a hardware wallet[1]. This hardware wallet ensures that it is not possible to read the private key from the device (like the ePassport). However, it will still be possible to sign transactions using this hardware wallet.

6.5.2. ePassport

To visualize how hard it is to break a security, the concept of universal security was introduced by Lenstra [41]. In this research the security level of an application is expressed in the amount of energy needed to break an encryption. The elliptic curve used by Dutch travel documents has a key length of 320 bit. According to Lenstra's research, an elliptic curve key of length 228 bit is comparable with an RSA key with bit length 2 380. This key has strength "global security" which means that the breaking of the key requires the equivalent of the amount of energy needed to boil all water on earth.

The implemented solution, however, uses ECB for signing the hash of a transaction. ECB means that different blocks are encrypted or signed separately. This may result in some vulnerabilities. ECB chaining has the following limitations [42]:

- If the blocks are the same, the ciphertext is also the same.
- If the blocks are independent, it becomes possible to replace blocks in the ciphertext
- If one bit is inserted or lost in the ciphertext, the decryption of the entire ciphertext will go wrong

The first limitation is not a problem with elliptic curves due to the way the signatures are generated. The signatures are generated non-deterministic, due to the addition of a random number. Signing the same data twice with the same key will result in two different valid signatures. Secondly, replacing independent blocks

[1]https://en.bitcoin.it/wiki/Hardware_wallet

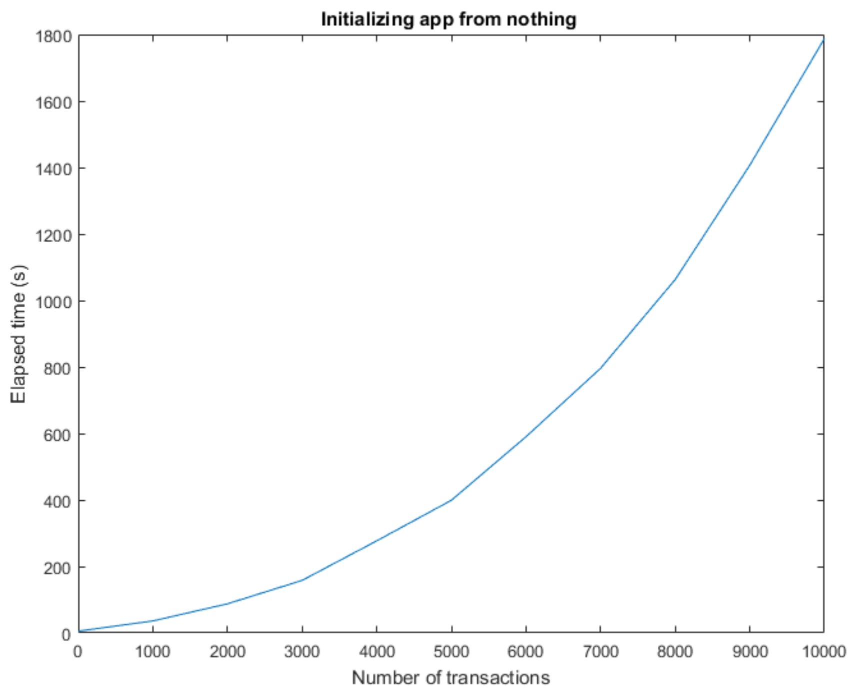


Figure 6.3: A graph of the time it takes to initialize the app when there are between 0 and 10 000 transactions in ePassportChain

is not really an issue in this implementation, because the only data that is signed is a SHA256 hash. The four blocks in this hash are not independent and cannot be used individually. At last, the insertion of a bit, which may occur due to hardware failure, will only result in a rejected transaction. It will not compromise the security.

This analysis is performed without any background knowledge of encryption and should be verified by an cryptographic expert before the prototype is used in the real world.

7 Quality assurance

Good quality code is essential for a successful software project. In accordance with the Project Plan (Appendix B) the quality of the code is maintained by creating tests, using continuous integration and by the expert feedback of the Software Improvement Group (SIG). This chapter elaborates on these specific components for maintaining quality code.

7.1. Software Improvement Group

There is no exact measure for good code. It is especially hard to define exact measures since software can have very different goals and therefore a totally different structure. However, the software engineering industry strives towards reaching agreements on code quality. This resulted in ISO 25010, which defines a quality model for code in more general terms [43]. The existing agreements on good quality code are mainly focused on maintainability: the ability to change code without much effort and generally being able to easily analyze code.

The best way to get an idea of the quality of code is to compare it to similar software. When software is comparable to most of the software that is released, it is certain that the software delivered has the same quality as the industry standard. When there is any difference, further analysis should point out whether or not that is positive for this particular piece of software.

With this idea in mind the Software Improvement Group (SIG) was founded. They provide a benchmarking service for code, comparing it to the data they have of their many costumers. SIG provides this service for all the Delft University of Technology Computer Science Bachelor projects. For these projects there are two benchmark points, one in week six of the project and one in week ten. The entire code base is uploaded to SIG, after which they provide feedback in the form of stars and an expert review. The stars (one to five) indicate the quality of the code in comparison to other code bases.

7.1.1. First code submission

The first submission included the code of the polling station app (at that time not yet including the connection with the blockchain) and some scripts to easily issue digital voting passes. SIG has the policy that when third party code is adopted, the new owner is fully responsible for the code. We make use of the MultiChain library which is about 230 000 lines of which we edited about 400 lines. Because it is unrealistic to check and fix multiple man years of code while we changed so little, we decided against including these libraries in the SIG submission. Such a large library would also defeat the purpose of the SIG upload, namely checking if we write good quality code.

The feedback on the submission was very positive. The code received four stars on the maintainability scale, this means that the code is above average maintainable. The advise was given to take a better look at the unit size, which was above average large. Especially the class relating to the camera preview for OCR had very large methods. As a positive aspect the amount of test code was mentioned.

With this feedback in mind the code was revisited to reduce the unit size and to refactor the code in order to better distribute responsibilities. Larger methods were split up into multiple methods having a more distinct responsibility, reducing the unit size. The camera related code was split up into multiple classes creating a better overview of where certain functionality resides in the code and thus increasing maintainability. Furthermore we continued creating unit tests where possible and integration tests for other parts.

7.1.2. Second code submission

In the second submission the same repositories as with the first submission were submitted, now with the polling station app in its full prototype form. Again, large library files were excluded. Because the same repositories were submitted, the progress of the code quality can be analyzed easily.

The second feedback was again positive as the maintainability of the code has increased. The unit size is slightly decreased due to the refactoring as mentioned in the previous section. A bigger decrease did not occur because new long methods have been introduced, which is something that needs to be taken care of. However, some methods are intentionally kept long because they follow a specific steps to make ensure that developers can comprehend how these methods work.

7.2. Testing

When editing parts of a software system it is almost inevitable that some functionality breaks and stops working as intended. It is very important to notice these failures early on, before the product is released to the public or before more code is created using failing building blocks. To help in noticing these failures and to ensure parts of the code are working as intended, tests should be created. These tests are able to run a small part of the entire code base in order to test their specific functionality. Combining tests with a building service like Travis CI[1] creates an automated system which warns when an alteration to the code breaks anything, this is called continuous integration.

We have made use of continuous integration for the entire developing process of the polling station app. Although this is generally considered good practice because issues are detected early on, not many issues were fixed with the help of Travis. Most of the time errors raised by Travis were caused by compatibility issues between Android and Travis or by setup file issues.

Some functionality of the polling station app is so complex and has dependencies in such a way that it can't be tested with regular unit tests and therefore is harder to test with continuous integration. Because of many issues with Travis and integration tests, this functionality was not included the continuous integration setup. How this functionality was tested is explained in detail in the sections that follow.

7.2.1. Passport connection

The passport connection related methods all work with the `PassportService` class from JMRTD. This `PassportService` handles all requests and responses to and from the passport and makes use of an `InputStream`. Because in unit tests a passport (hardware) can't be used, this `PassportService` is mocked as well as the stream of data coming from the chip.

The `InputStream` is mocked with data that represents a brainpoolP320r1 public key, which should get handled similarly to data from an actual passport. However, this raises an exception in the method that reads the public key from the data. Even when actual data from a passport is copied and used as `InputStream`, it still raises an exception.

Because a lot of time was spent on trying to get these tests to work without any result, it was decided to not test these methods by using unit tests. Instead AndroidTests or a testing plan could be set up to manually test the functionality of the passport connection before each release. Even though this means Travis CI can't be used for continuous integration tests, this proposed solution is a way to ensure no functionality is broken by new releases.

AndroidTests or instrumented/integration tests are used to test software on dedicated hardware. This project has a problem in that several user actions are required for testing the full functionality (e.g. scanning the MRZ and holding the passport to the phone). This makes it very hard to test for specific cases. Only the case of a successful passport connection can be tested, but this is the same as testing the functionality of the app. It was decided that creating separate tests for this is not worth the time. Testing for how the app handles the error cases (e.g. passport is held to the phone too short) is possible by using regular unit tests. Since the tests will handle the cases with an empty response, there is no need to mock any data streams, which makes

[1]https://travis-ci.org/

testing easier. Testing for the handling of the NFC tag discovery was not successful, because mocking NFC tag intents is not possible, methods to do so are not available in newer Android versions.

For each release to master, a testing plan should be executed in order to ensure non-tested functionality of the application is still working. The test plan for the passport connection can be found in section G.1 (Appendix G). Using a test plan instead of unit tests or integration tests has serious limitations. The amount of test cases is very limited and it is much harder to test for rare bugs. Furthermore, it is hard to maintain a consistent test environment, which would make it harder to debug. Although this setup is far from ideal, it is sufficient to ensure a working prototype for releases.

7.2.2. Optical character recognition

Because Tesseract uses a number of dependencies that cannot be mocked, AndroidTests were used instead of regular tests. With AndroidTests is it possible to execute tests in an emulated environment or on actual hardware.

Using AndroidTest and images of travel documents it was possible to verify the accuracy of the OCR. During the use of the application the OCR scans the camera input until it recognizes a correct MRZ. Therefore there is no need for 100% accuracy on each picture. To account for this, a threshold was set for accuracy in the tests. This accuracy was calculated using the Levenshtein distance, which is a measure for the difference between strings. The resulting MRZ string from the OCR scanning of the test images should at least have a reasonable accuracy, like 95%. By setting the accuracy threshold to just below the lowest performing test, any drop in performance of the OCR scanner can be detected by continuous integration, since test will start failing.

These tests are also essential in order to test for improvements of the OCR. Using the reported accuracy, improvements to the OCR can easily be detected. After creating these tests several trained data files were tested to see if any caused an improvement in scanning the MRZ. Eventually a trained data file was created based on the OCR-B TTF file. This resulted in a five percent increase in accuracy. In the real world, however, it resulted in much faster recognition of the MRZ. On passports where reading the MRZ used to be impossible, now the scanner recognized the MRZ immediately after the camera had a good focus. This improvement is important for the ease-of-use requirement of the system.

7.2.3. ePassportChain

Testing ePassportChain turned out to be very tough since MultiChain itself does not have any public test code [44]. This meant that a test framework, tests itself and a Makefile which compiles both, needs to be added. The test framework that has been used is called Catch[2]. With this framework it is possible to write automated unit tests. Unfortunately, creating a Makefile which compiles the tests was unexpected difficult due to build errors. These errors were eventually fixed after a lot of work.

Since only a minor part of the MultiChain code was altered. The implemented test suite only tests whether loading, storing, signing and verifying with keys works as expected. To ensure that changes to ePassportChain don't break functionality, a test plan has been created, which can be found in section G.2. This test plan must be completed after each change to ensure everything still works as expected.

[2]`https://github.com/philsquared/Catch`

8 Societal impact

Voting is a very important part of democracy. The core of democracy, having influence as a citizen, is compromised when anything goes wrong during the voting process. Because of this, it can be expected that adoption of the proposed system in this thesis or systems alike will cause society-wide debates and therefore have a serious impact on society even before they are implemented.

The fact that digital voting is often met with skepticism, is not strange. Fears for digital solutions include large scale fraud, violation of secret ballot, and meddling in election by other governments, as became clear when digital voting was discussed in the Dutch House of representatives on May 17th 2016 [45]. The argument of external governments meddling in elections has become more prominent since accusations of Russian interference in the 2016 US presidential elections [46].

8.1. Political response

Changing the voting process is politically sensitive. Even a relatively simple alteration like changing the size and layout of the ballot paper is met with resistance [47]. Especially proposals for digital alternatives are strongly opposed. In the past years every proposal for experimentation with digital voting was declined. Because there was so much negative feedback on the previous digital voting system, any new system will be looked at with skepticism. Because voting is such an integral part of the democratic system, any failure in the system degrades the democratic constitutional state. Anyone responsible for the failure will be facing a lot of criticism. A secretary who is responsible for the implementation of the failing system will possibly have to resign. For these reasons it is not strange that politics is hesitant to change anything in the voting system while no big problems have come to light in the current system.

People place trust in the current paper system because everything can be verified and it is hard to commit fraud on a large scale. After the parliamentary elections of 2017, it was uncovered that there were at least 14 000 differences between official election results and the vote count of municipalities [3]. It can be expected that this will spark a debate in parliament whether or not the current system is safe enough. This would also be the right moment to bring up solutions like the one proposed in this thesis.

8.2. Required legislative changes

The Dutch voting process is thoroughly described in the law (Kieswet). Any changes to the voting process therefore require changes in law. By adopting an experiment law for voting, it would be possible to experiment with other forms of voting processes without having to change the law describing the voting process. Such an 'experiment law' has been used to create the possibility for experiments with voting passes for Dutch citizens abroad [48] in the past. A proposed experiment law for allowing tests for the use of electronics during the elections was turned down by parliament in 2016 with about two thirds of parliament against [49]. It took about three months from the first proposal of the law till the vote in parliament.

It can be expected that when the majority is in favor of digitizing the voting pass, an experiment law could be passed within a few months. Changing the actual voting law (Kieswet) will probably take a lot longer, up to multiple years. This can be derived from the reasoning to propose the in 2016 turned down experiment law. It argued that the experiment law was needed to be able to start the use of electronic voting systems without having to wait for a law-change process that could last multiple years.

8.3. Trust in the system

It is obvious that the system must be secure to prevent anyone from influencing the election outcome and that people should trust the system. But trust is not directly correlated with security of the system. Pieters (2006) [50] argues that trust can be modeled as a combination of safety (does the system perform as it should under normal circumstances), security (is the system tamper proof) and how this safety and security is perceived. A system might be perceived as secure and therefore be trusted, even though it is not secure at all. The other way around is just as undesirable, because when people don't trust the system they might not want to vote or will not respect the election outcome. Trust in the system, therefore, is a key aspect of developing a working digital voting solution.

In 2007 De Jong et al [51]. researched the difference of trust in voting procedures. They concluded that people had a little bit more trust in voting machines than in a paper procedure. People had more confidence in the correct counting of the votes with machines. A paper procedure, however, was considered more anonymous. Since that time a lot has changed in the perception of digital solutions. Nonetheless, the fact that 90% of Dutch municipalities would like more research on digital voting solutions, shows that the public still has confidence in these solutions [52]. These municipalities would not show interest in these digital solutions if they know that their citizens do not trust it.

The trust of the general public in a digital solution will probably be influenced in a major way by the opinion of experts. Therefore developers of any digital voting solution should aim to convince these experts of the safety and viability of the system.

8.4. Changes for voters

In the most recent parliamentary elections in the Netherlands about 10.5 million people cast their vote, this is a turn-out of 81.9% [53]. This means that many people come in contact with the voting process. The system should be accessible for everyone and it should be known what the effects of a new system are on the voters.

Changes in a system or processes often cause problems as people need to adapt. It can be expected that in such a large group of people, like the eligible voters in the Netherlands, there will be people who are not aware of the system change. Therefore this thesis proposes a transition phase where people can opt-in for using the digital voting pass. After a few elections, when the system will be optimized and proven to be working as intended, people will have more trust in the system and it will be easier to oblige people to start using the new system.

8.4.1. Positive impact

The proposed system is easier to use for voters, since they only need to remember to bring their traveling document, which most people have on them most of the time. Because the system is easier, it could be expected that people will be more inclined to vote, even though it is hard to give a good estimation on this. An easier form of voting in Estonia, where people can vote via Internet, only slightly increased the voter turn-out [54]. Research by Allers and Kooreman showed no statistical increase in voter turn-out in Dutch elections comparing paper voting or voting machines in the 2003 and 2006 Dutch elections [55]. Because cultures and voting solutions differ so much it is hard to distill anything useful from previous research for the proposed solution in this thesis. Research and experiments should make clear whether or not a positive effect on voter turn-out can also be expected in the Netherlands.

In a day and age where a large part of processes are already digitalized, the voting system runs behind. The reputation of voting could be improved by having a digital system associated with it, making it less old fashioned. An open-source voting system involving blockchain technology maximizes transparency of the voting process. With the right infrastructure in place (which can be created by anyone) the voting process can become extremely transparent for any citizen with basic knowledge of technology. Increased transparency allows people to fully understand what happens with votes, which increases the trust in the voting process and democracy as a whole.

8.4.2. Negative impact

At first it might be hard for people to understand why our proposed system is safe. Furthermore the relationship between a travel document, a phone and a voting pass might be hard to understand. Very clear explanations of the system are essential in order to take away any misunderstandings about the system.

Related to this there are people who fundamentally distrust digital systems. Often this has to do with incomprehension of the technology. Effort should be put into persuading these people the system is safe.

8.5. Ethical implications

This section briefly discusses the ethical considerations that were made during this thesis. Elections are surrounded with multiple ethical issues. The two main issues are privacy and the right to vote.

Since the proposed system is transparent and open, it opens up the way to privacy issues. Information about voters, whether or not they voted and when is publicly available to everyone. In the Dutch society everyone can move freely in public space and everyone is able to see where other people are in public space. Polling stations are located in the public space and therefore information about whether people go to a polling station is also publicly available. We therefore did not consider it to be a problem that the blockchain shows this information. Furthermore it is not easy to link an address on the blockchain to a person since access to the travel document is required.

During the scanning of a travel document it is possible to read and save all kinds of privacy-sensitive information e.g. the social security number. In the app, none of this information is permanently saved and only the information necessary for BAC and the public key is read from the travel document. Thus the privacy of the voter is warranted to a high degree.

The second point considered is the accessibility to the voting system. Voting is a fundamental right of people and from an ethical point of view everyone should be able to claim their voting right with relative ease. A new system should not make it harder or impossible for certain people to vote. The proposed system strives to make the system easier for voters. No extra knowledge about the system is necessary in order to claim suffrage. Therefore the system is even more accessible than the old system, since only one document is needed for voting. For people who have serious issues with voting digitally, the paper system could be kept as an alternative since it is fully compatible with the proposed system.

9 Conclusions

This thesis presents a fully working prototype that can substitute the existing system of voting passes in the Netherlands for a digital one. It removes the necessity of sending paper voting passes to all Dutch citizens that are eligible to vote and improves the counting of votes due to less manual labor.

Using multiple open source libraries and frameworks we were able to integrate the ePassport standard with blockchain technology by signing blockchain transactions using the chip present in travel documents. This project is the first in the world to show this possibility. The digital voting pass is stored on the blockchain, which acts as a tamper-proof database. The combination of ePassport and blockchain guarantees that only with a valid travel document a voter can redeem his vote.

Combining fully open-source code and the natural transparency of blockchain technology creates a fully transparent system, which is a necessity in order to build trust for the system. Because of the usage of a decentralized setup of the blockchain the system has no single point of failure. Furthermore, the system simplifies the redeeming of suffrage and increases accuracy of voter turn-out, which improves the detection of fraud within elections. By developing an Android app, we presented a working prototype of how the system could be used in polling stations.

With a literature review, we showed that the key aspects of the proposed system are secure. Better analysis by security experts is still required in order to confirm this conclusion. Even though the system performs very well in a small testing environment, it has some scalability issues. These issues can be largely addressed by splitting the system up into parts and by realizing further optimization of the blockchain.

The use of the ePassport standard to authenticate people is not limited to the digitalization of the voting process. In fact it is also possible to apply this system to many other problems that need to deal with digital identity. This opens up some interesting directions for further research for solving the digital identity problem.

After performance issues are addressed, some minor improvements as stated in section 9.1 are needed, after which this new system can be integrated in the current Dutch voting system. When this is accomplished, the system can be seen as a first step towards a fully digital voting system.

9.1. Recommendations and improvements

Our solution to digitalizing the voting pass of the Dutch election system is not perfect and can be improved in multiple ways. This includes improvements for both security and integration in the current Dutch voting system. This section explains ways in which our solution could be improved in order to support all features of the current system and provide enhanced security and confidence in the proposed system.

9.1.1. Proxy voting

Proxy voting, the process of allowing someone to legally cast your vote for you, is not supported by our current implementation of the digital voting pass. We have, however thought about how this could be implemented. Before we zoom in on the possible implementation of proxy voting in the digital system, it is important to note though that in a working, complete digital voting system proxy voting becomes unnecessary. If a complete digital system is realized where any registered voter can vote digitally and remotely, the process of voting itself is so fast and simple that anyone can do it at pretty much any time during the elections.

However, in our digital solution to the voting pass each voter still needs to go to a voting booth in order to cast their vote. Therefore, proxy voting can still be desirable in this situation, in case a person is not able to go to the voting booth, because of, for example, physical reasons.

Currently, a travel document is needed at the polling station in order to claim a voting ballot. Since it is illegal to use someone else's travel document [56], it is impossible to cast a vote for another person by bringing their document to the polling station. Instead, a system needs to be implemented to allow voters that have received a digital voting pass to spend some other persons digital voting token. The simplest implementation of this would be to allow voters to transfer tokens to each other. Verification could be done by simply signing the transaction with both the voters' travel documents. This approach is also discussed in the research report (Appendix C). Another proposal to solve the problem of proxy voting in the proposed solution is to change the wallet of these voters to a multisignature one, allowing the proxy voters to spend those tokens as well.

In the current Dutch voting law, one person may not issue more than three votes in total. So, in order to conform to the law, this would need to be enforced within the blockchain so that any transaction that attempts to acquire more than three votes for an address is rejected.

According to our ideas, it should be possible to grant a vote to someone easily and quickly with an additional app. This app would need to read both the documents of the granter and the granted. The public key would be read from the document of the granted person, and then using the document of the granter, a transaction can be created and signed that sends the voting token to the wallet of the granted.

9.1.2. Multisignature wallets

The fact that MultiChain has multisignature support [57] is useful because it can add two additional features to our application. The first feature is that multisignature wallets can allow the government to withdraw anyone's voting rights. If the government key is given the authority to sign transactions of all wallets, it can simply create a transaction to take away someone's voting token and so his or her right to vote. This can be necessary when a person dies just before election day for example.

A second additional feature of multisignature wallets which is useful for our application, is about making the system less complicated for voters. In the current implementation a ballot paper can only be claimed using the same document that the government sent the token to. In reality some people have multiple different documents, and they do not care which one they use to vote. Instead of making people remember which of their documents to bring, their token could be sent to a multisignature wallet that allows both documents of that specific person to sign a transaction to spend the voting token.

9.1.3. Driver's licence support

Dutch drivers licences issued after November 2014 have the same chip embedded as ePassports [58]. This means that driver's licences starting from this date can also be used in the ePassportChain just like passports and ID cards. Since no driver's license was available during the development of our application, this functionality is not included in the polling station app. However, it should be quite easy to implement this functionality.

9.1.4. Possible safety improvements

Next to the above mentioned functional improvements, some security improvements can be made as well. These security improvements could ensure that a person showing up at a polling station, is actually the legal holder of the document. There are both biometric and regular authentication methods possible for this purpose. In the following paragraphs some possible security improvements are described.

The first security improvement could be facial recognition. Facial recognition can be implemented right away, since the picture on a travel document is downloadable by anyone having access to the document. This means that it is possible to load this picture and run facial recognition software on it, comparing it to the voter standing in front of a camera at the polling station.

Another more reliable method of ensuring document ownership could be fingerprint authentication. This however, would be much harder to accomplish and would be met way more resistance. First of all, since fingerprint information contains privacy sensitive information. Storing this information anywhere else than solely in memory for the authentication would be unacceptable. Additionally, it is currently impossible for a third party like ourselves to extract the fingerprint information out of the passport. In order to realize this the government would need to grant special privileges and an access key so that this information can be accessed.

A third option for additional authentication could be the combination of a brain wallet with the travel document. A brain wallet uses a list of words or a password as a seed in order to generate a private key. This can be used as an additional layer of security by making all wallets two of two multisignature, meaning that both private key signatures are required for the transaction to be valid. One of these signatures would be generated by a travel document, and the other would be generated by the device given the brain wallet pass phrase. This way both the travel document and the pass phrase would need to be present in order to redeem suffrage.

Bibliography

[1] Rop Gonggrijp and Willem-Jan Hengeveld. Nedap/groenendaal es3b voting computer, a security analysis, 2006. URL `http://wijvertrouwenstemcomputersniet.nl/images/9/91/Es3b-en.pdf`. Accessed: 2017-05-02.

[2] Elke stem telt, 2013. URL `https://www.rijksoverheid.nl/binaries/rijksoverheid/documenten/rapporten/2013/12/18/bijlage-bij-eindrapport-onderzoek-elektronisch-stemmen-in-het-stemlokaal/bijlage-elke-stem-telt-commissie-onderzoek-elektronisch-stemmen.pdf`.

[3] Ben Meindertsma and Bas de Vries. Duizenden stemmen meer 'verdwenen' bij verkiezingen, 2017. URL `http://nos.nl/artikel/2177871-duizenden-stemmen-meer-verdwenen-bij-verkiezingen.html`. Accessed: 2017-06-15.

[4] Stemmen vanuit buitenland blijft probleem, 2017. URL `http://nos.nl/artikel/2163145-stemmen-vanuit-buitenland-blijft-probleem.html`. Accessed: 2017-06-23.

[5] 2017. URL `https://www.nrc.nl/nieuws/2017/06/08/kiesraad-vond-partijen-met-nul-stemmen-merkwaardig-10984191-a1562169`. Accessed: 2017-06-23.

[6] 17 blockchain applications that are transforming society, 2016. URL `https://blockgeeks.com/guides/blockchain-applications/`.

[7] Jiawen Kang, Rong Yu, Xumin Huang, Sabita Maharjan, Yan Zhang, and Ekram Hossain. Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. *IEEE Transactions on Industrial Informatics*, 2017.

[8] Xiao Yue, Huiju Wang, Dawei Jin, Mingqiang Li, and Wei Jiang. Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. *Journal of medical systems*, 40(10): 218, 2016.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[10] Leden stembureau, 2016. URL `https://www.kiesraad.nl/verkiezingen/inhoud/gemeenteraden/stembureau/leden-stembureau`. Accessed: 2017-06-22.

[11] Reisdocumenten, 2008. URL `https://zoek.officielebekendmakingen.nl/kst-25764-40.html`.

[12] ICAO. Doc 9303-9: Machine readable travel documents, 2015. URL `https://www.icao.int/publications/Documents/9303_p9_cons_en.pdf`.

[13] Eric R Verheul. Remote document encryption-encrypting data for e-passport holders. *arXiv preprint arXiv:1704.05647*, 2017.

[14] ICAO. Doc 9303-11: Machine readable travel documents, 2015. URL `https://www.icao.int/publications/Documents/9303_p11_cons_en.pdf`.

[15] William Strauss and Neil Howe. Generation z. 2016.

[16] Leigh Campbell. Cool new things siri can do (like order you pizza), 2016. URL `http://www.huffingtonpost.com.au/2016/09/14/cool-new-things-siri-can-do-like-order-you-pizza_a_21472328/`. Accessed: 2017-06-22.

[17] Morris Dworkin. Recommendation for block cipher modes of operation. methods and techniques. Technical report, DTIC Document, 2001.

[18] Paul Syverson. A taxonomy of replay attacks [cryptographic protocols]. In *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings*, pages 187–191. IEEE, 1994.

[19] MultiChain. Multichain address format, 2015. URL `http://www.multichain.com/developers/address-key-format/`. Accessed: 2017-05-26.

[20] MultiChain. Address and key format, 2016. URL `http://www.multichain.com/developers/address-key-format/`. Accessed: 2017-06-25.

[21] ICAO. Machine readable travel documents part 10. Technical Report 9303, International Civil Aviation Organization, Montréal, Quebec, Canada H3C 5H7, 2015.

[22] Android. Mobile vision, 2014. URL `https://developers.google.com/vision/`.

[23] Tesseract. Tesseract ocr, 2017. URL `https://github.com/tesseract-ocr/tesseract`.

[24] Tesseract. Tesseract android tools, 2017. URL `https://code.google.com/archive/p/tesseract-android-tools/`.

[25] Android. Android developer docs, 2014. URL `https://developer.android.com/reference/android/hardware/camera2/package-summary.html`.

[26] Google. Camera2 google samples, 2017. URL `https://github.com/googlesamples/android-Camera2Basic`.

[27] Carl Colglazier. Improving the quality of the output, 2017. URL `https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality`. Accessed: 2017-05-19.

[28] Tesseract addons, 2017. URL `https://github.com/tesseract-ocr/tesseract/wiki/AddOns#community-training-projects`.

[29] Anyline. Train your tesseract. URL `http://trainyourtesseract.com`.

[30] Wikipedia. Machine-readable passport, 2017. URL `https://en.wikipedia.org/wiki/Machine-readable_passport#Nationality_codes_and_checksum_calculation`.

[31] Alan de Smet. Machine readable passport zone, 2004. URL `http://www.highprogrammer.com/alan/numbers/mrp.html`.

[32] ICAO. Doc 9303-3: Machine readable travel documents, 2015. URL `https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf`.

[33] Google Inc. Material design for android, 2014. URL `https://developer.android.com/design/material/index.html`.

[34] Google Inc. Normal and dangerous permissions. URL `https://developer.android.com/guide/topics/permissions/requesting.html#normal-dangerous`.

[35] MultiChain. Multichain permissions managment, 2017. URL `http://www.multichain.com/developers/permissions-management/`. Accessed: 2017-05-22.

[36] Gideon Greenspan. Multichain private blockchain, 2015. URL `http://www.multichain.com/download/MultiChain-White-Paper.pdf`. Accessed: 2017-05-26.

[37] CBS. Bijna 13 miljoen kiesgerechtigden op 15 maart, 2017. URL `https://www.cbs.nl/nl-nl/nieuws/2017/07/bijna-13-miljoen-kiesgerechtigden-op-15-maart`. Accessed: 2017-05-03.

[38] Paul Bakker. Asn.1 key structures in der and pem, 2014. URL `https://tls.mbed.org/kb/cryptography/asn1-key-structures-in-der-and-pem`. Accessed: 2017-06-25.

[39] Secp256k1, 2016. URL https://en.bitcoin.it/wiki/Secp256k1. Accessed: 2017-06-20.

[40] Jasper Bunskoek. Waar kun je stemmen in jouw buurt?, 2017. URL https://www.rtlnieuws.nl/buurtfacts/opmerkelijk/waar-kun-je-stemmen-in-jouw-buurt. Accessed: 2017-06-20.

[41] Arjen K Lenstra, Thorsten Kleinjung, and Emmanuel Thomé. Universal security. In *Number theory and cryptography*, pages 121–124. Springer, 2013.

[42] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C.* john wiley & sons, 2007.

[43] ISO. System and software quality models, 2011. URL https://www.iso.org/standard/35733.html. Accessed: 2017-06-25.

[44] How to implement a multichain test framework, 2017. URL http://www.multichain.com/qa/5162/how-to-implement-a-multichain-test-framework?show=5162#q5162. Accessed: 2017-06-21.

[45] Tweede Kamer der Staten-Generaal. Elektronische voorzieningen bij verkiezingen, 2016. URL https://zoek.officielebekendmakingen.nl/h-tk-20152016-83-25.html.

[46] David E. Sanger and Scott Shane. Russian hackers acted to aid trump in election, u.s. says. 2016. URL https://www.nytimes.com/2016/12/09/us/obama-russia-election-hack.html.

[47] Tweede Kamer der Staten-Generaal. Coalitie verdeeld over experiment met nieuw stembiljet, 2016. URL https://www.tweedekamer.nl/kamerstukken/plenaire_verslagen/kamer_in_het_kort/coalitie-verdeeld-over-experiment-met-nieuw. Accessed: 2017-06-15.

[48] Tijdelijke experimentenwet stembiljetten en centrale stemopneming, 2013. URL http://wetten.overheid.nl/BWBR0033598/2013-06-29. Accessed: 2017-06-15.

[49] Eerste Kamer der Staten-Generaal. Initiatiefvoorstel-taverne tijdelijke experimentenwet elektronische voorzieningen bij verkiezingen, 2016. URL https://www.eerstekamer.nl/wetsvoorstel/33354_initiatiefvoorstel_taverne. Accessed: 2017-06-15.

[50] Wolter Pieters. *Acceptance of Voting Technology: Between Confidence and Trust*, pages 283–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-34297-7. doi: 10.1007/11755593_21. URL http://dx.doi.org/10.1007/11755593_21.

[51] Menno de Jong, Joris van Hoof, and Jordy Gosselt. Voters' perceptions of voting technology. *Social Science Computer Review*, 26(4):399–410, 2008. doi: 10.1177/0894439307312482. URL http://dx.doi.org/10.1177/0894439307312482.

[52] Hugo van der Parre. Gemeenten willen graag terug naar computerstemmen, 2017. URL http://nos.nl/artikel/2165112-gemeenten-willen-graag-terug-naar-computerstemmen.html. Accessed: 2017-06-19.

[53] Kiesraad. Officiële uitslag tweede kamerverkiezing 15 maart 2017, 2017. URL https://www.kiesraad.nl/actueel/nieuws/2017/03/20/officiele-uitslag-tweede-kamerverkiezing-15-maart-2017. Accessed: 2017-06-15.

[54] Ülle Madise and Priit Vinkel. *Internet Voting in Estonia: From Constitutional Debate to Evaluation of Experience over Six Elections*, pages 53–72. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08117-5. doi: 10.1007/978-3-319-08117-5_4. URL http://dx.doi.org/10.1007/978-3-319-08117-5_4.

[55] Maarten A. Allers and Peter Kooreman. More evidence of the effects of voting technology on election outcomes. *Public Choice*, 139(1):159–170, 2009. ISSN 1573-7101. doi: 10.1007/s11127-008-9386-7. URL http://dx.doi.org/10.1007/s11127-008-9386-7.

[56] Wetboek van strafrecht artikel 447b, 1881. URL http://wetten.overheid.nl/BWBR0001854/2017-03-01#BoekDerde_TiteldeelIII. Accessed: 2017-05-02.

[57] MultiChain. Multisignature transactions. URL http://www.multichain.com/developers/multisignature-transactions/. Accessed: 2017-05-22.

[58] Rdw gestart met uitgifte nieuw rijbewijs met chip, 2014. URL https://www.verkeerspro.nl/rijschool/2014/11/18/rdw-gestart-met-uitgifte-nieuw-rijbewijs-met-chip/. Accessed: 2017-06-20.

A Original problem description

This appendix contains the original problem description as provided by the client in a contract at the start of the project.

A.1. Original Dutch

Het digitaliseren van de stempas, deze moet bruikbaar zijn op je smartphone. De Android app dient ontwikkeld te worden in Android Studio. Aan het einde van dit project verwacht zowel Milvum als de TU Delft een werkend prototype op een android smartphone met ondersteuning van API 21+. Milvum geeft de mogelijkheid om het prototype te testen op een Samsung Galaxy S7. De code wordt na oplevering open source gezet op Github. Daarnaast dient er gebruik gemaakt te worden van blockchain technologie. Deze blockchain technologie dient geschreven te worden met programmeertaal Python. Om de veiligheid van het digitale stemproces te waarborgen dient de smartphone geactiveerd te worden bij een gemeente en daarnaast dient deze smartphone een security check te doorstaan, waarbij wordt gecontroleerd of er geen kwaadaardige programma's aanwezig zijn. Dit project vormt een onderdeel van het digitaliseren van het gehele stemproces.

A.2. English translation

Digitizing the voting pass, this should be usable on a smartphone. The Android app needs to be developed using Android Studio. At the end of the project Milvum, as well as the Delft University of Technology, expects a working prototype on an android smartphone with support for API 21+. Milvum provides the opportunity to test the prototype with a Samsung Galaxy S7. The code will be made available open source on GitHub. Besides this blockchain technology needs to be used. This blockchain technology should be written in programming language Python. In order to guarantee the security of the digital voting process, the smartphone needs to be activated at a municipality and the smartphone needs to pass a security check, which checks whether or not malicious software is present. This project forms one part of digitizing the entire voting process.

B
Project plan

To ensure a smooth collaboration with all the stakeholders involved in this project, we will describe our vision about the process of this project in this appendix. Also the questions we will investigate during research and requirements of the prototype are listed here.

B.1. Client

The project is an assignment provided by the company Milvum, based in The Hague. Milvum is a software development company, focused on developing apps in corporate environments. They start in July with developing a prototype for digital voting as a pilot for several municipalities in the Netherlands. This bachelor project will cover one part of the digital voting system. The results and problems encountered during the process are of interest to Milvum in their efforts to develop a prototype for digital voting later this year. Milvum provides a work area, lunch, and two developers, who can provide access to their IT-infrastructure and share insights on minor struggles during development.

B.2. Problem description

This project forms one part of digitizing the entire voting process: the voting pass. Currently people get a paper voting pass in their mail, with this pass, in combination with identification papers, they can vote. Instead of a paper voting pass, people should be able to show their voting eligibility using their phone. This project creates a full replacement for the paper voting pass. The solution should ensure that the right people vote and that they can only vote once. Furthermore the solution needs to ensure that no external parties can influence the voting system. The device needs to get a security check to ensure no malicious software is present, and the created app has not been altered by a third party. This check and the activation of the device as a voting pass needs to be done at the town hall.
Digital voting solutions are often met with misgiving, therefore trust in the system is a key issue. The project results in a prototype android app which acts as a digital voting pass. The entire system will make use of blockchain technology.

B.3. Team members

In Table B.1 all team members are listed. Most of the production will be covered by the students from Delft University of Technology. They are required to use their skillset and expertise to work to an end product in a structured way.

B.3.1. Required skills

- Java development the Android ecosystem
- Python for the backend
- Git for version control
- LaTeX for documentation

Name	Role	Organization
Daan Middendorp	Student - Analysis and production	TU Delft
Wilko Meijer	Student - Analysis and production	TU Delft
Jonathan Raes	Student - Analysis and production	TU Delft
Rico Tubbing	Student - Analysis and production	TU Delft
Johan Pouwelse	TUD Coach	TU Delft
Salim Hadri / Arvind Jagesser	Product owner / Coach	Milvum
Michiel Hegemans	Development support	Milvum
Maarten van Beek	Development support	Milvum

Table B.1: Team members

B.4. Product requirements

Initial requirements can be found below, these requirements are derived from the project description and the first meeting with the client. More elaborate product requirements will follow from the research phase.

B.4.1. Functional requirements

- The app must keep track of the identity of the voter.
- The system must ensure that a voter does not vote more than once, either digitally, physically or both.
- The app and system must be secure.
- The app and system must be trustworthy.
- The app should allow proxy-voting to be possible.

B.4.2. Non-functional requirements

- Android app must be developed in Android Studio.
- Android app must use API 21+ (Lollipop).
- Blockchain technology must be developed in Python.

B.5. Workflow and collaboration

Below we define some agreements about the workflow and general collaboration in the project.

Meetings with coaches

- Weekly with Milvum coach
- Weekly with TU coach

Work location

- Milvum, Regulusweg 5, Den Haag

Working hours

- Normally 5 days per week 9h00-17h00

Technology supporting the workflow

- Github with Gitflow
- Waffle.io

Agile methodology characteristics

- Two-week iterations
- Daily stand-up meeting

- Meeting for iteration plan on Monday morning
- Retrospective meeting on Friday afternoon

The use of the agile methodology ensures a good overview of the state of the project. Each two-week sprint results in a working project and a chapter of the final report. Working agile creates a flexible workflow in which problems can be addressed quickly. On the first Monday of an iteration, a plan is made for how to reach the goal in the next two weeks. In the daily stand-up meetings, every group member pitches what they have done, what they are going to do, and if they encountered any problems that need to be addressed. Because of these meetings, every group member knows what the state of the project is and what his tasks are. By reflecting on the work process in the Friday retrospective meetings any issues in this regard can be dealt with swiftly so they don't hold back the project.
Gitflow and Waffle.io support the agile methodology by providing a structured way to assign tasks to group members, keep track of the state of tasks, and allow for easy collaboration between group members. Github allows for easy version control of the code, keeping track of issues with code and mapping tasks to results in code.

B.6. Code Quality

Good quality code is easier to maintain, to extent, and is less bug-prone. With trust as a key issue in this project, secure and easy-to-understand code is essential. To ensure good quality code, tests will be written for the system. Continuous integration by using Travis CI helps to detect problems with code early on. A pull-based development process with code reviews ensures understandable and qualitative code. Code submissions to SIG (Software Improvement Group) will give a good indication of the code quality compared to other systems and highlight areas where improvement is needed. The code will be written by keeping the definitions for maintainability by ISO 25010 in mind. SIG analyzes code according to these standards as well.

B.7. Research questions

From the problem description and the requirements we derived we define the following research question with sub questions that we will answer in the research report.
How can the paper voting pass in the Dutch voting system be replaced by a secure, trustworthy, and easy-to-use digital system?

- How does the current Dutch (paper) voting process work?
- What are existing digital voting systems?
- How to keep track of the identity of the voter?
- How to ensure that an individual cannot vote more than once?
- How to ensure the process cannot be manipulated by external parties?
- How to allow proxy-voting?
- How can we make the process trustworthy to skeptics?

B.8. Timeline

Deliverable	**Description**	**ETA**
Research report	Report about design decisions and an explanation why these decisions were made	8 May
Technical architecture	Architecture of all components and design choices	15 May
GUI Design	Design of the user interface made by external party	29 May
SIG upload	Code submission to SIG	Before 1 June
Prototype	Prototype showing a working digital implementation of the voting pass	19 June
SIG upload	Code submission to SIG	Before 26 June
Final report	A complete description of the whole process from begin to end according to the guidelines of the manual	26 June
End presentation	A final presentation with a description of the problem, solution, demo and afterwards questions.	Beginning of July (3-5)

Table B.2: Timeline of the project

C Research report

The research report is the concluding document of the research phase, which was conducted in the first two weeks of the project. This appendix contains the research report in its form as it was submitted after these two weeks. Since then new insights were found, so parts will be outdated.

C.1. Introduction

Currently, voting in the Netherlands is done in an old fashioned way, by paper and pencil. However, there is a strong call for digital voting systems from municipalities in the Netherlands, according to a survey by the public news organization NOS[1]. Propositions for digital voting are often met with skepticism, because people have major security concerns with digital voting [59]. The goal of this project is to implement a digital alternative for the paper voting pass, this system must make use of blockchain technology. With this digital voting pass voters must at least be able to do the same things as with the paper variant. The digital system should improve the voting process for voters in some way, so voters will have some incentive to switch to the new system. If voters have objections to the system it is highly unlikely that it will be implemented, so gaining the approval of voters for the system is a key aspect. This approval consists of two parts: usability or ease-to-use and trust that the system is secure. Since the lives of people involve many digital solutions, it is likely that in the future a cumbersome paper voting process could invoke more frustration with the elections and in the worst case even cause people to refrain from casting their vote. A digital solution should make voting easier and prevent voting from becoming old-fashioned or outdated. Furthermore since elections have major influence on the lives of people, they should trust that the system won't be defrauded. The scope of the project is limited to replacing the voting pass for two reasons: to be able to finish the project within 10 weeks and to give the possibility for the government to set at least a small step in the direction of a fully digital voting process.

During the first days of the project we determined the rough requirements of the project and from that derived the research subjects that are discussed in this report. More information about these first days can be found in the Project Plan. These first days also resulted in the following research question which this report answers. *How can the paper voting pass in the Dutch voting system be replaced by a secure, trustworthy, and easy-to-use digital system?* Multiple architectural choices are considered and substantial decisions are made, this report elaborates these decisions and concludes the research phase of the project.

C.2. Current Dutch voting system

In order to better understand the setting of the problem, this section will give a short outline of how the Dutch voting system currently works.

A few weeks before the election day municipalities send voting passes to a voter's home address. These arrive at least 14 days before the election. Every Dutch citizen of 18 years and older is eligible to vote for Dutch and European elections, with exception of people whose voting rights have been revoked by a judge. Revoking people's voting rights rarely happens, at the start of 2016 the voting rights of a total of 56 people had been revoked [60].

[1] http://nos.nl/artikel/2165112-gemeenten-willen-graag-terug-naar-computerstemmen.html

If a voter is not able to cast his vote himself, he can authorize another voter to vote on his behalf, this is called proxy-voting. There are two ways to do this:

1. For votes who live in the same municipality: by filling in the back side of the voting pass
2. For voters who live in different municipalities: by requesting a proxy-voting pass from the local municipality

A voter can only cast votes for himself and a maximum of two other people [61].
On election day, a voter brings his voting pass and his identification card to a polling station. Here, this person's identity is checked and if all is in order. The voter exchanges the voting pass for a ballot paper on which he marks his vote. At the end of election day all votes are counted by hand and the results are announced [62].

C.3. Existing digital solutions

In the past and present digital voting is practiced around the world by different governments. This section discusses some digital voting technologies and list advantages and disadvantages of these techniques.
In the first digital voting technique, voters come to a polling station and use a punch card to cast their vote [63]. The punch card has several ballot positions, where each positions represents a pre-determined candidate. A voter can use a punch device, such as a stylus, to create a hole in the card, this hole will be the vote on a candidate. A counting machine is now able to determine on which candidate, and thus also the party, has been voted. An example of such as system is Votamatic [64]. The advantage of this technique, and actually all other digital voting techniques mentioned, is that there is no need to count the vote by hand. A disadvantage is chad, chad is the chunk that is punched out of the card. If the chad is completely out of the card, the punch card is valid and a counting machine will register this vote. If this is not the case, a vote might not be registered by the counting machine and thus result in an inaccurate result [65]. In the 2000 general election a recount by hand, in Florida, was demanded by Gore (the opponent of Bush), because of chad [65].
Another technique that was used in the Netherlands, is the use of a direct-recording electronic voting system. In this system voters come to a polling station and cast a vote on a special machine [66]. Afterwards, the results of a polling station are put on a USB stick and is transported to one of the twenty locations were the results of all polling stations are counted [67]. These machines are not trusted by some people as discussed in section C.4.
In Arizona a pilot with digital voting was run [68]. In this pilot a person could register online, via Election.com, and state that he or she wants to vote online. After registration this person would receive a PIN via mail, this PIN is needed to cast this persons vote. Apart from the PIN, the person must answer two personal questions so stealing a PIN does not directly hand out a vote.
In Estonian voters can vote with their ID card. Their ID card contains a PKI (Public Key Infrastructure), this allows a person to sign a document with his ID [69]. To sign a document with an ID card, a PIN must be provided. Voters can download special software to cast their votes via the Internet. The software will ask for a persons ID and PIN, when this entered correctly a centralized server will be contacted. This server checks if this person is eligible to vote and if so return the set of candidates. The person can now cast his or her vote and the vote is send to the centralized server, where it will be stored. After this, the server returns a personal QR code. With a verification app and this QR code, a person can verify if his vote is correctly recorded. Security of this system is a controversial topic, in [69] researchers have shown that the software contains several vulnerabilities. An attacker would be able to disable voting in 75 minutes via a denial-of-service attack. Besides this, an attacker would be able to perform a shell-injection which allowed root access to centralized servers.

C.4. Trust and the old Dutch digital system

Trust is an integral component in many kinds of human interaction, and elections are no exception. Many varying definitions of trust exist, a definition applicable to this project, taken from Mui et al: "[Trust is] a subjective expectation an agent has about another's future behavior based on the history of their encounters." [70].
A voter needs to trust the system that the vote cast by him or her is actually counted and participates in the election, that no extra votes are added, and that the vote has the same value as any other vote cast by anyone else.

C.4.1. Trust issues

Some people have an instant disapproval when it comes to voting digitally [71], perhaps not without reason. In 1967 some municipalities in The Netherlands already started with using computers to vote [72], since October 2007, voting computers are no longer in use due to lobbying of the group "Wij Vertrouwen Stemcomputers Niet" [73]. Mostly due to a security analysis that showed multiple technical issues with the used machines [74]. Issues addressed by this group included that the Nedap computers were 'non-voter verifiable', meaning that voters had no way of verifying that their vote was actually counted. In the current (paper) system this can be achieved because each voter has the right to watch while his vote is counted [75]. Another problem shown by this group relating to voting computers is that there usually is one central machine involved, that, if in control by a malicious party, could manipulate the outcome of the election [71]. Opponents of the Nedap computer also had problems with the closed source implementation they had and the non-transparent voting process [59].

C.4.2. Design philosophies

Evidently, by looking at the history of electronic voting, it can be seen that providing any digital solution to voting will be greeted with a lot of skepticism. It is clear that, in order to be trusted, a digital solution to the elections needs to be as open and transparent as possible, while still remaining secure. This transparency can be achieved by using all open source technology, both in software as in hardware and should be enough to silence skeptics and make voters trust the system. Digital voting must be voter verifiable and preferably be a decentralized implementation such as a decentralized blockchain, so that a single compromised machine does not endanger the rest of the network. End-to-end integrity checks will ensure that the correctness of the data is verified and that any anomalies will be detected, so the voters can trust the results of the elections.

C.5. Proposed solution

This section gives an outline of the proposed solution and states which questions about the implementation and technologies follow from this.

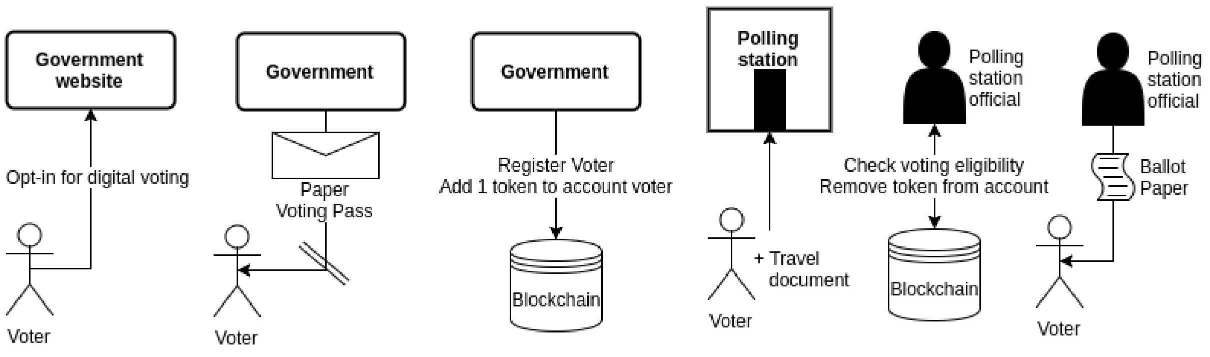


Figure C.1: Schematic rendering of the proposed system

The proposal for a solution consists of a blockchain instance that defines a vote token with a centralized owner, in Figure C.1 a visualization of the proposed voting process is given. Every person would be able to go to a government website (e.g. "MijnOverheid") and register him or herself as a digital voter. Then, this person would need to choose from a list of their identification documents that contain a contactless chip (this list is already available). There are two possible cases: Either the government knows the public key of this travel document, or not. In the second case there could be a mechanism implemented where the user must register his travel document first, this can happen remotely. At some point, for example after the digital voting registration period, the government, being the owner of the blockchain, can associate a single vote token to the accounts of registered voters.

At the time of the elections, the person would go to a voting booth and bring only the identification document that was registered. There, a voting booth official will scan his document using a NFC scanner installed on a smartphone or tablet, so the public key that corresponds to the document can be read. This app will check the chain to see if the public key still has an outstanding balance. If so, a transaction is made and signed, using the voter's document, to send the vote token to a government wallet. After this transaction is confirmed it is ensured that it was valid and a passing message would be shown on the official's device. After this, a ballot

paper can be given so the person can cast his or her vote.
From this proposal questions arise about the architecture. These questions will be discussed and answered in this paper:

1. Which blockchain technology is best suited?
2. How to keep track of the identity of a voter using traveling documents?
3. How to prevent double-spending of a voting pass?
4. How to allow proxy-voting?
5. How to ensure the devices used in the voting process are secure?

C.6. Blockchain

Key aspects of blockchain technology are its decentralized network, immutability and the transparency of everything that happens on the blockchain. The decentralized structure and its structure of linked blocks of data makes it extremely hard to alter data [76]. Transparency ensures that everyone can check exactly what happened in the voting process, which makes it very hard to defraud the system without detection [77]. This makes blockchain technology very interesting to use in digital voting implementations. Transparency in the chain creates no problems for ensuring secret ballot in this project, because the transparent transaction only shows that a vote has been cast. More specifically, it only shows that a voting pass has been exchanged for a ballot paper. This physical ballot paper is used to cast the actual vote.
This section discusses which blockchain implementation would be best to use for this project and how to map the identity of the voter to an account on the blockchain so the voting eligibility of a person can be checked.

C.6.1. Blockchain implementation

There is a variety of choices for the implementation of blockchain technology in the election process. Multiple smart contracts on a chain like Ethereum or Bitcoin using Counterparty are considered[78]. Another option is setting up a blockchain for this special purpose. Such a chain could be created using MultiChain [79] or OpenChain [80]. The blockchain as implemented in Tribler, called TrustChain, was also considered [81]. Smart contracts, which have emerged in the last two years, are computerized transaction protocols which execute terms that are written in the contract [82]. These contracts can be validated by every node. Using smart contracts on a well-known and well-reviewed chain like Ethereum, which already has a large network of miners backing it, makes a monopoly attack (51% attack) unfeasible. Ethereum smart contracts have almost the same functionality as a self-made blockchain would. Custom tokens can be generated, such as in [83], which could represent whether the owner has already claimed his voting pass or not. These can be distributed from a single authority to all vote-eligible people. A problem in this set-up is that verification of the transactions will need to happen inside the smart contract, which is an expensive operation in terms of gas, Ethereum's processing fee. Our rough estimates show that it will be a few euros per verification. Furthermore, there will have to be some sort of mapping that maps private/public key pairs from the passport to key pairs on the network, this mapping would probably have to be made outside of the Ethereum network. This is necessary because the cryptographic implementations in the passport are different than in the Ethereum or Bitcoin network. Another advantage of the Ethereum network is that it is well-known and trusted, so this choice may receive less scepticism than others.
OpenChain is more like a transaction chain than a blockchain. It has no blocks but chains all transactions together. It can be anchored to the Bitcoin blockchain to benefit from the irreversibility of its Proof of Work [84]. The downside of OpenChain is that it relies on a client-server model where there is only 1 server node that verifies transactions [80]. Even though integrity can still be proven this opens up an attack surface for DDoS attacks. It this validator node is brought down, no transactions can be verified anymore and the network is halted. Since OpenChain uses a client-server model, the Java integration would be as easy as speaking to some endpoints in HTTP.
MultiChain is a toolset to easily setup a personalized blockchain, based on the Bitcoin core, that can be as private or public as needed. It features a permission system where nodes can be allowed or disallowed to do things like mining or even sending transactions. This is necessary to ensure only the government can be a mining node, otherwise a third party could supply so much processing power that it can do a monopoly attack on the network. MultiChain relies on nodes that mine in order to verify transactions [79]. This can be

multiple nodes so the network is strong against DDoS attacks. MultiChain has a Java API library, but it does not seem to be finished or have very much support [85].
The advantage of using a personalized blockchain such as MultiChain or OpenChain is that those projects can be forked. Therefore the transaction signing implementation can be altered, so that it uses the same cryptographic keys as the ePassport, eliminating the need for a mapper. This can be changed so ePassport public keys are actual wallet addresses and a transaction signed by the document could be broadcast to the network immediately. This offers great simplification compared to a setup using smart contracts.
When using existing running blockchain platforms, transaction fees apply. This includes running smart contracts on Ethereum which costs gas [86], and on the Bitcoin chain using counterparty, which costs regular transaction fees [78]. These transaction fees can be seen as paying for electricity. When the government hosts its own blockchain they would have to pay for their own electricity and hardware to keep the network online. It would be hard to analyze these costs beforehand, but since running a node or mining can be seen as the conversion of electricity to cryptocurrency [87], the costs of running a capable blockchain or paying for the smart contract are in the same ballpark. Therefore, these transaction fees play no significant role in deciding on a blockchain implementation. However, maintaining a specialized blockchain would require buying the hardware in the first place, which is costly, and additionally it would require a lot of extra processing power just to protect the network against monopoly attack.
The Tribler Multichain, also called TrustChain, is a blockchain developed by S.D. Norberhuis in a master thesis for the Delft University of Technology. It features (near) instant verification times, each transaction is immediately put in a block in the chain. In TrustChain, nodes do not have a full copy of the chain, only of those blocks where they were one of the parties that made a transaction. Since most of the transactions will be with one party, the government, there won't be a complex network of chains so it defaults to a regular blockchain. The added benefit for voters of not having to store the entire chain is lost, since voters probably won't run their own nodes, because of issues with uptime, and the government will run the nodes for them. Crawlers can be used to verify the integrity of the chain. Using this blockchain would require taking it out of the Tribler codebase and adapting it to the needs of this project. This is considerably more effort than a solution like MultiChain or Smart contracts. In Table C.1 an overview can be found of some of the most important features that were considered for choosing a blockchain implementation.

Feature	**MultiChain**	**TrustChain**	**Smart Contract**	**OpenChain**
Confirmation times	Configurable, minimum 2 seconds	Instant	Depends on blockchain (ETH ~15sec, BTC ~10min)	Instant
Robustness / Security	Strong: decentralized, multiple government nodes	Strong: Decentralized, multiple nodes	Strong: existing network with many nodes	Questionable, single server node that does verification
Cost	Moderate, multiple verify nodes needed to protect against DDoS	Moderate, government also hosts nodes of voters	Expensive transaction fees	Cheap, single server node for verification
Language and ease of Java integration	C++ with a Java API lib	Python, in production currently	Solidity, with easy Java integration	C# with easy Java integration
Documentation	Moderate	Scarce	Lots	Moderate

Table C.1: Comparison of important features

C.6.2. Conclusion
Due to the fast reforming landscape of blockchain technologies, a right decision made today, might be completely fallacious within a short period of time. This makes it difficult to make choices based on the current available information. Is the platform still actively maintained within one year?
Based on this, the preferred choice goes to the MultiChain.com platform, because of the decentralized structure, the available API's and documentation. If this approach turns out to be suboptimal, then we might

continue the project with the OpenChain platform which is not decentralized, but seems to be much easier to configure. This platform will be modified to support transaction signatures with dutch ePassports.

C.6.3. Blockchain-identity

A blockchain can be seen as a database where every move can be originated to a user. Due to this, there needs to be a solid identity which handles the access control and signs every transaction. In most cases, blockchain accounts consist out of a private and public key pair, which can be used to send and receive transactions. There is a wide range of options to store these key pairs. Bitcoin for example uses software keys which are stored in the client. This makes it vulnerable for attackers to confiscate these keys and perform unauthorized transactions. Because of this, it is desirable to store the key pair in a more secure environment.

Biometric passport key pair

Identification using PKI is something which has been implemented by several governments. For example the Belgian National Identity Card, which is introduced in 2005, contains a PKI by default [88]. This is used for signing electronic documents and logging in to several governmental services, like tax returns. There is not much criticism about the security grade of this implementation, but one of the major drawbacks of this implementation is that a specific card reader is required for these cards, which almost no citizens owned. This is the reason why it is not widely used [88].
Dutch driver's licences, identity cards and passports are designed according to the ePassport (or biometric passport) ICAO standard [89]. This means that there is an NFC-chip implemented in each of these documents which can be read with freely accessible hardware. The rise of the NFC applications in all kinds of industries, from banking to travelling has brought us to point where almost everyone has access to a reader [90].
The ICAO standard has described several obligatory and optional specifications. One of these specifications is the implementation of a so called AA (Active Authentication). This implementation consists of a private key stored into the chip which cannot be copied, a readable public key, and a signing function. This makes it possible to sign data using the private key in the chip [91].
The chip in Dutch travel documents is manufactured by a French company and is not implemented exactly the same as other European Union passports. Due to this, AA was not implemented with RSA keys, like supported by most ePassport reading libraries. After some research and different experiments it turned out that Dutch travel documents are based on ECDSA (Elliptic Curve Digital Signature Algorithm) combined with a BRAINPOOLP320r1 curve. Elliptic Curves are fortunately really common in blockchain. This makes it easier to interchange those signing mechanisms.
One major issue in this system is that this signing function is only able to sign eight bytes. A typical Bitcoin transaction has a size of 200-250 bytes [92]. Due to this, it is not possible to sign a complete transaction or a SHA-256 hash, which results in a 32 bytes hash, of it. There are also other hashing algorithms available which create smaller hashes, but they are not recommended for cryptographic usage, because of the higher risk of duplicates. To still make it possible to use the travel document for signing transactions, the signing and verifying algorithms will be modified so that they will process partial signatures of the hash. This is less secure than signing the entire hash. The next ICOA travel document standard should contain a signing method for 32 bytes to make it more secure.
Other papers discourage the use of the passport as a digital smartcard, because of the lack of any additional authentication such as a PIN [93]. This is not completely true, to have any interaction with the passport, there needs to be a Basic Access Control performed. This consist of sending a part of the MRZ (Machine Readable Zone) which is printed in the passport to the chip, so that it is guaranteed that the person accessing the data on the passport also has physical access to it.

Alternative identity solutions

If transaction signing using the ePassport proves to difficult to implement, another technique to securely sign transactions is needed. There needs to be another identity vehicle with the following specifications:

- Basic Access Control. This protects the communication between the chip and reader using encryption. Before data can be read from the chip, a key needs to be provided. In the ePassport this key consists of the date of birth, the date of expiry and the document number [91].

- A signing function. For signing data with the card itself using a private key stored inside the card. This private key is not readable from the card itself, it can only be used to sign data with the card which can then be verified. This ensures that the private key can not be stolen unless the entire document is stolen.

There are different physical smart cards available which are suitable for this purpose. A major drawback of this decision is that it is expensive and complex to provide these smart cards to the voters.
Instead of a smart card, a smartphone could also be used to sign transactions. Most devices contain a so called secure element in their hardware. This secure element makes it, like the ePassport solution, possible to sign documents using the inaccessible chip. However, the API of this secure element is not available for public usage. The app needs system level permissions to access this chip [94].
Another option is to store the key in memory of the app. In this scenario a voter would receive or generate a private key which is stored on the person's device. This private key is then used for signing transactions in the blockchain. However, a private key on a device will pose some security problems, as the private key can be stolen with malware for example. If the private key is compromised, an attacker can take away the individual's right to vote by signing away the vote token. Since the government has only distributed the tokens to addresses it knows belong to people that have signed up online and so will not receive a physical ballot paper, this attack can not add votes for a specific party. The attacker could only take ballot papers (tokens) away from voters and possibly use them himself to get extra votes for himself.

C.7. Double voting

Double voting is something that should obviously not be possible in any election system. The digital voting-pass solution should not allow this. Since it is initially assumed that the digital voting pass can be used in parallel with the paper one, it needs to be made sure that a person can not cast a vote using both systems. Obviously the digital system itself should also not allow for multiple votes to be cast by a single person (unless that person is warranted to vote for another).
If a blockchain transaction is made to resemble a person having cast his or her vote, this problem is analogous to the double spending problem. The transaction in question must first be verified in order to be sure that the voter is not trying to double-vote. Only after it is verified that the person had not already cast a vote, a ballot paper can be handed out. This verification time should be considered when choosing a blockchain platform, because waiting a long time at the voting booth is very impractical.
This waiting time is a direct result of the blockchain's choices for block generation time. Since these shorter confirmation times also allow for faster transactions in crypto-currencies which is desirable, one can expect that these delays will only decrease over time. Furthermore there are other blockchain technologies which make use of (near) instant verification, so verification time can be considered as a non-issue.

C.8. Proxy-voting

Because the digital voting pass should be a complete replacement of the paper voting pass, proxy-voting should be possible with the digital voting pass. A short description of proxy-voting can be found in section C.2.

C.8.1. Problems and constraints

Dutch law and the characteristics of proxy-voting raises certain problems that the system has to deal with.
A big issue with proxy-voting is checking if someone freely approved of giving someone else authority to vote on his behalf. There are two constraints against forcing proxy-voting in the current system that are important to note:

- A signed voting pass and a copy of an ID card are required [95]
- It is illegal to force someone to authorize another person to vote for him and to actively recruit for proxy-voting [96]

The digital system should at least give the same protection as the current system, and should strive to give better protection.
Death of voters poses another issue. When a voter dies, his voting pass is rendered invalid [97]. The unique identifier of his voting pass is then put on a list of invalid voting passes and no one will be able to cast a vote with his voting pass. The digital solution should propose a way to render votes invalid, even when they are transferred to another person. If the proxy-voter dies the proxy-votes should be returned to their original owner.

In the current system a warranter can revoke the proxy-voter's right to vote on his behalf if the proxy-voting is done by handing over a voting pass. If the proxy-voting is done via a written request at the local municipality, this is not possible. The digital system should provide a way to revoke the proxy-voting rights.

C.8.2. Possible solutions

When creating a cryptocurrency, either an entire new network or a cryptocurrency based an existing blockchain like Ethereum, it is possible to set certain restraints on the network. In the system it can be controlled how many coins are handed out, but also some transaction constraints like a maximum wallet size, which in turn can limit the amount of proxy-votes a person can cast. Disabling authorization of proxy-voting a few days before election day makes it easier to detect if votes were somehow stolen. Proxy-voting for someone in another area with local elections won't be possible, since each local election would correspond to a different system. This is also not possible in the current voting system, because by law a proxy-voter is required to cast his own vote at the same time as the proxy-vote, which wouldn't be possible if the votes are for different local elections.

C.8.3. Proposed systems

There are multiple ways to implement a voting pass solution in a current blockchain technologies. In this section the pros and cons of three approaches will be compared.

Proxy-voting via regular transactions
The simplest system for proxy-voting is regular transactions of cryptocurrencies.
In cryptocurrencies like Bitcoin and Ethereum the basics of the transaction of a 'coin' or token come down to the following. A token is transferred from one owner to another. A transaction uses the previous transaction of the token and the public key of the receiving party in a hash. The sender signs this hash with his private key. This allows the system to check if the sending party is actually the owner of the token. Because a hash of the previous transaction of the token is included in the new transaction, the token can be tracked [87, 98].
This transaction system can be used for proxy-voting. Let us say someone we will call Voter 1 (V_1) would like to authorize a friend, Voter 2 (V_2), to vote on his behalf. He would make a simple transaction to move his voting token to the wallet of V_2. All he would need for this is the public key of V_2 and the transfer can be made. This system, however, has quite a few weaknesses.

- A person has no say in whether or not he becomes a proxy-voter, he could however decide to just not use the voting power.
- Because a transaction is a very simple and fast operation, there is not much time to think about the decision. This also leads to a higher risk of people authorizing proxy-voters under pressure.
- A person should verify manually if he submitted the right target address for the token, which could lead to errors and sending voting tokens to the wrong person.
- The token of V_1, which was sent to proxy-voter V_2, could be sent to Voter 3 (V_3). This leads to a situation where V_3 can cast a vote on V_1's behalf without his consent.
- There is no possibility to revoke the rights to proxy-vote, once a transaction is sent and accepted the original sender has no influence over the token.

The current paper system does deal with these problems and thus this digital solution can't be used as a proper replacement.

Proxy-voting with smart contracts or modified transaction functions
The way transactions work can be influenced or modified to enable the use of many more rules and constraints on transactions. This can either be done via smart contracts like the ones Ethereum uses, by modifying the transaction function, or adding specialized transaction functions to a blockchain implementation like MultiChain.
Revoking proxy-voting rights - Let's say Voter 1 V_1 enters into a proxy-voting contract with V_2. This proxy-voting contract says that V_2 gets the voting token of V_1 a certain time before election day if V_1 doesn't submit some kind of objection (this could be implemented in many ways). Alternatively V_1's token can be transferred to V_2 at the start of the contract and can be transferred back when the contract is voided.

Additional verification - To prevent giving proxy-voting rights to the wrong person or V_1 giving proxy-voting rights to V_2 without consent, smart contracts can be used to implement a verification system. In this system V_2 should perform a verification action to confirm that he accepts to act as a proxy-voter, after that V_1 should confirm that he actually wants to give his voting rights to V_2.
Resending a token - To prevent V_2 from sending V_1's token to V_3, a simple addition to the contract concerning transactions is needed. This contract should check whether the sender of a token is the original owner, if this is not the case the token shouldn't be sent.
To make it easier for the user to set up proxy-voting an app should be developed, which calls the appropriate functions or creates a smart contract. This could create a vulnerability in the system when the device of a user is compromised, this could allow an attacker create a fake app which executes modified smart contracts. Some additional thought should be put into how to protect voters from this vulnerability. The upside is, that these transactions can still be traced on the blockchain and stolen tokens can be rendered invalid, just like it would happen if a paper voting pass is stolen. Thus, the integrity of the voting process wouldn't be compromised.

Voting with someone else's wallet

Another way to allow proxy-voting is to allow people to vote with the actual wallet of the warranter. This would need to be done with some kind of written statement that a person is authorized to proxy-vote. This system would mean no improvement in comparison to the current proxy-voting system. Also, depending on the implementation of the wallet/identity verification this could run into legislative issues. For example, if a wallet is recognized based on ID-card verification, as a proxy-voter, a person would need to bring the ID-card of the warranter with him, which is inconvenient for the warranter and also illegal [99]. A situation where a person can borrow the identifying markers of another person, and can basically act like he is another person, is not the right path to go down from a legal point of view.

C.8.4. Best system

Three options for implementations to allow for proxy-voting in digital voting passes were proposed. The third option, voting with someone else's wallet, is least applicable of the three choices, because of legislative and fraud issues. There would be virtually no trustworthy check to see if the warranter has actually allowed the proxy-voter to vote or if the proxy-voter just stole or copied the needed accessories for verification. This would cause people to distrust the system and therefore it would be hard to get this system widely accepted. Using regular transactions wouldn't be a proper replacement of the paper voting pass, cause of the mentioned weaknesses and problems. The option with specialized transaction functions or smart contracts expands on the regular transactions and by doing that, tackles the main issues with that option. The system does get more complex and thus has higher development cost, but since trust and precision are very important factors in a digital voting solution, this increase in development costs will be worth it. The added verification for proxy-voting and added simple revoking rights make this solution much better than regular transactions and the increased ease of use makes this an improvement to the current paper proxy-voting system. Voters don't need to bring anything else, but their travel document. This option is the way to go for the digital voting pass system.

C.9. Device security

One of the arguments against digital voting is that the devices that are used, could be tampered with. To overcome this issue, a device that is used in the voting process must be secure. Two techniques that can be used to secure a device and app are anti-tampering and obfuscation. Besides this, static analysis tools can be used to improve security.

C.9.1. Anti-tampering

Anti-tampering techniques can be used to ensure that the app itself has not been altered. Three methods that can be used are [100]:

- Verifying app sign at runtime
- Verifying the installer
- Environment checks

The first method, verifying the app sign at runtime, uses the fact that each Android app from the PlayStore should be signed with the developers private key [101]. Verifying the installer is a method that retrieves the name of the installer (such as com.android.example.app) from Android and checks this with a hardcoded value. The last method is mainly used to test if a debugger is attached to the app.
SafetyNet is an API from Android which provides access to Google services. With these services it is possible to determine the safety of a device [102]. This API can thus be used to ensure that device itself has not been tampered with.

C.9.2. Obfuscation

Since all software created in this project is open-source, reverse engineering is not needed to comprehend how the app works. This means that code obfuscation techniques are useless, but if a decision is made to switch to a closed-source application, code obfuscation is a useful technique. Two well-known obfuscators for Android and Java are DexGuard[2] and SafeGaurd[3].

C.9.3. Tools

The use of security static analysis tools can improve the security of application. These static analysis tools can find for example faults where `java.util.Random` is used instead of the more secure `java.security.SecureRandom`, or hardcoding private keys in the source code. Examples of tools that can be used are Find Security Bugs and QARK.

C.10. Glossary

Some jargon used in the Netherlands to describe items used in the voting process do not have a well-defined English translation. This short glossary defines the translations used in this report.

- ballot paper - stembiljet
- proxy-voter - gemachtigde
- proxy-voting - volmachtstemmen
- secret ballot - stemgeheim
- voting pass - stempas
- warranter - volmachtgever

[2] https://www.guardsquare.com/en/dexguard
[3] https://www.guardsquare.com/en/proguard

Research report bibliography

[59] Tweede Kamer. 20ste vergadering 2e kamer. URL http://wijvertrouwenstemcomputersniet.nl/images/b/be/HAN8061A02.pdf. Accessed: 2017-05-02.

[60] CBS. Bijna 13 miljoen kiesgerechtigden op 15 maart, 2017. URL https://www.cbs.nl/nl-nl/nieuws/2017/07/bijna-13-miljoen-kiesgerechtigden-op-15-maart. Accessed: 2017-05-03.

[61] Rijksoverheid. Kan ik iemand machtigen om voor mij te stemmen bij verkiezingen? URL https://www.rijksoverheid.nl/onderwerpen/verkiezingen/vraag-en-antwoord/iemand-machtigen-om-te-stemmen-bij-verkiezingen. Accessed: 2017-05-03.

[62] Rijksoverheid. Hoe kan ik stemmen bij verkiezingen? URL https://www.rijksoverheid.nl/onderwerpen/verkiezingen/vraag-en-antwoord/hoe-en-waar-kan-ik-stemmen-bij-verkiezingen. Accessed: 2017-05-03.

[63] ACE. Punch cards. URL https://aceproject.org/ace-en/topics/et/eth/eth02/eth02b/eth02b1. Accessed: 2017-05-09.

[64] Verified Voting. Votamatic. URL https://www.verifiedvoting.org/resources/voting-equipment/ess/votamatic/. Accessed: 2017-05-09.

[65] Carter M. Yang. Presidency hinges on tiny bits of paper. URL https://cseweb.ucsd.edu/~goguen/courses/275f00/abc-chads.html. Accessed: 2017-05-10.

[66] Kiesraad. Rood potlood en elektronisch stemmen, . URL https://www.kiesraad.nl/verkiezingen/inhoud/tweede-kamer/stemmen/rood-potlood-en-elektronisch-stemmen. Accessed: 2017-05-10.

[67] RTL. Zo werkt het softwaresysteem dat onze stemmen telt. URL https://www.rtlnieuws.nl/nederland/politiek/zo-werkt-het-softwaresysteem-dat-onze-stemmen-telt. Accessed: 2017-05-10.

[68] R. Gibson. Elections online: Assessing internet voting in light of the arizona democratic primary. *Political Science Quarterly*, 116(4):561–583, 2001. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0348238547&partnerID=40&md5=fd266dc88d06b5450e0a0a6d57f8ca46.

[69] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J.A. Halderman. Security analysis of the estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 703–715, 2014. doi: 10.1145/2660267.2660315. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84910610575&doi=10.1145%2f2660267.2660315&partnerID=40&md5=f4325abbf47127701145b5df2409bc1f.

[70] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.7256&rep=rep1&type=pdf. Accessed: 2017-05-02.

[71] Foundation wij vertrouwen stem computers niet, 2011. URL http://wijvertrouwenstemcomputersniet.nl/. Accessed: 2017-05-02.

[72] Derk Stokmans. Overheid was naïef met stemcomputer. *NRC Handelsblad*, page 2, 18 April 2007.

[73] Redactie webwereld. Stemcomputers afgeschaft, actiegroep blij, 2007. URL http://webwereld.nl/overheid/35564-stemcomputers-afgeschaft--actiegroep-blij. Accessed: 2017-05-02.

[74] Rop Gonggrijp and Willem-Jan Hengeveld. Nedap/groenendaal es3b voting computer, a security analysis, 2006. URL http://wijvertrouwenstemcomputersniet.nl/images/9/91/Es3b-en.pdf. Accessed: 2017-05-02.

[75] Kiesraad, . URL https://www.kiesraad.nl/verkiezingen/inhoud/tweede-kamer/uitslagen/stemmen-tellen. Accessed: 2017-05-02.

[76] 51% attack. URL https://learncryptography.com/cryptocurrency/51-attack. Accessed: 2017-05-19.

[77] Yuanfeng Cai and Dan Zhu. Fraud detections for online businesses: a perspective from blockchain technology. *Financial Innovation*, 2(1):20, 2016.

[78] Counterparty. URL https://counterparty.io/. Accessed: 2017-05-03.

[79] MultiChain. Multichain. URL http://www.multichain.com/. Accessed: 2017-05-09.

[80] OpenChain. Openchain, . URL https://www.openchain.org/. Accessed: 2017-05-09.

[81] Tribler. Multichain. URL https://github.com/Tribler/tribler/wiki/Multichain-specifications. Accessed: 2017-05-06.

[82] Alex Norta. *Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations*, pages 3–17. Springer International Publishing, Cham, 2015. ISBN 978-3-319-21915-8. doi: 10.1007/978-3-319-21915-8_1. URL http://dx.doi.org/10.1007/978-3-319-21915-8_1.

[83] Ethereum. Creating a cryptocurrency contract in ethereum. URL https://www.ethereum.org/token. Accessed: 2017-05-02.

[84] OpenChain. Anchoring and ledger integrity, . URL https://docs.openchain.org/en/latest/general/anchoring.html#anchoring. Accessed: 2017-05-09.

[85] Hubert Marteau. Multichainjavaapi. URL https://github.com/SimplyUb/MultiChainJavaAPI. Accessed: 2017-05-09.

[86] Ethereum devs. A next-generation smart contract and decentralized application platform, 2014. URL https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 2017-05-09.

[87] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2014. URL http://gavwood.com/paper.pdf.

[88] Danny De Cock, Christopher Wolf, and Bart Preneel. The belgian electronic identity card (overview). In *Sicherheit*, volume 77, pages 298–301, 2006.

[89] RVIG. Kenmerkenfolder reisdocumenten 2011. URL https://www.rvig.nl/binaries/rvig/documenten/brochures/2011/11/02/kenmerkenfolder-2011/kenmerkenfolder-2011.pdf. Accessed: 2017-05-10.

[90] Statista. Forecast installed base of nfc-enabled phones worldwide from 2013 to 2018. URL https://www.statista.com/statistics/347315/nfc-enabled-phone-installed-base/. Accessed: 2017-05-10.

[91] ICAO. Machine readable travel documents, 2015. URL http://www.icao.int/publications/Documents/9303_p9_cons_en.pdf.

[92] BitFury Group. Block size increase, 2015. URL https://bravenewcoin.com/assets/Whitepapers/block-size-1.1.1.pdf.

[93] Zdeněk Říha, Václav Matyáš, and Petr Švenda. Electronic passports. *Sborník příspěvkŭ*, page 5, 2008.

[94] Nikolay Elenkov. Accessing the embedded secure element in android 4.x. URL https://nelenkov.blogspot.nl/2012/08/accessing-embedded-secure-element-in.html. Accessed: 2017-05-08.

[95] Kieswet artikel l 14, 1989. URL http://wetten.overheid.nl/BWBR0004627/2017-04-01#AfdelingII_HoofdstukL_Paragraaf3. Accessed: 2017-05-02.

[96] Kieswet artikel z 4&8, 1989. URL http://wetten.overheid.nl/BWBR0004627/2017-04-01#AfdelingVI. Accessed: 2017-05-02.

[97] Kieswet artikel j 7a, 1989. URL http://wetten.overheid.nl/BWBR0004627/2017-04-01#AfdelingII_HoofdstukJ_Paragraaf2. Accessed: 2017-05-02.

[98] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL https://www.bitcoin.org/bitcoin.pdf.

[99] Wetboek van strafrecht artikel 447b, 1881. URL http://wetten.overheid.nl/BWBR0001854/2017-03-01#BoekDerde_TiteldeelIII. Accessed: 2017-05-02.

[100] Scott Alexander-Bown. Android security: Adding tampering detection to your app. URL https://www.airpair.com/android/posts/adding-tampering-detection-to-your-android-app. Accessed: 2017-05-02.

[101] Android. Sign your app. URL https://developer.android.com/studio/publish/app-signing.html. Accessed: 2017-05-02.

[102] Android. Safetynet. URL https://developers.google.com/android/reference/com/google/android/gms/safetynet/SafetyNet. Accessed: 2017-05-02.

D
Requirements

In this appendix the requirements of the proposed solution are discussed. The proposed solution is split up into three components: blockchain, polling station app and voter helper app. Each component has its own requirements. All requirements are prioritized using the MoSCoW method[1].

D.1. Blockchain

For the blockchain implementation the following requirements are established.

Must haves

1. The blockchain must verify transactions with the brainpoolP320r1 elliptic curve.
2. A voter must be able to claim his vote right only one time and no more, either with a digital voting pass or physical voting pass
3. The government must be in control of the blockchain
   (a) The government must be able to transfer votes to wallets of eligible individuals
   (b) The government must be able to control which nodes are able mine
4. The blockchain must accept a transaction from an address, when this address has sufficient funds and the transaction was signed by a passport
5. The blockchain technology must be decentralized

Should haves

6. A voter should be able to transfer a vote to a different voter who is eligible to vote (proxy-vote)
7. The government should be able to freeze accounts
8. A transaction should be contained in a block (confirmed) within twenty seconds

Could haves

9. A voter could be able to use all types of identification documents without specifying which one this person is voting with

D.2. Polling station app

For the polling station app the following requirements are established.

[1] See `https://www.interaction-design.org/literature/article/making-your-ux-life-easier-with-the-moscow`, for more information

Must haves

1. The app must have an option to manually input the data from the machine readable zone
2. The app must be able to read the public key from a travel document using NFC
3. The app must be able to show the balance of voting tokens of a voter
4. The app must be able to build an unsigned transaction
5. The app must be able to sign an unsigned transaction using a travel document
6. The app must be able to broadcast a signed transaction to the blockchain
7. The app must be able to display that a broadcasted transaction is confirmed (thus contained in a block)
8. The app must work on Android devices running Android API level 21 and higher

Should haves

9. The app should be able to show a list of transactions made with the scanned passport
10. The app should be able to use OCR to read the ePassport's MRZ
11. A polling station official should be able to choose an election from a list of elections

Could haves

12. The app could have manual control for flash and focus in OCR scanning mode

D.3. Voter helper app

For the voter helper app the following requirements are established.

Must haves

1. A voter must be able to see his or her current balance of voting tokens in the app
2. A voter must be able to transfer a voting token to another voter (proxy-voting) with the app
3. The app must be able to display the transaction history of a voter

Should haves

4. A voter must be able reclaim voting token from proxy-voting with the app

Could haves

5. The app could be able to verify if the chip in a travel document is working correctly
6. The app could have an explanation of how the digital voting process works
7. The app could have information about the voter turnout
8. The app could have information about the nearest voting stations of a voter

Won't haves

9. The app won't have an option to register for digital voting

E User Experience design

In order to create a smooth polling station app experience, the help of a user experience designer was offered by the client. The following user experience design emerged from a collaboration with Angelo Croes (angelo@milvum.com), UX designer at Milvum.

E.1. Polling station app - iteration 1

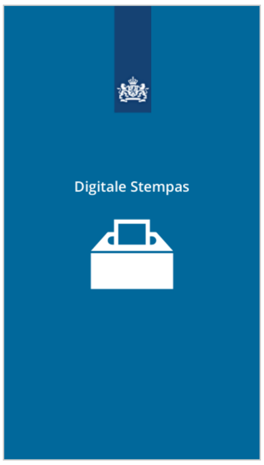
The first iteration was based on some hand drawn sketches of the workflow of the application. The designer translated this to a clickable demo, which can be found in Figure E.1.

E.2. Polling station app - iteration 2

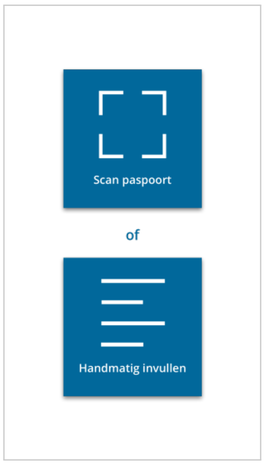
After some feedback provided was provided to the designer, e.g. the use of Material Design and the process bar while reading the NFC chip. An updated design was delivered, which can be found in Figure E.2.

E.3. Polling station app - final looks

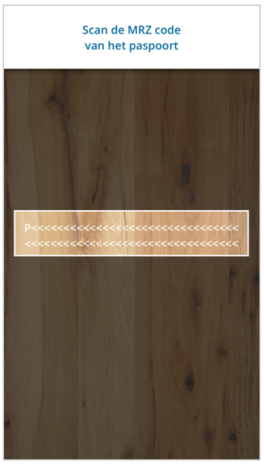
Finally, the design was translated to an Android application by building the activities. Screenshots of the final application can be found in Figure E.3.



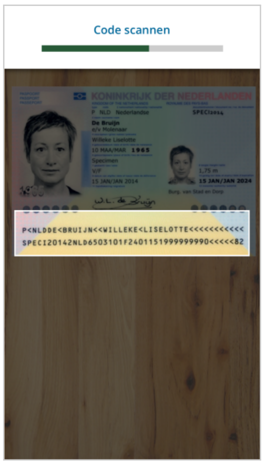
(a) Splash screen



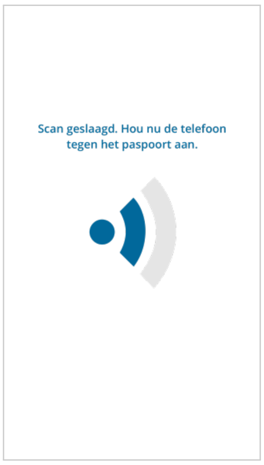
(b) Main menu



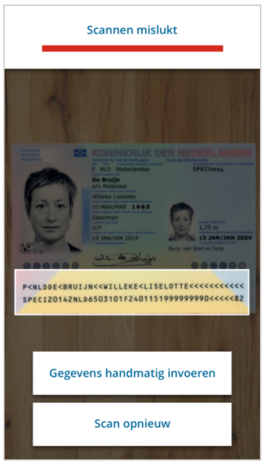
(c) OCR I



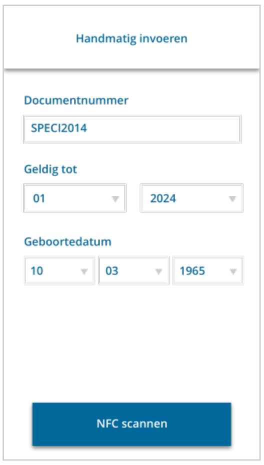
(d) OCR II



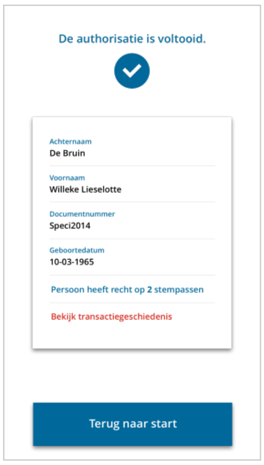
(e) NFC



(f) OCR error



(g) Manual input



Achternaam
De Bruin

Voornaam
Willeke Lieselotte

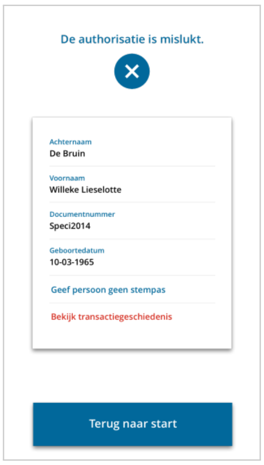
Documentnummer
Speci2014

Geboortedatum
10-03-1965

Persoon heeft recht op 2 stempassen

Bekijk transactiegeschiedenis

(h) Authorized



Achternaam
De Bruin

Voornaam
Willeke Lieselotte

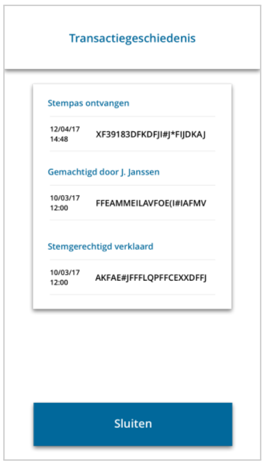
Documentnummer
Speci2014

Geboortedatum
10-03-1965

Geef persoon geen stempas

Bekijk transactiegeschiedenis

(i) Not authorized



Stempas ontvangen

12/04/17 14:48 XF39183DFKDFJI#J*FIJDKAJ

Gemachtigd door J. Janssen

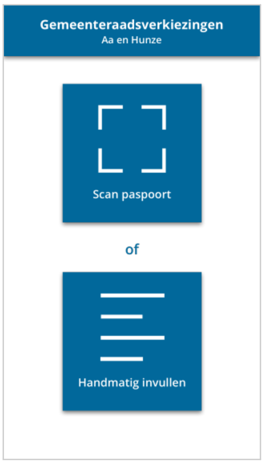
10/03/17 12:00 FFEAMMEILAVFOE(I#IAFMV

Stemgerechtigd verklaard

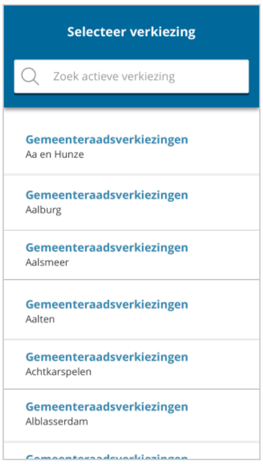
10/03/17 12:00 AKFAE#JFFFLQPFFCEXXDFFJ

(j) Transaction history

Figure E.1: Iteration one


Gemeenteraadsverkiezingen
Aa en Hunze

(a) Main menu


Selecteer verkiezing

Gemeenteraadsverkiezingen
Aa en Hunze

Gemeenteraadsverkiezingen
Aalburg

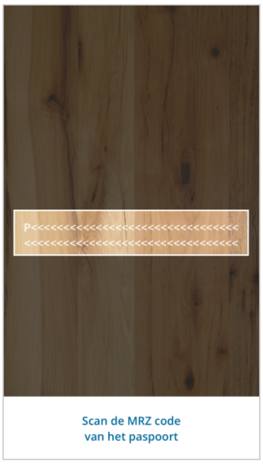
Gemeenteraadsverkiezingen
Aalsmeer

Gemeenteraadsverkiezingen
Aalten

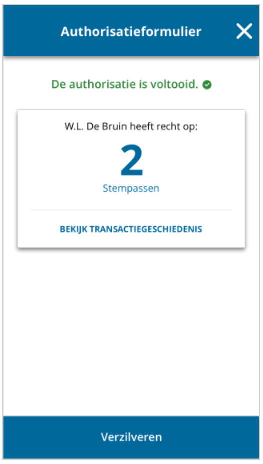
Gemeenteraadsverkiezingen
Achtkarspelen

Gemeenteraadsverkiezingen
Alblasserdam

(b) Choose election



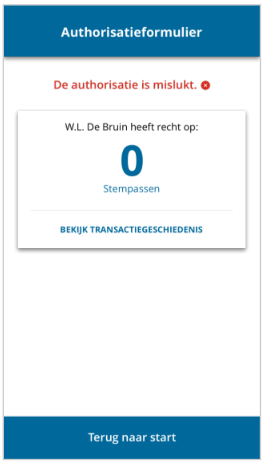
(c) OCR


Authorisatieformulier

De authorisatie is voltooid.

W.L. De Bruin heeft recht op:
2
Stempassen

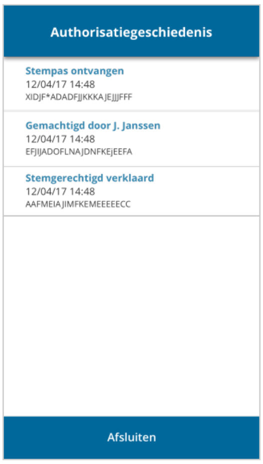
(d) Authorization success


Authorisatieformulier

De authorisatie is mislukt.

W.L. De Bruin heeft recht op:
0
Stempassen

(e) No voting passes


Authorisatiegeschiedenis

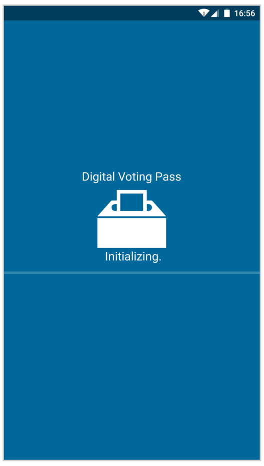
Stempas ontvangen
12/04/17 14:48
XIDJF*ADADFJJKKKAJEJJJFFF

Gemachtigd door J. Janssen
12/04/17 14:48
EFJIJADOFLNAJDNFKEjEEFA

Stemgerechtigd verklaard
12/04/17 14:48
AAFMEIAJIMFKEMEEEEECC

(f) Transaction history

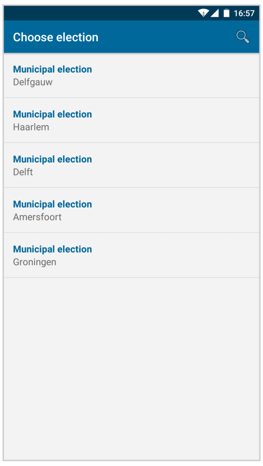
Figure E.2: Iteration two


16:56

Digital Voting Pass

Initializing.

(a) Splash screen


16:57

Choose election

Municipal election
Delfgauw

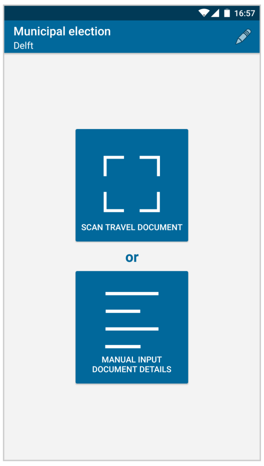
Municipal election
Haarlem

Municipal election
Delft

Municipal election
Amersfoort

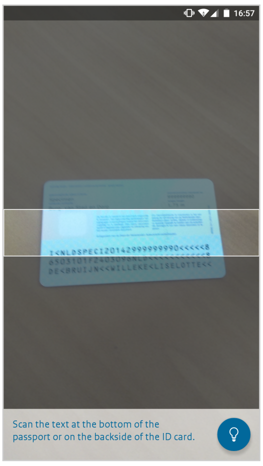
Municipal election
Groningen

(b) Choose election


16:57

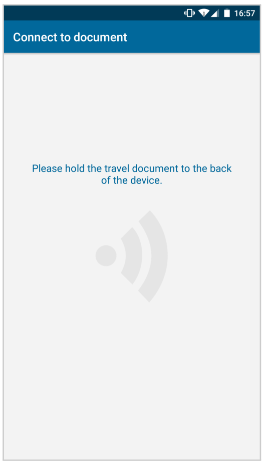
Municipal election
Delft

(c) Main menu


16:57

Scan the text at the bottom of the passport or on the backside of the ID card.

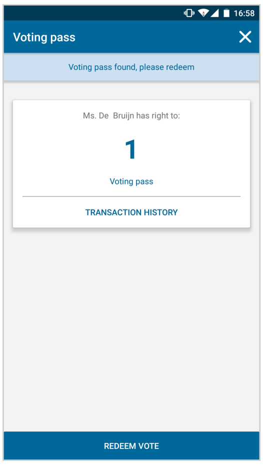
(d) OCR


16:57

Connect to document

Please hold the travel document to the back of the device.

(e) NFC


16:58

Voting pass

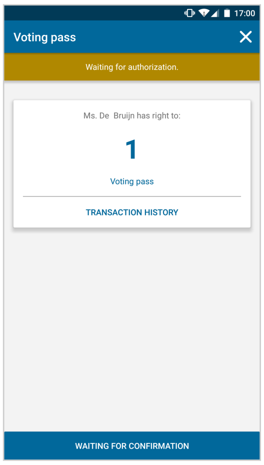
Voting pass found, please redeem

Ms. De Bruijn has right to:

1

Voting pass

(f) Authorization succes


17:00

Voting pass

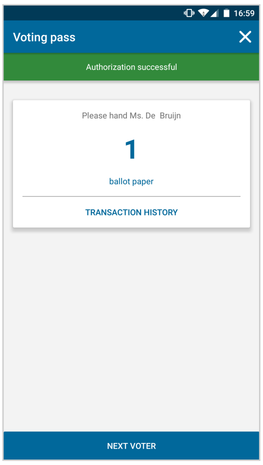
Waiting for authorization.

Ms. De Bruijn has right to:

1

Voting pass

(g) Waiting for confirmation


16:59

Voting pass

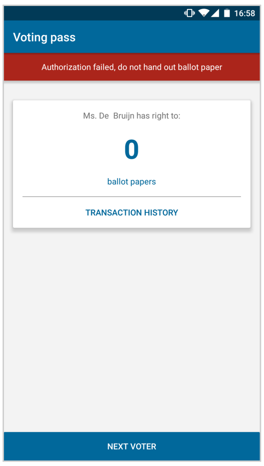
Authorization successful

Please hand Ms. De Bruijn

1

ballot paper

(h) Confirmed transaction


16:58

Voting pass

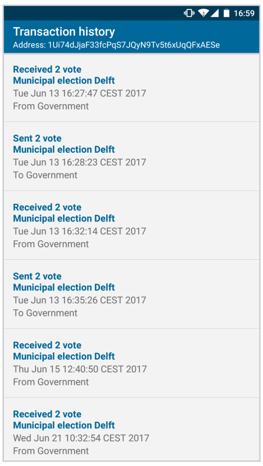
Authorization failed, do not hand out ballot paper

Ms. De Bruijn has right to:

0

ballot papers

(i) No voting passes


16:59

Transaction history
Address: 1Ui74dJjaF33fcPqS7JQyN9Tv5t6xUqQFxAESe

Received 2 vote
Municipal election Delft
Tue Jun 13 16:27:47 CEST 2017
From Government

Sent 2 vote
Municipal election Delft
Tue Jun 13 16:28:23 CEST 2017
To Government

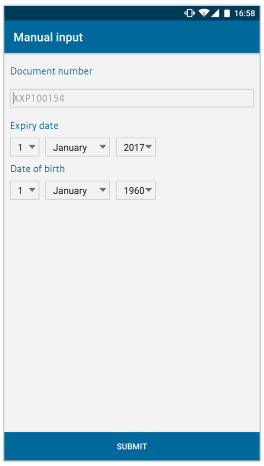
Received 2 vote
Municipal election Delft
Tue Jun 13 16:32:14 CEST 2017
From Government

Sent 2 vote
Municipal election Delft
Tue Jun 13 16:35:26 CEST 2017
To Government

Received 2 vote
Municipal election Delft
Thu Jun 15 12:40:50 CEST 2017
From Government

Received 2 vote
Municipal election Delft
Wed Jun 21 10:32:54 CEST 2017
From Government

(j) Transaction history


16:58

Manual input

Document number: XXP100154

Expiry date: 1 January 2017

Date of birth: 1 January 1960

(k) Manual input

Figure E.3: Final result

F
ePassportChain parameters

In this appendix almost all parameters of ePassportChain that can be adjusted are discussed. For each parameter, a description of what its function is, is given. Next to this, the value of each parameter of ePassportChain is given with an explanation why this value was chosen. Note that the descriptions come directly from the MultiChain parameters guide[1].
The parameters are divided into six tables, each covering a section. First, Table F.1 shows the basic chain parameters. The next table, Table F.2, shows the parameters permissions. Subsequent, Table F.3 describes the consensus parameters. Table F.4 describes the mining runtime parameters while Table F.5 describe the mining parameters. Finally, in Table F.6 the transaction parameters are given.

Parameter	**Description**	**Value**	**Explanation**
chain-protocol	Use MultiChain for a MultiChain blockchain or bitcoin for a bitcoin-style blockchain with no permissions, native assets or streams.	MultiChain	Necessary to have permissions in the blockchain
chain-description	Textual description of the blockchain for display to users.	Elections X	
root-stream-name	Name of the root stream for general data storage (leave blank for none).	root	Default
root-stream-open	Allow anyone with send permissions to write to the root stream.	true	Default
chain-is-testnet	Whether to set testnet to true in the output of various JSON-RPC API calls. This is for compatibility with Bitcoin Core and does not affect any other testnet-like behavior.	false	The blockchain should not be a test net
target-block-time	Target average number of seconds between blocks, i.e. delay for confirming transactions. If this is below 10 seconds, it is recommended to set mining-turnover low, to minimize the number of forks.	10	See subsection 4.5.2
maximum-block-size	Maximum number of bytes in each block, to prevent network flooding by a rogue miner.	1MB	Default, should be able to hold 5k transactions, see final report

Table F.1: Basis chain parameters

[1]See `http://www.multichain.com/developers/blockchain-parameters/`

Parameter	Description	Value	Explanation
anyone-can-connect	Apply no restriction to connecting to the network, i.e. nodes do not require connect permissions.	true	For transparency reasons
anyone-can-send	Apply no restriction to sending transactions, i.e. signing transaction inputs.	false	Government can freeze wallets if person in question loses his right to vote, (e.g in case of incarceration or death)
anyone-can-receive	Apply no restriction to receiving transactions, i.e. appearing in transaction outputs.	true	Ability to proxy-vote
anyone-can-receive-empty	Apply no restriction to addresses which appear in transaction outputs containing no native currency, assets or other metadata. Only relevant if anyone-can-receive=false. This allows addresses without receive permission to include a change output in non-asset transactions, e.g. to publish to streams.	false	Default
anyone-can-create	Apply no restriction to creating new streams.	false	No streams, except the root stream, are used so unnecessary
anyone-can-issue	Apply no restriction to issuing (creating) new native assets.	false	Issuing means creating new election which is only reserved for the government
anyone-can-mine	Apply no restriction to mining blocks for the chain, i.e. confirming transactions.	false	Mining reserved for government nodes to protect against monopoly attack
anyone-can-activate	Apply no restriction to changing connect, send and receive permissions of other users.	false	Only admin is allowed to do this
anyone-can-admin	Apply no restriction to changing all permissions of other users.	false	Obvious
support-miner-precheck	Support advanced miner permission checks by caching the inputs spent by an administrator when setting admin or mine permissions – see permissions management for more information.	true	Default
allow-p2sh-outputs	Allow pay to scripthash outputs, where the redeem script is only revealed when an output is spent. See permissions management for more information about permissions and P2SH addresses.	false	p2sh is not needed to create and send/verify transactions. This is thus an unnecessary risk (when turned on), which can be avoided.
allow-multisig-outputs	Allow multisignature outputs, where more than one address is explicitly listed in a transaction output, and a given number of these addresses are required to sign in order to spend that output. See permissions management for more information about permissions and multisignature outputs.	false	Isn't needed

Table F.2: Global permissions parameters

Parameter	**Description**	**Value**	**Explanation**
setup-first-blocks	Length of initial setup phase in blocks. During the setup phase, the constraints specified by the other parameters in this section are not applied.	60	This does not matter since the government will send about 13 million transactions, which will create a lot of blocks
mining-diversity	Minimum proportion of permitted miners required to participate in round-robin mining to render a valid blockchain, between 0.0 (no constraint) and 1.0 (every permitter miner must participate). Unlike mining-turnover, this is a hard rule which determines whether a blockchain is valid or not.	0.75	Default
admin-consensus-admin	Proportion of permitted administrators who must agree to modify the admin privileges for an address, between 0 (no consensus required) and 1 (every admin must agree).	1	All admins for safety
admin-consensus-activate	Proportion of permitted administrators who must agree to modify the activate privileges for an address, between 0 and 1.	1	All should agree in case an admin node gets hacked.
admin-consensus-mine	Proportion of permitted administrators who must agree to modify mining privileges for an address, between 0 and 1.	1	All admins agree for safety reasons
admin-consensus-create	Proportion of permitted administrators who must agree to modify stream creation privileges for an address, between 0 and 1.	1	There is only stream, and no more. So default one
admin-consensus-issue	Proportion of permitted administrators who must agree to modify asset issuing privileges for an address, between 0 and 1.	1	All should agree in case an admin node gets hacked.

Table F.3: Consensus parameters

Parameter	Description	Value	Explanation
lock-admin-mine-rounds	Ignore forks that reverse changes in admin or mine permissions after this many (integer) mining rounds have passed. A mining round is defined as mining-diversity multiplied by the number of permitted miners, rounded up. This prevents changes in the blockchain's governance model from being reversed and can be overridden by each node using the lockadminminerounds runtime parameter.	10	Default value
mining-requires-peers	A node will only mine if it is connected to at least one other node. This is ignored during the setup phase or if only one address has mine permissions, and can be overridden by each node using the miningrequirespeers runtime parameter.	true	An isolated node mining is useless to the network
mine-empty-rounds	If there are no new transactions, stop mining after this many rounds of empty blocks. A mining round is defined as mining-diversity multiplied by the number of permitted miners, rounded up. This reduces disk usage in blockchains with periods of low activity. If negative, continue mining indefinitely. This is ignored during the setup phase or if target-adjust-freq>0, and can be overridden by each node using the mineemptyrounds runtime parameter.	5	No mining needed when there is no election in progress or no passes being cashed,
mining-turnover	A value of 0.0 prefers pure round robin mining between an automatically-discovered subset of the permitted miners, with others stepping in only if a miner fails. In this case the number of active miners will be mining-diversity multiplied by the number of permitted miners, rounded up. A value of 1.0 prefers pure random mining between all permitted miners. Intermediate values set the balance between these two behaviors. Lower values reduce the number of forks, making the blockchain more efficient, but increase the level of mining concentration. Unlike mining-diversity, this is a recommendation rather than a consensus rule, and can be overridden by each node using the miningturnover runtime parameter.	0.1	Is low to reduce forks since the block time is set low

Table F.4: Mining runtime parameters

Parameter	Description	Value	Explanation
skip-pow-check	Skip checking whether block hashes demonstrate proof of work.	false	Obvious
pow-minimum-bits	Initial and minimum proof of work difficulty, in leading zero bits. (1 - 32)	4	Should be low so it easy to mine a block and thus conform transactions
target-adjust-freq	Interval between proof of work difficulty adjustments, in seconds, if negative - never adjusted. (-1 - 4 294 967 295)	-1	Difficulty should not increase, to ensure blocks are still mined fast enough
allow-min-difficulty-blocks	Allow lower difficulty blocks if none after 2*.	false	Difficulty is already low, so this is not necessary

Table F.5: Mining parameters

Parameter	**Description**	**Value**	**Explanation**
only-accept-std-txs	Only accept and relay transactions which qualify as 'standard'.	true	Default
max-std-tx-size	Maximum size of standard transactions, in bytes. (1024 - 100 000 000)	2 048	Transactions are currently maximal 500 bytes so 2 048 is more than enough
max-std-op-returns-count	Maximum number of OP_RETURN metadata outputs in standard transactions. (0 - 1 024)	10	Default
max-std-op-return-size	Maximum size of an OP_RETURN metadata output in a standard transaction, in bytes.	2 097 152	Default
max-std-op-drops-count	Maximum number of inline OP_DROP metadata elements in a single output in standard transactions.	5	Default
max-std-element-size	Maximum size of data elements in standard transactions, in bytes.	8 192	Default

Table F.6: Transaction parameters

G

Test plans

Tests plans are used to ensure that pieces of code which hard to test work correctly. Two tests plans have been created, one which tests if the passport connection is functional and one for ePassportChain. The passport connection test can be found in section G.1, while the test plan of ePassportChain can be found in section G.2.

G.1. Passport connection

For each release to the master branch the following steps must be taken to ensure the connection with the passport still works as intended.
Do all of the below for a Dutch passport and a Dutch ID card, both issued after March 9th 2014. The whole test suite should be done on two different devices in order to uncover device specific issues.

Successful connection

1. Start up the app
2. Wait for the app to be initialized
3. Choose an election
4. Press manual input button
5. Put in the document details of the travel document and submit
6. Start connection
7. Hold travel document to back of phone
8. Wait until the passport connection is completed
9. **The authorization screen should be showed**

G.1.1. Bad BAC key

1. Start up the app
2. Wait for the app to be initialized
3. Choose an election
4. Press manual input button
5. Put in wrong document details and submit
6. Start connection
7. Hold travel document to back of phone
8. **The passport connection should fail and display an error indicating the document details are wrong**

G.1.2. Bad NFC connection

1. Start up the app
2. Wait for the app to be initialized
3. Choose an election
4. Press manual input button
5. Put in the document details of the travel document and submit
6. Start connection
7. Hold travel document to back of phone for a very short time failing the passport connection
8. **An error indicating to retry holding the travel document to the phone should be displayed.**

G.2. ePassportChain

This test plan describes how ePassportChain must be tested after each change. To following steps should be performed:

1. Create a new ePassportChain: `./multichain-util create NAME`
2. Start the daemon: `./multichaind NAME`
3. Open an CLI which is connected to the new chain: `./multichain-cli NAME`
4. Get the address which able to create an asset with `listpermissions issue`, call the obtained address X
5. Create a new asset with: `issue X asset 1 000 1`
6. Create a new address: `getnewaddress` call this address Y
7. Give this address receive permissions: `grant X receive`
8. Send a token from X to Y: `sendasset Y asset 1`

If any of these commands fail, the changes have broke something and further inspections is necessary.

Digital Voting Pass

One step towards the digitalization of the entire voting process

Project on behalf of our client Milvum under supervision of the department of Distributed Systems at Delft University of Technology, the project will be publicly presented on July 3, 2017.

Description

The challenge of this project is to design a solution, supported by modern technology, which increases the trustworthiness of the voting process and decreases the manual labor. In order to narrow the scope, the main focus lies at the digitalization of the voting pass, the document which is needed at the polling station to prove that the citizen is eligible to vote.

In order to create such a trustworthy system, the research is focused at two parts. The first part explores the possibilities with blockchain technology. In the second part, the capabilities of the biometric chip in ePassports are investigated by performing several experiments.

After this research, the process continues with an agile approach and started the development of a digital system that could make the voting pass superfluous. This digital system consists of several interconnected parts. The polling station app and the ePassportChain implementation are the most essential parts. The app is used at the polling station to redeem the suffrage. Furthermore, the ePassportChain implementation stores the voting passes in a decentralized way.

The project is first of its kind to conceptualize the direct connection between a blockchain and a travel document. This technological advancement is valuable to the client, because they are starting a new project in collaboration with five municipalities to digitalize the entire voting process, where identity is a major subject.

Team members

Wilko Meijer
: Contributions: Connecting the travel document with the app, connection of app with blockchain, unit and integration tests and implementation of general app functionality.

Daan Middendorp
: Contributions: The initial connection with the travel document proof of concept, ePassport blockchain address generation. Alternative blockchains exploration, the election management utility and wallet implementation in app with transaction builder.

Jonathan Raes
: Contributions: Scanning of MRZ code in app, downloading blockchain to application, general app functionality, testing.

Rico Tubbing
: Contributions: Modifying MultiChain to support the brainpoolP320r1 elliptic curve, building a raw ePassportChain transaction and general app functionality.

Client, coach and contact

Salim Hadri	CEO at Milvum
Johan Pouwelse	Distributed Systems at Delft University of Technology
Team Digital Voting Pass	Contact <digitalvotingpass@gmail.com>

The final report of this project can be found at: `http://repository.tudelft.nl/`.