Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Electronic Election System for Academic Senate

Bachelor thesis

2018
Adam Štefunko

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Electronic Election System for Academic Senate

Bachelor thesis

Study programme:	Computer Science
Study field:	2508 Computer Science
Department:	Department of Computer Science
Supervisor:	RNDr. Jaroslav Janáček, PhD.

Bratislava, 2018
Adam Štefunko

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Adam Štefunko
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Electronic Election System for Academic Senate

Annotation: This thesis deals with design and implementation of an electronic election system for academic senate. It defines functional and security requirements for an election system and analyzes to what extent the designed solution conforms to the requirements.

Aim:
- define requirements for an election system
- design a solution
- analyze the level of conformance of the designed solution to the defined requirements
- implement the solution as a web application for online voting

Supervisor:	RNDr. Jaroslav Janáček, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	26.10.2017

Approved: 26.10.2017 doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Adam Štefunko
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Electronic Election System for Academic Senate
Systém pre elektronické voľby do akademického senátu

Anotácia: Táto práca sa zaoberá návrhom a implementáciou systému pre elektronické voľby
do akademického senátu. Definuje funkčné a bezpečnostné požiadavky na volebný systém a skúma, do akej miery navrhnuté riešenie tieto požiadavky spĺňa.

Cieľ:
- definovať požiadavky na volebný systém
- navrhnúť riešenie
- preskúmať, do akej miery riešenie zodpovedá definovaným požiadavkám
- implementovať riešenie v podobe webovej aplikácie na online hlasovanie

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 26.10.2017

Dátum schválenia: 26.10.2017 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgement: There are many people to which we want to express our thanks.

In the first place, we would like to thank the supervisor of our bachelor thesis, **RNDr. Jaroslav Janáček, PhD.**, for irreplaceable help and pieces of advice in various topics. These include thoughts on how to design a voting scheme or what encryption method to choose. The time we spent with him was very inspirational for us. Especially, our supervisor helped us so much when we found a crucial error in our system.

A few people quickly helped us with several implementation problems. With help of **Jakub Šimo**, we discovered many *JavaScript*'s secrets and we found out why we were not able to install *Python* packages on our *iMac*. This issue was, fortunately, solved.

Our voting system is developed for our faculty. Therefore, several faculty services needed to be used within it. **Mgr. Matej Zagiba** explained to us how *Cosign* works and gave us numerous examples.

Last but not least, we would like to thank the members of **ŠKAS** (our faculty's student senate), and especially their vice-chair **Mgr. Júlia Pukancová**, for suggesting us the idea of developing an electronic election system for our faculty.

Abstract

Electronic election represents a modern way of collaborative decision-making. It can, in many aspects, simplify and automatise traditional election processes. However, it raises new security and usability questions. These include easy-to-use interface, cost-effectiveness, reliability of the system or correct computation of the votes. In this thesis we try to deal with them and implement our own electronic election system. Inspired by existing solutions, we propose our own electronic voting scheme. Our proposal is, then, used to develop an electronic election system. This system uses an Internet browser to cast a vote and is run on a server, which also collects the votes. At the end of the election process, the votes are counted in the machine for vote counting. This system is developed on top of several cryptographic primitives, such as asymmetric encryption or secret sharing, which are also described in this thesis. Our system is, finally, analysed regarding the defined requirements.

Keywords: electronic election system, electronic voting scheme, internet application, academic senate

Abstrakt

Elektronické voľby reprezentujú moderný spôsob skupinového rozhodovania sa. V mnohom dokážu zjednodušiť a zautomatizovať tradičné voľby, avšak prinášajú so sebou nové bezpečnostné a používateľské otázky. Sú to napríklad ľahko použiteľné užívateľské rozhranie, rentabilita, spoľahlivosť systému či správne spočítanie hlasov. V tejto práci sa s nimi snažíme vyrovnať a implementovať náš vlastný elektronický volebný systém. Po inšpirovaní sa existujúcimi riešeniami navrhneme vlastnú elektronickú volebnú schéme. Z tejto schémy potom vyvinieme vlastný elektronický volebný systém. Tento systém využíva na hlasovanie internetový prehliadač a beží na serveri, ktorého úlohou je aj zbierať hlasy. Na záver volieb sú hlasy sčítané v sčítacom zariadení. Tento systém je postavený na báze niekoľkých kryptografických metód, ako je napríklad asymetrické šifrovanie alebo zdieľanie tajomstva. Tieto sú taktiež popísané v našej práci. Na záver analyzujeme náš systém na základe definovaných kritérií.

Kľúčové slová: elektronický volebný systém, elektronická volebná schéma, internetová aplikácia, akademický senát

Contents

List of Figures

Introduction

Lives of modern people are accompanied with modern digital technologies. Many people cannot even imagine their life without a computer devices. Governments and companies often promote their use in a variety of fields. Since its beginning, Internet has spread out throughout the whole society and services like Internet banking, social networks and e-mail are used on daily basis by plenty of people in this world.

There are many attempts to fully digitalise governmental administration, including the election process. However, this raises several questions of trust and many usability, security and accuracy requirements that every electronic voting system should follow have been defined.

Throughout the centuries, several traditional voting systems have been developed. These include *ostraca*—pieces of broken pottery having been used in Athens—or classic ballot papers in envelopes, as we know them today. These systems have been being improved over a long period of time and they represent a confidential way of expressing one's opinion.

Despite reliability of traditional voting mechanisms, many institutions are interested in introducing modern technologies to the field of voting. These include either special-purpose machines or personal computers. Use of these technologies, of course, makes us meet new challenges. It took the previous generations several centuries to solve many usability, security and reliability issues makers of electronic voting systems need to solve almost immediately. Despite all the considerations [24], electronic—particularly Internet—voting is on its best way to become very popular.

Using cryptographic methods, such as blind signatures, anonymous channels or homomorphic encryption [16], it is possible to find an elegant way to solve all these issues. There have been quite a few attempts to design a trustworthy election system. Some of the most successful approaches are election systems developed for Estonia [18, 26], Norway [12, 13, 14] and Switzerland [27]. They have been run as pilot projects by the countries' governments.

Out faculty has also expressed their interest in electronic voting. It is planned to be used by our students to choose their representatives it our faculty's academic senate. In this thesis, we design and implement a minimal, yet secure remote electronic election system. It can enable the students to cast a vote with use of nothing but an Internet

browser in their personal computer or mobile device. We consider all the limitations of such technology that can have impact on our solution.

Finally, we analyse our solution and we try to figure out to what extent it meets defined usability, security and accuracy requirements.

Chapter 1

Description of the Electronic Election System

In this chapter, we talk about some possible definitions of the electronic election system, we propose a definition with which we want to work and we present some requirements on such a system. We also discuss two main approaches to the system in terms of technology used and describe our choice of the system we use for our solution.

1.1 Definition of the Electronic Election System

To have the system correctly and precisely designed, it is necessary to have it clearly defined. At first, we need to say something about the electronic election or voting itself. We have borrowed descriptions from two large online encyclopaedias. We think they are amongst the first sources to which a person being interested in this particular topic is introduced. According to Wikipedia, electronic voting "refers to voting using electronic means to either aid or take care of the chores of casting and counting votes" [28]. In comparison, Encyclopaedia Britannica offers this description of electronic voting, "a form of computer-mediated voting, in which voters make their selections with the aid of a computer" [4]. It can be observed that the two descriptions are vague in terms of roles, or how the actual voting and counting is processed. They only say that computers are used to manipulate with the votes and that voters are enabled to vote using computers. For the use of our thesis we define a computer system which performs manipulation with the votes and which is responsible for the electronic election. This is the definition we propose:

Definition 1: *Electronic Election System* is a computer system which

i. authenticates the voter,

ii. enables the voter to cast a vote,

iii. securely transfers and stores the vote,

iv. counts the votes.

This definition simply enumerates all the basic roles of our electronic election system. We want the election process to be as automated as possible; therefore, the definition is directly derived from the tasks performed during regular election for our academic senate. The system authenticates a voter, so it checks whether the voter is authorised to vote; it enables every authorised voter to cast their vote; it transmits and stores every vote in a way that no vote is lost or changed; and after the voting period has ended, it counts all the votes and outputs the result.

All the authorities and voters involved in electronic elections need to follow a particular electronic voting scheme (or protocol). This scheme prescribes procedures which should be proceeded during the voting process and which should describe how the electronic election system should perform its roles. Such scheme usually consists of three stages [29]:

1. **Initialisation**, during which the elections are announced, questions are being made, and all the private and public keys are being generated.

2. **Voting**, during which voters are casting their votes: ballots are being created and then sent.

3. **Counting**, during which ballots are being opened and counted, and the final results of the elections are being published.

Several existing voting schemes are discussed in Chapter 2 and our proposed voting scheme is described in Chapter 3.

1.2 Requirements on an Electronic Election System

Every electronic election system must satisfy several requirements to make sure that all its tasks have been performed unmistakably and that the result is demonstrably correct. In this section, we present several requirements which we find essential for the purposes of the elections to the academic senate. We also compare them to the requirements presented in bibliography sources on this topic.

P. P. Bungale and S. Sridhar from Johns Hopkins University, Baltimore have named several *Requirements for an Electronic Voting System* [5]. They have divided them into two categories: "Functional Requirements" and "Security Requirements". We have chosen several of them which we find most important for our electronic election system, and we have added a few that are not included in Bungale's and Sridhar's

original paper, yet we find them very important. We have partially followed their division. However, besides adapting the terminology we have added an extra category for the sake of better distinction. Thus, we divided them in these three categories: *usability*, *security*, *accuracy*.

Usability requirements are *easy-to-use user interface*, *mobility*, *cost-effectiveness* and *confirmation*.

- **Easy-to-use user interface.** The system should have a user interface which voters can use easily with (almost) no instructions provided. On top of that, voting options should be displayed in a way that no candidate is disadvantaged.
- **Mobility.** The voters should not be restricted to a certain place where they can vote. In terms of electronic voting, the voter should not be limited to a certain type of technology used for voting. In spite of Bungale's and Sridhar's opinion, who call voting via the Internet "*infeasible* both for security issues as well as social science issues" [5], we advocate Internet voting because we think that in certain conditions, it is preferable and can meet our requirements (particularly this one).
- **Cost-effectiveness.** The technology used for electronic voting should not be expensive and hard to implement, yet it must provide adequate functionality and security, so it can be effectively used as an electronic election system.
- **Confirmation.** Each voter should have the chance to confirm that their vote corresponds to their own decision and have the chance to modify their vote before committing it. Voters also should have the chance to verify whether their vote was correctly transferred and stored.

Security requirements are *secrecy*, *anonymity*, *reliability* and *incoercibility*.

- **Secrecy.** There must be no chance to determine how a voter voted.
- **Anonymity.** No vote must be associated with a voter's identity.
- **Reliability.** The system must be robust enough, so that no votes are lost or illegally changed in any case, and it must be ensured there is no malicious code or bugs. Also, the system should be simple enough because such system offers fewer possibilities for the attackers and, thus, is less vulnerable.
- **Incoercibility.** There must be no way in which voter can prove how they have voted. This prevents from anyone else's impact on voter's choice and it also prevents from vote-selling.

Accuracy requirements are *authorisation*, *uniqueness and limitation*, *persistence* and *correct computation*.

- **Authorisation.** It must be secured that only authorised voters can cast their votes.
- **Uniqueness and limitation.** Each voter can participate in an election by no more than one vote and no more candidates than allowed can be chosen.
- **Persistence.** It must be guaranteed that votes remain intact after they have been committed and sent.
- **Correct computation.** The votes must be correctly computed according to the published rules of the election.

For comparison, Rjašková [29] has provided a set of seven requirements, which we introduce to the reader: *eligibility*, *privacy*, *individual verifiability*, *universal verifiability*, *fairness*, *robustness*, *receipt-freeness*, *incoercibility*. We strongly believe that almost each our requirement finds its counterpart in one or more Rjašková's requirements, and vice versa. For instance, counterpart to confirmation is individual verifiability and counterpart to reliability is robustness. Receipt-freeness means that there is no way to prove how a voter voted [9]. This can be substituted for incoercibility and vice versa.

1.3 Types of Electronic Election System

In this section, we present two possible types of electronic election system, which are the most common, and of which we choose one for our purposes. Then, we compare those two types regarding our requirements which we presented in Section 1.2. We also present their advantages and their drawbacks. Finally, we give reasons for our choice of used type of electronic election system.

Regarding the technology used during the voting phase (as described in 1.1), there are two main types of electronic election system: *e-voting* and *i-voting* [4].

1.3.1 E-voting

This type uses special-purpose machines designed directly for the purposes of the elections. These machines either directly record the ballots or they optically scan traditional paper ballots, which are then stored in their internal memory.

Reliability relies on testing the machines during the initialisation phase and on confidence that the same software is running during the whole election process. However, thanks to relatively smaller number of individual machines used during the election

and thanks to higher possible responsibility from the authority, these machines can be checked for malicious software and controlled without much difficulty. Also, network communication between a device of this type and any other device is generally disabled. Hence, it is possible to implement this type of electronic election system in a way that it does not break security requirement in such a manner which is fatal to the election process.

However, regarding functional requirements, we find this type of system awkward. Firstly, special-purpose hardware is built to hold the tasks during the election process and these machines are placed in special-purpose polling stations where the voting process is controlled by a commission. Voters have to come to the polling station where they can cast their ballot. Depending on the software in these machines, it is possible that the voter has to learn how to use the machine before they cast their vote.

1.3.2 I-voting

The second type, on the other hand, uses Internet to hold the communication between the authorities and the voter that enables the voter to be authorised and to cast their vote.

One of the biggest security issues with this type of the system is the relatively large amount of independent devices, owned mainly by the voters themselves, which can run a malicious software which might not, depending on the security support of the individual device, be spotted. Rubin notes how many violent acts may potential attackers perform: "view every aspect of the voting procedure, intercept any action performed by the legitimate user with the potential of modifying it without the user's knowledge, and further install any other program of the attacker's desire—even those written by the attacker—on the voter's machine" [24]. The second issue is the Internet communication itself. Thus, cryptographic protocols, some of which are to be described particularly in chapters 2 and 3 and which are largely discussed in [29], are used to prevent Internet communication from vulnerability during the election process and to assure to a great extent that security and accuracy requirements are not to be broken.

Despite all the security issues, mobility and cost-effectiveness are two important advantages of this type of electronic voting system. A voter needs nothing, but their computer or mobile device with either a special-purpose application or Internet browser. Moreover, it is relatively cheap to develop a special software that runs on a user's device. There is left to consider whether to use Internet browser or a special-purpose application. We find the first option as preferable from the point of view of mobility, whilst the second is, in our opinion, preferable when considering security. We should still bear in mind that a lot of important details lie on the way how the system is

developed, which is to be discussed more deeply in chapters 3 and 4.

1.3.3 Comparison

All these facts and expectations described above have been considered for the purposes of our choice. We state that not all requirements we present in Section 1.2 are equally important for the purposes of the system. Also, some requirements can be equally satisfied using either type. Mobility and cost-effectiveness have perhaps been the most crucial during our decision process because our system is to be made for no more than a thousand of voters involved in one election. Albeit not the most secure, our system should provide reasonable security using accessible solutions. Some other security and functional requirements, such as anonymity, reliability and persistence, are essential, but these can be adequately achieved using either type of the system and appropriate protocols.

E-voting systems are ideal for elections involving a large number of voters, e.g. parliamentary elections, which is not our case. Thus, we decided to use i-voting because it meets our functional requirements perfectly and it also provides adequate security thanks to cryptography. Possible security issues can be treated quite quickly without any difficulty also thanks to the relatively small amount of individuals involved in the elections.

Chapter 2

Summary of Existing Solutions

Several computer scientists have been working on electronic election systems until this day. We introduce and describe some techniques used to design electronic voting schemes as well as some existing solutions to the electronic elections.

2.1 Electronic Voting Scheme Techniques

Electronic voting systems use several cryptographic primitives to assure integrity, authenticity and correct transfer of votes as well as to preserve voter's anonymity. There are at least three specific techniques used to design an electronic voting scheme. These are *blind signatures*, *anonymous channels* and *homomorphic encryption* [16]. They are used besides symmetric and asymmetric cryptography, certificates, digital signatures, etc. Here, we describe the ideas on which they are based and some of their properties. We also briefly describe *mixing networks*, *onion routing* and *David Chaum's untraceable electronic cash protocol*.

2.1.1 Blind Signatures

In many voting systems, blind signatures are used to detach the vote from the voter's identity. The basic principle of such schemes is that blinded empty ballot is sent to the authority alongside voter's personal information. After verifying voter's identity and correctness of the ballot, it is blind signed by the authority and sent back to the voter. Then, voter unblinds the ballot, fills it and sends to votes collector. Simple example of such a scheme is described in [2].

As written in [16], all protocol requirements, except receipt-freeness, are accomplished in schemes using blind signatures when some other cryptographic primitives, such as encryption, are used together with blind signatures. But, these schemes have restricted usability due to the fact that in order to ensure fairness and verifiability,

voters need to be active in at least two phases. However, there are some protocols that overcome this functional drawback.

2.1.2 Verifiable Anonymous Channels

We use anonymous channels to send anonymous messages. This means that there is no way to trace the received message back to the sender. Anonymous channels can be built using *mix networks* or *onion routing*.

- **Mix Network** uses a chain of proxy servers that mix the multiple messages and send them in a random order to the next node, which might be either another mixing server or a different type of device.

- **Onion Routing** encapsulates a message in new layers of encryption. Analogous to this are layers of onion, hence the name. The message is, then, transmitted through a series of subsequent routers, each decrypting the actual top layer of encryption and uncovering the message's next destination. In every moment of the message transfer, only precedent and following location is known to each *onion router*, thus making the sender's identity untraceable [15].

It is necessary to ensure that no messages are dropped or substituted during the transfer. Therefore, *proofs of correct computation* need to be provided. Channels that are able to do so are called *verifiable anonymous channels*. Their main drawbacks are computational complexity and inefficiency [16].

2.1.3 Homomorphic Encryption

Use of homomorphic encryption is based on the desire to decrypt the sum of votes without being able to decrypt the individual ballots. It means that no voter can get anyone else's vote, while the calculation of the votes remains publicly verifiable. For that reason, voters can openly authenticate to the server [17].

However, such schemes have several serious drawbacks. Not only do they require computationally intensive zero-knowledge proofs, but they also do not allow voter to choose multiple options [16].

2.1.4 Untraceable Electronic Cash Protocol

In [7], David Chaum introduces a protocol which can be used to send electronic cash while preserving sender's anonymity and avoid double-spending at the same moment. Moreover, if a sender tries to send several copies of an electronic coin, their identity

can be revealed. The protocol also provides a method that can be used to reveal double-spender's identity. With a few modifications, this protocol is useful for electronic voting to prevent double-voting.

The Chaum's protocol can be described in the following way. It can be noticed that we present only the most important steps to the reader.

In this example, the symbol $\cdot$ denotes concatenation. Let n be an RSA modulus and let f and g be two-argument collision-free functions. Let u be sender's account number and v be a counter associated with this number and kept in the bank.

1. **Formation**

   (a) The sender chooses $k \in \mathbb{N}$ values a_i, c_i, d_i and r_i, where $1 \leq i \leq k$. All a_i, c_i, d_i and r_i are modulo n.

   (b) The sender sends to the bank k *blinded candidates* $B_i = r_i^3 f(x_i, y_i) \pmod{n}$, where $x_i = g(a_i, c_i)$ and $y_i = g(a_i \oplus (u \cdot (v + i)), d_i)$.

   (c) The bank randomly picks $k/2$ indexes from $\{i_1, ..., i_k\}$. The sender shows their a_i, c_i, d_i and r_i to the bank for each picked i. From these, the electronic coin C is formed following steps described in [7].

2. **Payment**

   (a) In order to pay, the sender sends their coin C to the shopkeeper.

   (b) The shopkeeper randomly chooses $k/2$ binary values $x_1, ..., x_{k/2}$. For each $x_i \in \{x_1, ..., x_{k/2}\}$:

   - If $x_i = 0$, the sender shows the shopkeeper a_i, c_i and y_i.
   - If $x_i = 1$, the sender shows the shopkeeper $a_i \oplus (u \cdot (v + i)), d_i$ and x_i.

   (c) The shopkeeper sends obtained values to the bank in order to verify the coin C.

When the sender tries to use C twice, it can be observed that with high probability complementary binary values will be sent to the bank for at least one x_i and the sender's identity will be revealed. This is possible because in this situation, all the values i, a_i and $a_i \oplus (u \cdot (v+i))$ are known, from which the sender's identity can be easily extracted.

Although not designed directly for the purposes of electronic election, this protocol can be used, requiring only slight modifications, such as making u a unique personal identification number given to every voter prior to the election. Such a modified scheme can be found in [22].

2.2 Existing Electronic Election Systems

Many governments are interested in replacing traditional election schemes based on paper voting with electronic election systems. At the turn of the millennia, *SERVE* was an ambitious experiment by the United States of America in the field of Internet voting. It was later cancelled, mostly due to large security weaknesses SERVE showed. Switzerland was amongst the first countries to use electronic elections in some of their cantons. Estonia was the first country to run nationwide electronic parliamentary elections. In this section, we describe pioneering election systems developed for Estonia, Norway and Switzerland.

2.2.1 Estonian Internet Voting

Estonian identification cards (ID cards) have a chip that stores asymmetric cryptography key pair, which their owners can use to digitally sign documents. This has been used to develop an internet voting protocol, which was first used during the election in 2005, while it was still possible to vote traditionally. Information about this protocol comes mainly from [26].

The protocol used in Estonian internet voting system uses method of *double envelope*. While the outer envelope is used to provide the voter's identity, the inner is used to cryptographically protect the vote. These entities play role in this protocol: *voter*, *voting commission*, *voting application*, *server*, *ballot box*, *counting device* and *certification authority*.

The protocol consists of three stages, which we now describe:

1. **Initialisation.** Election asymmetric encryption key pair is generated. The public key is given to every voter while the private key is divided using a secret-sharing scheme and each member of voting commission is given their piece. The voting application is digitally signed and provided to every voter.

2. **Voting.** Voter, who has validated their ID card, connects to the server and launches the voting application on their device. In order to establish secure connection, the voter needs to provide a PIN code associated with their ID card. The voter, then, chooses their favourite options to form the vote. When the vote is formed, it is padded using *OAEP* padding scheme with randomness r and encrypted with the election public key using *RSA*. After that, the voter digitally signs the vote using their ID card. The vote encryption forms the lower envelope and assures the vote protection, while the digital signature forms the upper envelope and identifies the voter. The *double-enveloped* vote is, finally, sent to the server.

The server verifies voter's identity and strips off the upper envelope. It sends the vote in the lower envelope to the ballot box and associates it with a random token t. The voter can check whether their vote was correctly recorded using a verification app on their smartphone. The app reveals r and t in the form of a *QR* code. The voter uses a smartphone app to scan the *QR* code and to compare the encrypted vote stored in the ballot box with a simulated vote encrypted using r and corresponding the voter's choices. Verification can be done no later than half an hour after casting the vote and up to three times per a single vote.

3. **Counting.** Valid encrypted votes stored (without the signatures) in the ballot box are burned to a DVD, on which they are transferred to the counting device. The private key is reconstructed and loaded onto the counting device. It produces the result, which is burned on another DVD and published.

Estonia's Internet Voting Committee claim that, in terms of security and reliability, this voting protocol is equal to traditional elections. However, it seems to be controversial. Tests run by a team from the University of Michigan show that the system can be successfully attacked on both the client-side and the server-side. In one type of client-side attacks we take advantage of the fact that the voter can cast a vote more than once. Malware is installed on the voter's computer that sniffs the voter's PIN code while the voter is electronically voting. The malicious program waits until the verification period expires or until the voting application is closed and the *QR* code can no more be scanned. As soon as the *ID* card has been inserted into the computer, the malware open a hidden mock version of the voting application and submits a replacement vote [26].

2.2.2 Norwegian Internet Voting Protocol

In Norway, trial of Internet elections was first organised in 2011. A protocol that would satisfy the security expectations was defined for the Norwegian elections [12] and a new cryptographic protocol was published two years later [13]. A new instantiation of the cryptosystem underlaying Internet voting in Norway was introduced and is described in [14].

In the original paper, associated with the election in 2011, a simplified voting protocol is described. We believe that this protocol provides enough information about the ideas on which the Norwegian internet voting is based. The players in the protocol are: *voter*, *voter's computer*, *ballot box*, *receipt generator* and *decryption service*.

The voting scheme is divided into three stages and the following steps are processed:

1. **Key generation.** Asymmetric encryption keys and three secret parameters for the election are generated, first for the decryption service, second for the ballot box and the third for the receipt generator. From these, associated public parameters are computed. Expected return codes for each voter are also generated and sent to the voter.

2. **Voting.** When the voter choses their options, voter's computer sends the encrypted vote to the ballot box. With the cooperation of receipt generator, receipt codes are generated and sent to the voter. Those are checked by the voter, if they correspond with the expected receipt codes.

3. **Counting.** Submitted votes from the ballot box are sent to the decryption device. There, they are decrypted and counted in order to get the final result.

According to the author of the protocol, its security properties are not perfect, yet it is reasonable to be used for an i-voting experiment. In order to assure that the vote remains confidential, the voter's computer should not be corrupted. If the vote is not correctly submitted, although not corrupted before being submitted, the voter can see this through receipt codes and can complain. The author also concludes that "any corrupt infrastructure player may prevent the election from completing." [12].

After the elections in 2011, the protocol used for it was gradually renewed and improved. One of the last introduced improvements on the underlying cryptosystem was that new techniques were used to prove the knowledge of the decryption of the encrypted text. These new techniques have the same effect as former; however, due to their use, the underlying cryptosystem has a better security proof [14].

2.2.3 Swiss Online Voting Protocol

There are multiple public voting events in Switzerland every year. Not only can the citizens choose their representatives in Federal Assembly, but also referenda are organised federally or regionally.

Cantons of Geneva, Zürich and Neuchâtel were the first to introduce electronic voting. In 2013, The Federal Council of Switzerland published a framework which provides security, verifiability and functionality requirements in order to allow all the voters to vote electronically. In a publication by Scytl [27], a protocol ensuring cast-as-intended verification is provided. This protocol has been created on the basis of Norwegian internet voting protocol (described in this thesis in 2.2.2). It is basically the improvement of that protocol "by not needing to rely on the strong assumption that two independent server-side entities do not collude to preserve voter privacy" [27]. The building blocks of the protocol are *ElGamal assymetric encryption scheme*,

bit-length prime numbers representing voting options, *pseudo-random function family*, *signature scheme* made up of probabilistic algorithms and a *verifiable mix network*.

These are the participants in the Swiss voting scheme:

- **Election Authorities**, responsible for the whole elections;
- **Voters**, who express their opinion by casting a vote;
- **Registration Authorities**, who provide voters with all the information needed;
- **Voting Server**, which receives, proceeds and stores the votes;
- **Voting Device**, responsible for enabling the voter to select their options, forming the vote and correctly send it to the voting server;
- **Code Generator**, which generates return codes from the votes;
- **Auditors**, in charge of verifying the correctness and integrity of election process.

The Swiss voting scheme consists of following processes divided into four phases:

1. **Configuration.** A set of voters' identities ID is defined and published. The protocol uses several key pairs, of which public keys are published. Those keys are *election public key*, *global code generation public key*, *signing public key*. Voting options are also published in this phase. The global code generation private key is given to the code generator and the registration authorities, and the signing private key is given only to the registration authorities.

2. **Voter registration.** In order to register to the election, voter provides their identity $id \in ID$ to a registration authority. Then, voter is given their public and private keys, a set of return codes associated with particular choices and confirmation and finalisation codes. Also, voter's public key is published together with reference values associated with return codes and a validity proof for finalisation code.

3. **Voting.** When the voter is successfully authenticated, voter's id and public key are stored in their voting device. The voter votes by choosing from the voting options and entering their private key. From these, the vote is created and sent together with voter's id to the voting server.

   When the server successfully receives the vote and the voter's identity, return codes are generated and shown to the voter, who, then, is asked to confirm the vote. The voter does so by providing their confirmation code to the voting device. Subsequently, the confirmation message is sent to the server. When

the vote is confirmed and finalisation code is sent back to the device and the voter checks whether it matches the finalisation code they have obtained during the registration.

4. **Counting.** The counting algorithm takes the votes stored on the voting server, the election private key and validity proofs for finalisation codes and produces the final result, which is, then, verified.

The described protocol requires the voter to type several private values to the voting device. This means to copy by hand 52 or 410 characters when *Base32* is used [27]. The voter cannot perform such a task. Therefore, the protocol also describes a usability layer, whose role is to reduce the length of values that the voter needs to directly provide for the voting system. Its description can be read in the original paper a we do not include it in this thesis.

Chapter 3

Our Solution

The main purpose of this chapter is to present our own solution to the electronic election system. We describe players in the final voting scheme we implement, the voting scheme itself and the format in which the vote is stored and read. Then, we bring some description of existing security schemes and services we use in our solution. We also discuss some other proposals of the voting scheme, which we modified or rejected.

3.1 Our Voting Scheme

Our voting scheme is similar to the scheme that has been developed for elections in Estonia. This scheme is described in Chapter 2. Both schemes are based on the principle that voter's identity cannot be known to any player besides that particular voter when their vote can be decrypted or read. It means that computer that is responsible for decrypting the votes cannot obtain any information about any voter who casted their vote.

We bear in mind that we require the voting process to run in an Internet browser and that this technology has several limitations. The reader can read more about limitations of the technology used in Chapter 4.

3.2 Players in Our Voting Scheme

We shortly describe each player in our voting scheme and their purpose.

- **Voter** V authenticates to server for vote collection S using *authentication authority* T and casts their vote using *voting application* A. They also can cast their vote in paper form in a special-purpose room.
- **Election commission** B are in charge of key generation during the first phase. They hold secret key SK generated for *machine for vote counting* M and share

corresponding public key PK. They also securely transfer encrypted data from *server for vote collection* S to M. Data can be, then, decrypted using SK, provided to M by B. In the end, they publish the result of the elections. Last but not least, they are responsible for paper votes casted in a special-purpose room designed for the elections.

- **Database** D stores information about all the candidates and identities of all authorised voters. This information is then used by A and S to display list of possible candidates and authorise *voter* V. This database is physically stored on server S and is accessible only to S.

- **Voting application** A provides user interface for V in which they can cast their vote. It also validates the vote, converts it to a proper format and encrypts it using PK. Then it sends the vote together with the voter's identity.

- **Server for vote collection** S receives encrypted vote and voter's identity and checks whether this identity corresponds to any of those stored in *database* D. If the encrypted vote comes from an authorised voter, then it stores the encrypted vote and the voter's identity in its internal database of votes. This server also provides storage for voting application A and database D.

- **Machine for vote counting** M receives all the votes (without identity of any voter) stored in S's internal database and secret key SK. It, then, decrypts all the received votes using SK, validates them and computes the result of the elections, which is finally published. We point out that this machine is not connected to any network prior or during the elections. This machine is restricted to be in off-line mode until the results of the elections are published.

- **Authentication authority** T is responsible for authentication of V and provides S with V's identity, which must *1-to-1* correspond to a value in D. For purposes of our voting scheme, we use *Cosign*, which is a service commonly used for authentication of students studying at our faculty. From this point, we mean *Cosign* any time we talk about T. Similarly, we use term *UK login* of voter V instead of *V's identity*. More information about *Cosign* is provided in Section 3.5 and Chapter 4.

3.3 Phases of Our Voting Scheme

Our voting scheme is designed to have three subsequent phases: *initialisation phase*, *voting phase* and *counting phase*. The basic purpose of these stages is discussed in chap-

ter 1. In this section, we provide a detailed description of what is done during each of these phases.

3.3.1 Initialisation

1. Database D with a table of candidates is created and stored on S. For each candidate, the table contains a unique ID and name of the candidate.

2. A table of authorised voters is created in D. For each voter, this table contains their *UK login*.

3. A special *UK login* is created for the election commission B. This login is used when a voter casts their paper vote.

4. Public key PK and secret key SK are generated. They are to be used to encrypt the vote by the voting application A and decrypt it by the machine for vote counting M, respectively.

5. Using *Shamir's Secret-Sharing Scheme*, the key SK is divided into n parts, where n is the number of members of B. The key SK can be reconstructed from any combination of k parts, where k is the smallest number of members of B that have to sign the *Protocol of elections*. The reader can read more about the scheme in Section 3.5.

6. Public key PK is shared with all the voters by inserting it into A as a constant value for the whole elections.

3.3.2 Voting

For each voter V:

1. Voter V opens a specific Internet location in their browser and logs in using *Cosign* in order to authenticate to S.

2. If V is successfully authenticated, voting application is launched in their Internet browser. This application contains a form in which there are three options for each candidate.

3. When V submits the form, it is then validated regarding the rules of the elections. When the form is valid, vote in the format described in Section 3.4 is created and this vote is encrypted using *OpenPGP* standard and PK. Finally, the vote and voter's *UK login* are sent to S via an *HTTPS* connection.

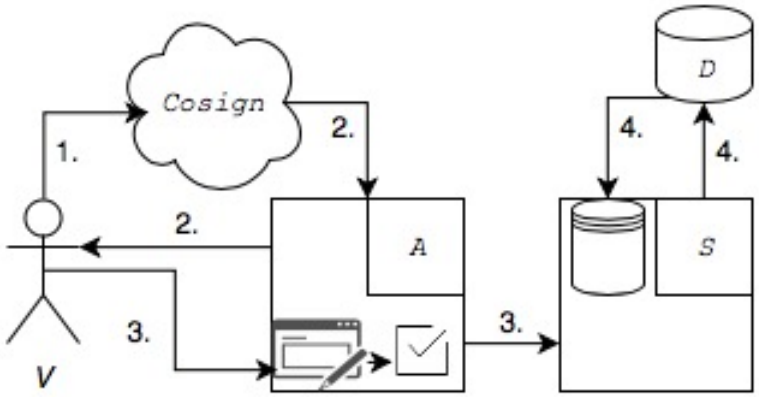

Figure 3.1: Voting phase of our scheme

4. When S receives the encrypted vote, it compares the sender's *UK login* with logins in the table of authorised voters stored in D. If it matches exactly one record, then it is compared whether there is a record with the same login. If there already is such a record and the vote in this record is different form the special `0` character vote in S's internal database of votes, this vote is rewritten with the new one. If there is not such a record, the login and the encrypted vote are stored in the database.

5. If V decides to use paper vote and goes to the special-purpose voting room, the election commission B authenticate to S using their special *UK login*. When they give V a ballot paper, they send a special `0` character vote together with V's *UK login*, which rewrites V's electronic vote if they have sent one and prevents V from casting an electronic vote.

3.3.3 Counting

1. Server S is disconnected from network and off-line mode is enabled.

2. Using a secured USB device, the election commission B transfers encrypted votes from S's internal database of votes to machine for vote counting M. It is crucial that *UK logins* stored in this database be excluded from the transfer.

3. With aid of at least k members of election commission B, secret key SK is reconstructed. Then, it is inserted into M and used to decrypt votes.

4. Decrypted votes are, then, validated by M. All votes that are in the right format and follow the rules of the elections are counted and the final result is published.

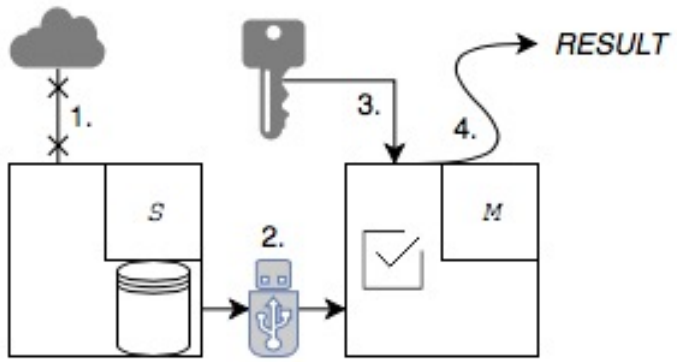

Figure 3.2: Counting phase of our scheme

3.4 Format of the Vote

Definition of format to which each vote is formatted prior to being sent to the server for vote collection S is also part of our voting scheme. We want the vote to be stored in one compact file and the choices to be clearly separated. It is known how many candidates are in the elections. Let M be the number of candidates. Hence, we propose that the vote is of the following format:

```
<candidate_number_1#value_1>...<candidate_number_M#value_M>
```

Here, `M` is the representation of number of candidates M.

This format represents a string of ordered pairs (c_i, v_i), where $c_i \in \{1, ..., M\}$ is `candidate_number_i` and $v_i \in O$ is `value_i`, $i \in 1, ..., M$. Value v_i is numerical representation of one of the voting options and O is the set of all possible numerical representations. For example, 1 represents YES, 0 represents NO and $O = \{0, 1\}$.

Let $(c_1, ..., c_M)$ be an ordered set of all candidate numbers. Then, $(c_1, ..., c_M)$ is a permutation of $\{1, ..., M\}$.

It is obvious that the candidate number 1, for instance, is given YES if and only if the vote contains exactly one ordered pair of value $(1, 1)$.

3.5 Security Tools Used in Our Voting Scheme

In our voting scheme, we use these cryptographic protocols and schemes as well as security services to securely transfer and store the data.

3.5.1 OpenPGP

Developed to provide a secure way to communicate electronically and store data, *OpenPGP* (Open Pretty Good Privacy) uses asymmetric, symmetric encryption and

hash algorithms. Not only can it be used by traditional email clients, but also by message transfer services that use cryptographic solutions to prevent any person from undesired intervention. One obtains or generates *OpenPGP* private and public keys when they want to encrypt data using this standard. In order to do this, one sets a passphrase to be used to decrypt the encrypted data with a particular *OpenPGP* private key.

Callas et al. provide a specification of *OpenPGP* in [6]. More can be also read in [10].

3.5.2 Shamir's Secret-Sharing Scheme

In [25], Adi Shamir proposes a solution to the problem of dividing data into n pieces in such a way that these data can be reconstructed using k of these n pieces, but attempt to reconstruct the data with fewer than k pieces results in no information about it. Such a scheme is called *(n, k)-threshold scheme*. The scheme he proposed is based on polynomial interpolation.

Let $p(x)$ be a polynomial of degree d. It is obvious that $d + 1$ distinct points $(x_0, p(x_0)), ..., (x_d, p(x_d))$ are sufficient to define $p(x)$ and that $p(x)$ cannot be defined with fewer than $d + 1$ distinct points.

In Shamir's scheme, assume that data D is a number. Then, to divide D into n pieces in such a way that it can be reconstructed by using at least k pieces, we follow these steps:

1. We pick a random $k-1$ degree polynomial $p(x) = a_0 + a_1x + ... + a_{k-1}x^{k_1}$, where $a_0 = D$.

2. We, subsequently, evaluate $D_1 = p(1), ..., D_n = p(n)$. Then, each of n distinct pieces of secret consists of an ordered pair $P_i = (i, D_i)$, where $i \in 1...n$.

3. In order to reconstruct data, at least k pieces of secret $P_{i_1}, ..., P_{i_k}$ are put together. Polynomial interpolation is then used to find $p(x)$, which is the Lagrange polynomial for the set of used pieces of secret. This makes recovering data D possible because $p(0) = a_0$ and $p(x)$ has been constructed in such a way that $a_0 = D$.

The scheme as described above consists only of the basic idea on which secret sharing is based. In reality, we use modular arithmetic instead of its real counterpart. Let $p > \max(D, n)$ be a prime number. Then, the set of integer coefficients $a_0, ..., a_{k-1}$ is picked randomly from a uniform distribution over integers in $[0, p)$. Values $D_1, ..., D_n$ are also computed modulo p. The set of coefficients forms a field $\mathbb{Z}_p$, which makes polynomial interpolation and data retrieval possible [25].

This scheme has some useful properties. It can be observed that the individual pieces are at most of the same size as the original data. With k fixed, some pieces can be added or deleted without affecting the other, or the pieces can be changed without changing the original data. It is also possible to reflect hierarchy in a particular group, such as election commission, in a way of distributing the pieces. For example, chairperson of the commission is given three pieces, vice-chairperson two and the other members one.

3.5.3 Cosign

Comenius University in Bratislava uses *Cosign*, a secure single sign-on web authentication system originally developed at the University of Michigan [20]. It is based on web cookies.

Authentication data are sent to the central server, which uses an authentication protocol to verify these data. The server, then, sets the user a web cookie, controlled by *Apache* filters during every user's attempt to access addresses secured by the system. During the whole process, *TLS* is used for a secured communication.

In Chapter 4 is described how authentication via *Cosign* has been integrated into our system.

3.6 Other Proposed Solutions

In the course of designing the final voting scheme for our election system, we proposed some other solutions. Those were either rejected, or the final scheme was based on them and their modifications. There are two elementary design classes being in consideration during the whole process. Those are based on question whether server gives client any additional information used to authorise the sender of the vote. Regarding this, we consider two meaningful classes, which we call *authorisation by a token* and *authorisation by user information*. In all of the schemes we have looked at, at least three players represented by computers are needed: a *client* C, an *authentication server* A and a *counting machine* B.

3.6.1 Authorisation by a Token

In this design, authorisation in provided by a *token*. This token assures B that the vote comes from an authorised voter without associating it with a particular voter. In order to preserve anonymity, token must be generated or handled with in a way that it cannot be associated with any particular voter. In Chapter 2, we describe some existing solutions using this design.

Here, we briefly describe voting phase of a scheme that uses server A's signature to prove that the vote comes from an authorised voter.

Let V be voter and $n \in \mathbb{N}$, $n \geq 2$, be constant defined by the scheme. Then, for each voter V:

1. Voter V logs in and voting application is opened on C.
2. Client C sends n encrypted votes and V's identity to A, where 1 of those is the real vote and the other votes are fake.
3. If V is an authorised voter, A signs all n votes and sends them back to C.
4. Client C sends the signed real vote to B.
5. Server B validates the vote and if it recognises the signature with which the vote is signed, then it stores the vote.

The main issue with this scheme is that it may enable the voter to double-vote when their voting application is corrupted. In Chapter 2 we introduce the reader to David Chaum's scheme, which was designed to be used with electronic cash to prevent double-spending. When using a modification of this scheme to prevent double-voting, we face a problem. In order to achieve this, additional data about the voter have to be generated and stored on client's side. Of course, these data cannot be stored on voter's personal computer or smartphone, since our scheme is supposed to be mobile and these data have to be accessed from any device of those kinds. This means that a cloud storage in which these data can be stored has to be provided. This storage has to be secured in a way that only voter can freely access the data. It raises new security tasks, which we think the reader can imagine. We do not say that it is impossible to implement such a system, but we think that it is not necessary to implement such a complex system for purposes of our type of elections.

We have also come up with a totally different scheme using mix networks, referred to in Chapter 2, to mix the tokens in order to assure anonymity.

3.6.2 Authorisation by User Information

Using voting schemes in this class, C assures that the vote comes from an authorised voter by providing a publicly inaccessible piece of information unique for that voter that can be matched with a record in A's memory. In order to retain anonymity, the vote must be in an illegible form during the whole time it can be associated with this piece of information.

Here, we provide a simple scheme that has been modified and partly used in the final voting scheme.

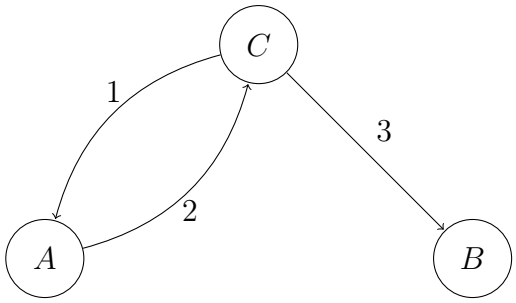

Figure 3.3: Authorisation by a token

Let V be voter. Then, for each voter V:

1. Voter V logs in and the voting application is opened on C.
2. Client C sends encrypted vote and V's identity to A.
3. If V is authorised to vote, then A sends the encrypted vote without V's identity to B.
4. Server B stores the vote.

This scheme has several security issues. Let corrupted A send B encrypted votes and voters' identities. These votes can be, then, decrypted by B, which also knows who casted which vote. Thus, it can be revealed how voters voted.

Due to this fact, we propose in our final solution that the server A stores the votes in an encrypted form and that the device B stays in off-line mode during the whole election. Moreover, the server A is turned off before the votes are provided to B. This disables all the possible communication between A and B. We only need to assure that nothing but individual encrypted votes are transferred from A to B by the election commission.

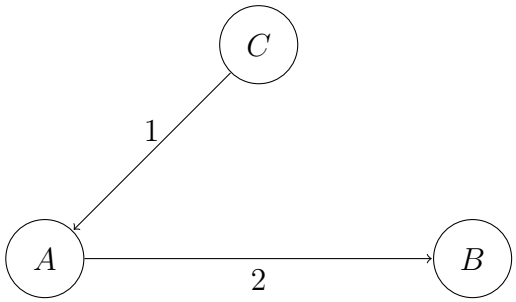

Figure 3.4: Authorisation by user information

Chapter 4

Implementation of Our Solution

Our task for this thesis is to develop a functioning electronic election system that can be used by our faculty's academic senate. In this chapter, we describe some important implementation details and we provide examples of important parts of our code.

4.1 Overview

For the desired system, we implement the electronic voting scheme described in Chapter 3. This scheme consists of several entities—players, which together make our election system and some of which are represented by computers. These players have to be implemented independently, each running its own program that provides all its functionality and communication with other entities. Analysis of the final solution can be found in Chapter 5.

We have chosen *Python* [11] and *JavaScript* [19] as our primary programming languages. For databases, we use *SQL* [8], a standard language in database programming. Our used encoding is *UTF-8*.

We name this system **SEVAS** (**S**ystem of **E**lectronic **V**oting for **A**cademic **S**enate). The source code for this system is attached to this thesis and consists of several Python and JavaScript files that provide the voting functionality and the administration of votes, and of some additional configuration and database files. More information about the source files can be found in the included `README.md` file. The code for this system is publicly available and can be found on GitHub. The reader can access it here: `github.com/adamcho14/SEVAS.git`.

In this election system, we have three possible answers for each candidate: *positive*, *negative* and *neutral*. However, if one wishes, the code for this system can be easily changed to use a different number of possible answers.

4.1.1 Technical Requirements

In order to run the system correctly, there have to be some technical requirements accomplished. On the client-side, the voter needs to have one of these supported Internet browsers: *Mozilla Firefox*, *Google Chrome* or *Safari*.

The system is developed to run on *Linux* distributions and the system can be built from existing *Python* source files. In order to provide the authentication, the *Linux* voting server needs to have *Apache 2* server installed with a module that provides *Cosign* functionality. More about *Cosign* can be read in Chapter 3.

The voting is provided by a series of *CGI* scripts. Common Gateway Interface (CGI) is a protocol used to run external programs—*CGI* scripts—under a server. It forwards the request sent by the client to the external program, which is executed, and the output of the program in the form of a response is sent via the server to the client [23]. These scripts are very often used to generate dynamic web content.

In order to manage the keys and certificates, *GnuPG* has to be installed on the machine for vote counting. It is a full-featured and free implementation of *OpenPGP* standard [21]. We access *GnuPG* functionality via the `gnupg` module for *Python*.

4.1.2 External Libraries and Packages Used

In almost every file, we include packages `cgi` and `cgitb` to provide the *CGI* functionality. We also use several external libraries or packages to implement services that we do not program directly. They are used in different parts of our election system:

- **Front end JavaScript vote forming and vote encryption.** JavaScript cryptography library `openpgpjs`.
- **Back end Python internet voting and vote collection.** We import Python packages `os` and `sqlite3`.
- **Machine for vote counting code.** In this code, packages `json`, `sqlite3` and `gnupg`.
- **Code for key and vote administration.** Here, we import packages `sqlite`, `gnupg` and `secretsharing`.

4.2 Database

We use an open-source database library *SQLite* [1], which is an embedded database with no separate server process. We strongly believe that this database library is sufficient for our system because it is used as a server-side data storage. The client does

not communicate directly with the database, but they either send data to the server, which are stored in the database by the server, or all Select statements are first posted by the server and their results are sent to the client. We, too, do not expect very high traffic because our system is created for elections with no more than about a few thousand voters (our faculty has about a thousand students).

We have also chosen *SQLite* mostly because we find it sufficient for the purpose of this particular implementation, that is to show how our electronic election system works. If the user of this system wishes, *MySQL*, *PostgreSQL* or another *SQL* server can be chosen for the practical use instead of *SQLite* library. It is obvious that it requires some minor changes in implementation, which can be easily executed.

Database structure of our voting system consists of two separate databases stored in the memory of the server for vote collection: *database of persons* and *database of votes*. There are discusses in the following subsections.

4.2.1 Database of Persons

The purpose of this database is to store information about both the candidates and the voters. They are used to display the list of candidates and to provide the list of authorised voters. Thus, this database has two tables, whose structure is described below. Not only do we provide the reader with the information about the columns in these particular tables, but we also show them how we create the tables. In this description, we use standard *SQL* syntax to present the way in which the tables can be created:

```
CREATE TABLE Candidates (
        CandID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
        LastName varchar(255) NOT NULL,
        FirstName varchar(255) NOT NULL);

CREATE TABLE Voters (
        UKLogin varchar(255) NOT NULL PRIMARY KEY);
```

Here can be seen that the table of candidates consists of all the information that is displayed to the voter. We suggest that the `CandID`s be in ascending order starting from number 1. The table of voters contains only the voter's *UK login*. The other information are not important for the server and can be fetched from the the data provided by *Cosign*.

The reader can obtain some more information about the usage of these databases in the next sections.

4.2.2 Database of Votes

This database is the internal database of the server for vote collection. Its purpose is to store encrypted votes paired with the voters' *UK logins*. The database is made up of a table that can be created with the following *SQL* statement:

```
CREATE TABLE Votes (
        UKLogin varchar(255) NOT NULL PRIMARY KEY,
        Vote varchar(255) NOT NULL);
```

4.3 Voting

This application provides the voter with interface that allows them to authenticate and cast their vote. This application runs on the server for vote collection as a series *CGI* scripts. The application works as following:

1. When the voter types the *WWW* address of the Internet elections, the main page with the login button is displayed.

2. When the voter clicks on the login button, they are redirected to the University's central login webpage, where they can authenticate.

3. When the voter has been successfully authenticated, a form with the voting options, the maximum number of chosen options and a button with the label *Create a vote* is displayed.

4. When the voter's desired options are chosen, the voter clicks on the button. If they have chosen more than the maximum number of options, an alert is displayed. Otherwise, the vote is created, encrypted and displayed on the page together with a button whose label says *Send the vote*.

5. When this button has been clicked on, the voter is informed on the page whether or not their vote has been successfully processed. At this moment, they have successfully casted a vote and they can log out, or, for some of the reasons described in Chapter 3, they were not allowed to cast a vote.

In order to provide all the functionality, the files `openpgp.min.js` and `form_processing.js` have to be included in the source code for the voting application. The former provides necessary *OpenPGP* functionality and the latter includes *JavaScript* code for client-side vote validation, formation and encryption.

4.3.1 The Form

The main part of the voting application is the *HTML* form. It is sent using the *POST* method. It contains a list of candidates from the database of persons, described in Subsection 4.2.1. There are three radio buttons for every candidate, each representing one of the possible answers: *positive*, *negative*, or *neutral*. To each of these answers a particular integer value, obligatory for the whole election, has been assigned. In our implementation, positive answer is represented by `1`, negative by `2` and neutral by `3`. This form also includes field *vote*. This is to be filled with the encrypted vote. More about vote formation can be read in Subsection 4.3.2.

Some parts of our code use *JavaScript* and the voter's *UK login* is displayed in the *HTML* source code. This means that the voter can change the code of the page and send corrupted data, such as a vote that is not in the right format. Therefore, votes are validated during the counting period.

4.3.2 Vote Formation and Encryption

We use a *JavaScript* program to form the final vote from the voter's answers. The format of the vote is described in Section 3.4. Here, we describe what is done with the data from the form in order to become the encrypted vote.

It is checked whether the number of positive answers in the form does not exceed their maximum number defined by the election commission. All the candidate *ID*s and the values corresponding the checked answers are put together to form the vote. The vote is, then, encrypted using *OpenPGP* encryption provided by the `openpgpjs` library. For the encryption, the election public key is used. More about managing the election keys and certificates is written in Section 4.7.

4.4 Vote Collection

Using the *POST* method, the final vote is sent together with the encrypted *UK login* to the server for vote collection. There, a *CGI* script that verifies the voter and inserts the vote to the database of votes, described in Subsection 4.2.2, is executed.

The code of this past consists mostly of Select, Insert and Update *SQL* statements. The script decrypts the *UK login* and checks whether the particular voter has their record in the database of persons. If this is true, then it checks whether the voter has their vote recorded in the database of votes and whether this vote has the special `0` value. If the value is `0`, the vote is not recorded. If there already has a vote been recorded with the voter's *UK login* in the database, the value is updated. Otherwise, a new record with the voter's *UK login* and the received vote is created. Finally,

```
function createVote(radios) {
    var vote = "";
    for (i = 0; i < radios.length; i++) {
        if (radios[i].checked) {
            vote += "<";
            vote += radios[i].name.toString();
            vote += "#";
            vote += radios[i].value.toString();
            vote += ">";
        }
    }
    return vote;
    }
```

Figure 4.1: Code for the vote formatting function

it shows the voter a message saying whether the vote was successfully recorded or the voter is not allowed to vote.

```
import sqlite3

connection = sqlite3.connect("votes.sqlite")
cursor = connection.cursor()
cursor.execute("INSERT␣INTO␣votes␣VALUES(?,␣?)", (UKlogin, vote,))
connection.commit()
connection.close()
```

Figure 4.2: Example of a code using the *SQLite* library

4.5 Vote Transfer

This task is performed by the server for vote collection at the beginning of the counting period.

The vote transfer program selects encrypted votes from the database of votes to produce a list of all votes different from the special `0` vote. Let us call this list V. The list V is, then, saved in *JSON* encoded file F. JavaScript Object Notation (JSON) is a data interchange format derived from the JavaScript object literals defined in the *ECMAScript Programming Language Standard*. It is text-based and language-

independent. It can represent *null*, *numbers*, *boolean values*, *strings*, *arrays* and *objects*. Specification of *JSON* is contained in [3].

The file F is saved to a USB flash drive by the voting commission and transferred to the machine for vote counting.

4.6 Vote Counting

The *JSON* file F containing the votes is moved from the USB device to a particular directory. Also, the election private key is reconstructed and saved in a particular file. The machine for vote counting must, also, be provided with the list of candidates. A special *Python* script is written to create the list from the database of persons. These files are read by the counting program running on the machine for vote counting. The file undergoes several procedures in order to have the votes extracted, decrypted and counted:

1. The original list of votes V is reconstructed from F. Each vote $v \in V$ is a string representing the vote in the desired format of the vote.

2. Each vote $v \in V$ is decrypted and parsed. The result of parsing is a list of characters C which form the string v.

3. Each vote, now represented by C, is checked whether it is of the proper format and answers for each candidate, represented by the integer values, are extracted.

4. All valid votes with extracted values are counted and the final result is published by the machine.

Vote decryption, validation, extraction and counting are more deeply described in the following subsections.

4.6.1 Decryption

Vote decryption is fully provided by `gnupg` module mentioned in Subsection 4.1.1. It loads the election private key and returns the decrypted vote as a `string`.

4.6.2 Validation

The purpose of the validation is to check whether the vote is of the proper format. More formally:
Let the vote $v = v_1v_2....v_n$, where $n \in \mathbb{N}$ and $v_1, ..., v_n \in \{0, ..., 9, <, >, \#\}$. It can be noticed that subsequences that contain `0` immediately after `<` or `#`, for example `<1#09>`, are permitted. Such case can be represented in a way that the initial `0` be omitted

from the final representation of the value during the extraction process. According to this, 09 would become 9.

We call the vote v *valid* if it is accepted by deterministic finite automaton $A = \{Q, \Sigma, \delta, q_0, F\}$, where $Q = \{q_0, ..., q_4\}$, $\Sigma = \{0, ..., 9, <, >, \#\}$ and $F = \{q_0\}$. We do not describe the δ function, since it can be easily reconstructed from the diagram in Figure 4.6.2. There, the word *digit* represents any symbol of digit 0,...,9.

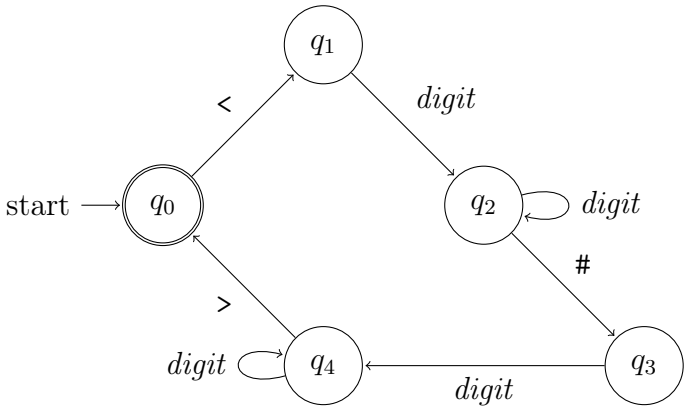

Figure 4.3: Deterministic finite automaton accepting valid votes

The reason why we describe A is that the code for vote validation is an actual simulation of this automaton. Therefore, we find A a good abstraction of the vote validation process as it is implemented in our voting system.

4.6.3 Extraction

One of the most important task of this part of our system is to extract all the ordered pairs *(canditate_ number, value)* from all valid votes. During this process, it is also double-checked whether the individual votes do not contain more positive answers than allowed.

At the beginning, a counter q for the *positive* answers is initialised and set to zero. Dictionary D is initialised. In D, candidate numbers c would form the key for which a value v associated with the answer is stored. Then, list C, representing the vote in the valid format, is looped through. Let us show to the reader how a single dictionary element $D[c]$ with value v is formed.

It can be said that the algorithm works in two distinct states. The first state is when a < symbol has been read, but the next # symbol has not been reached, yet. In this state, a candidate number is read digit by digit. Let k be the already read part of the candidate number c and the first n digits thereof have already been read. We assume that there is at least one digit before the next # symbol. We move on to the next digit d. Because the numbers are represented in the standard decimal

position notation, k is changed in the following manner: $k := 10k + d$. This process is finished when we reach a # symbol. Finally, we do $c := k$.

In the second state we have read a # symbol, but the next > symbol has not been read, yet. In this state we form the value v associated with the candidate number similarly to what was done in the first stage. When we reach a > symbol, we assign v its value and we do $D[c] := v$. If v equals the value that represents the *positive* answer, we increment q by 1.

When the whole vote was processed, we check whether q has not exceeded the maximum number of positive answers defined by the election commission. If so, we return zero, otherwise we return D.

During the extraction process, we deal with invalid votes that contain at least one of these:

- two keys k_i and k_j such that $k_i = k_j$, where $i \neq j$,
- invalid candidate number,
- invalid voting option.

Such votes are displayed to the voting commission and are not counted.

4.6.4 Counting

This is the final part of our election system. The votes with extracted values are counted and the final result is displayed in this part.

Let us recapitulate that we have 3 types of possible answers: *positive*, *negative* and *neutral*. At the beginning, $3n \in \mathbb{N}$ variables are initialised and set to zero, where n is the number of candidates. Let i be a particular candidate number and j be a value representing a particular type of answer. Then, $c[i][j]$ represents the count of answers of type j given to candidate number i. For every candidate i, it is looped through every vote v. For every value j, if j is associated with key i in vote v, then $c[i][j]$ is incremented by 1.

Finally, it is displayed for every candidate how many positive, negative and neutral answers they were given.

4.7 Administration

Some administration programs are also included in our election system. They are used by the election commission for *key generation and secret sharing of the election private key* and to *administer the paper voting process*. Here, we provide a complete list of them and shortly describe the function of each one:

```
num = 0  # checks the number of yeses
values = {}
j = 1  # the first number in the string
while j < len(l):
    key = 0
    val = 0
    while l[j] != '#':
        key = 10*key + int(l[j])
        j += 1
    j += 1  # the first number after "#"
    while l[j] != '>':
        val = 10*val + int(l[j])
        j += 1
    if val == 1:
        num += 1
    values[key] = val
    j += 2  # the first number after "<"

if num > CAND_NUM:
    return 0

return values
```

Figure 4.4: Code for extraction of the values

- **Paper voting.** This is used by the election commission to record that a voter has voted traditionally. The election commission uses their internal *UK login* and password to log in, but they do not authenticate with *Cosign*. Instead, they use a special login page. Then, a form displays and they type in the *UK login* of the voter that is voting traditionally and confirm it. In this implementation the commission's internal *UK login* and password are not encrypted, but are simply stored as variables in the *CGI* script used. However, it is more secure when a hash is used to store the data, but we did not implement it. But, it can be done in the future.

- **Key management.** Here, the electronic election public and private keys can be generated and the private key can be shared among members of the election commission.

  In order to generate the keys, the election commission may use our key generating

program. It also uses `gnupg`. The program generates a public key, which is saved to a file and a private key, which is, automatically, shared among the members of the election commission using the secret-sharing program, we also implement.

It splits the given key into several parts using `secretsharing` library for *Python*. The shares are, then, saved to specific files and given to the members of the election commission. Our key is longer than the limit of the library we use. Therefore, the key is shared line by line. For instance, in file `share_i_j.txt`, `i` is the *ID* of the share and `j` represents the j-th line of the key. Files with the same *ID* are stored in the same directory `shares_i`. Then, a similar program reconstructs the key from shares given by some of the members. The members copy the files containing their parts of secret to the directory in which is the program saved. The members of the commission type in their *ID*s and the program recovers the key.

In this implementation, however, the private key is saved to a file, which can be loaded to the key-sharing program. Its due to fact, that the vote generation program cannot find the secret-sharing program in the present file structure.

Chapter 5

Analysis of Our Solution

In Chapter 1, we established several requirements concerning *usability*, *security* and *accuracy* of an electronic election system. In this chapter we briefly and informally analyse how our solution presented in Chapters 3 and 4 meets these requirements.

5.1 Usability

Here, we discuss the usability of our electronic voting system concerning these points:

- **Easy-to-use user interface.** Our user interface is provided by an online application. Users can easily log in using the usual method used by our faculty. The actual voting is done by clicking on a few radio buttons and clicking on two submit buttons. The voter is given the information on the number of candidates they can vote in favour of. On the other hand, we could say that giving answer for every candidate might be time consuming. But, the *neutral* option is set as the default option. It means that the voting time can be reduced just to giving *yes* answers to the desired candidates while the others are left with the default answer.

- **Mobility.** The voter is not restricted to a particular type of device, nor have they to stick on an individual device. What is more, no voting application is needed to be installed. The voters can access voting everywhere. The only thing they need is a device with a supported Internet browser and an Internet connection.

- **Cost-effectiveness.** The system requires two to three computer devices on the server side, which can be costly depending on the resources of the faculty. Yet, all the software used for the system is open-source, so that it does not require any additional financial resources.

- **Confirmation.** When the vote is formed, the voter can still change their mind and create another vote before the former would be sent. Moreover, using the elec-

tronic voting, the voter can cast a vote as many times as they want to and only the most current one remains recorded. Yet, confirmation is not fully implemented in our solution. For example, the Estonian voting protocol, described in Chapter 2, implements an application that the voter can use to check their vote. It might be possible to implement such an application for our electronic election system, too. However, this is not included in our implementation partly due to preserving *incoercibility*.

5.2 Security

In this section, we discuss how secure our solution is. We analyse these requirements:

- **Secrecy and anonymity.** When the vote is formed and sent to the server for vote collection, it is in an encrypted state and the server does not possess the election private key, which must be used to decrypt the votes. Instead, this private key is shared among members of the voting commission, using a secret-sharing scheme. When the votes are transferred to the machine for vote counting, they are no more associated with voters' identities. Therefore, they can be decrypted and counted there. Keeping secrecy and anonymity relies mostly on the election commission. They should not provide the server for vote collection with the election private key. They must not provide the machine for vote counting with voters' identities associated with the votes, either. The most vulnerable part of our code is retrieving the encrypted votes from the database of votes and saving them to the USB flash drive. At this very moment, a bug or undesired intervention can cause lost of secrecy and anonymity.

- **Reliability.** Our voting scheme is very minimal and mostly uses known techniques. Votes are stored in a database and associated with an individual identity for most of the time. No anonymous channels, described in Chapter 2, are used. Therefore, chances that the votes are dropped or substituted while they are transferred or stored are low. Moreover, a suspicious voter can cast their vote again if they think there is chance that is has not been recorded correctly. Reliability also rests on the election commission during administration of paper voting and during the maintenance of server-side devices used. On the other hand, voters are responsible for they devices and they can contain malware, which can affect the election. Our implementation relies on at least two cryptographic libraries. It means that there can be some bugs in them, as well. If we find that out, we are ready to update our implementation according to the findings.

- **Incoercibility.** The voter can change their vote in any moment. That means

that, from the client side, there is no such a way in which the voter can prove how they voted most recently. On the other hand, the encrypted votes are stored together with *UK login*s of the voters' and the voters have access to the encrypted vote stored in their Internet browser during the voting session. If an attacker succeeds in accessing the database of votes, they can compare the encrypted voter's vote in the database with the encrypted vote from the voter's Internet browser.

5.3 Accuracy

It is important to make sure that only valid votes are recorded, all the votes are accurately counted and the right result is displayed. Now, we discuss how our solution meets our accuracy requirements:

- **Authorisation.** In our system, voters are authorised twice. First, they need to log in using *Cosing*, an authentication system used by our University, before the voting application is even displayed to them. When they send their vote together with their identity provided by *Cosign*, the server for vote collection checks whether they are allowed to vote. If not, the sent vote is, simply, not recorded.

- **Uniqueness and limitation.** Every voter participating in the electronic election is allowed to cast as many votes as they want to. However, only the most recent vote remains recorded. As soon as they participate in the paper election, their electronic votes are erased and they are not able to participate in the election anymore. The voting application checks whether the voter does not vote for more than allowed number of candidates in a single vote. Due to the fact that the code of the voting application can be changed in the Internet browser and an invalid vote can be sent, this is double-checked during the counting phase of our election system.

- **Persistence.** Sent votes are stored in the database and handled with according to the voting scheme. They are transferred only once—from the server for vote collection to the machine for vote counting. However, the votes can be irreversibly changed by the election commission if they accidentally make a record about a voter that has not been involved in paper voting, yet.

- **Correct computation.** The machine for vote counting is responsible for the correct computation. Therefore, this depends on the correctness of the particular implementation of the machine for vote counting. We have not discovered any bugs regarding vote computation in our implementation.

Conclusion

Estonia Norway and Switzerland are piloting countries in developing and organising electronic elections. Their election systems represent reasonable solutions and use several cryptographic primitives to provide enough security and reliability. Representatives of our faculty also expressed interest in electronic elections as a tool of democracy for our students and this thesis deals with this desire.

Firstly, we defined the electronic election system and its three basic phases, and we presented some usability, security and accuracy requirements. This was important because we needed to describe the topic with which we were about to deal as clearly as possible. Keeping in mind these requirements, we also discussed two main types of electronic election system in terms of their functionality.

We also gave examples of existing systems and described technologies they used in order to retain all the aspects of a secure and reliable electronic election system. We also discussed some problems that those systems had.

This led us to our own proposal of the electronic election system. We listed players in this system, including voters or vote counting machine. These players played several roles in three phases of our proposed voting scheme. This scheme uses several technologies, such as *secret-sharing scheme*, *Cosign* authentication system and *OpenPGP* encryption standard. To provide the reader with some information on how our system had been designed, we described several proposed schemes and we explained to the reader why we either rejected them or built on top of them.

As part of our this thesis, we also implemented the proposed system as an Internet application. This included implementation of the voting application, which can run in the Internet browser, server for vote collection, machine for counting the votes and some supplementary administration programs.

Finally, we analysed our system regarding the usability, security and accuracy requirements we defined.

We must admit that we met challenges of various kinds during the implementation of our system. In the first place, we needed to learn how to work with various cryptographic libraries, particularly `gnupg` for *Python* and `openpgpjs` for *JavaScript*. We were not a hundred percent successful in using these libraries, but we think that we managed to create a reasonable implementation. We also learnt a lot about the work

with *CGI* scripts, and *OpenPGP* encryption and decryption. Although we possessed a theoretical basis in those topics, we needed to learn from scratch how to practically use them in order to create a working system. Last but not least, we learnt new information about the Internet infrastructure of our faculty and how to use *Cosign*, a secure single sign-on used by our university.

We hope that this thesis represents only the beginning of this electronic election system. There are several other goals that we want to be accomplished. First, the system needs to be deployed and tested in the environment of our faculty in order to enable and test the *Cosign* functionality. This includes configuration of the server and the computers on which the system is to run. Then, the system can be improved and extended. A smartphone application can be one of such extensions. We think there is a lot to do in the system itself, as well. For example, customisable answers is something we have not implemented, yet. Also, the system is still not fully automated. The election public key, for instance, has to be copied to the *JavaScript* file and formatted by a member of the commission. And last but not least, our system must be used during a real election in order to find out all its advantages and drawbacks and to figure out to what extent we accomplished all our desires.

Bibliography

[1] SQLite [online]. Retrieved May 12, 2018, from `http://www.sqlite.org`.

[2] Standford University Applied Gryptography Group. Electronic Voting [online]. Retrieved April 21, 2018, from `https://crypto.stanford.edu/pbc/notes/crypto/voting.html`.

[3] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format [online]. 2017. Retrieved May 11, 2018, from `http://buildbot.tools.ietf.org/html/rfc3875`.

[4] Encyclopaedia Britannica. Electronic voting [online]. Retrieved January 24, 2018, from `https://www.britannica.com/topic/electronic-voting`.

[5] Prashanth P. Bungale and Swaroop Sridhar. Requirements for an Electronic Voting System. Department of Computer Science. Johns Hopkins University, Baltimore. Available at `http://www.cs.jhu.edu/~rubin/courses/sp03/group-reports/group4/group4_requirements.pdf`.

[6] Jon Callas, Lutz Donnerhacke, Hal Finney, David Shaw, and Rodney Thayer. OpenPGP message format. Technical report, 2007.

[7] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography*, pages 319–327. Springer, 1988.

[8] Refsnes Data. SQL Introduction [online]. Retrieved May 9, 2018, from `https://www.w3schools.com/sql/sql_intro.asp`.

[9] Stéphanie Delaune, Steve Kremer, and Mark D Ryan. Receipt-freeness: Formal definition and fault attacks. In *Proceedings of the Workshop Frontiers in Electronic Elections (FEE 2005), Milan, Italy*, 2005.

[10] Michael Elkins, David Del Torto, Raph Levien, and Thomas Roessler. MIME security with OpenPGP. Technical report, 2001.

[11] Python Software Foundation. Welcome to Python [online]. Retrieved May 9, 2018, from `https://www.python.org`.

[12] Kristian Gjøsteen. Analysis of an internet voting protocol. *IACR Cryptology ePrint Archive*, 2010:380, 2010.

[13] Kristian Gjøsteen. The Norwegian Internet Voting Protocol. *E-Voting and Identity*, pages 1–18, 2012.

[14] Kristian Gjøsteen and Anders Smedstuen Lund. The Norwegian Internet Voting Protocol: A new Instantiation. *IACR Cryptology ePrint Archive*, 2015:503, 2015.

[15] David Goldschlag, Michael Reed, and Paul Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM*, 42(2):39–41, 1999.

[16] Rolf Haenni, Eric Dubuis, and Ulrich Ultes-Nitsche. Research on e-voting technologies. *Bern University of Applied Sciences, Tech. Rep*, 5, 2008.

[17] Yehuda Lindell Jonathan Katz. *Introduction to modern cryptography*. Boca Raton, Fla.: Chapman & Hall/CRC Press, 2008.

[18] Epp Maaten. Towards remote e-voting: Estonian case. *Electronic Voting in Europe-Technology, Law, Politics and Society*, 47:83–100, 2004.

[19] Mozilla. JavaScript [online]. Retrieved May 9, 2018, from `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.

[20] University of Michigan. cosign: web single sign-on [online]. Retrieved April 16, 2018, from `http://weblogin.org`.

[21] The GnuPG Project. GnuPG. Retrieved May 14, 2018, from `https://www.gnupg.org`.

[22] Michael J Radwin and Phil Klein. An untraceable, universally verifiable voting scheme. In *Seminar in Cryptology*, pages 829–834, 1995.

[23] David Robinson. The common gateway interface (CGI) version 1.1 [online], 2004. Retrieved May 10, 2018, from `http://buildbot.tools.ietf.org/html/rfc3875`.

[24] Aviel D. Rubin. Security Considerations for Remote Electronic Voting. *Communications of the ACM*, 45(12):39 – 44, December 2002.

[25] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[26] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J Alex 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. ACM, 2014.

[27] Scytl Secure Electronic Voting. Swiss Online Voting Protocol.

[28] The Free Encyclopedia Wikipedia. Electronic voting [online]. Retrieved January 24, 2018, from `https://en.wikipedia.org/wiki/Electronic_voting`.

[29] Zuzana Rjašková. Electronic Voting Schemes. Master's thesis, Faculty of Mathematics, Physics and Informatics. Comenius University, Bratislava, April 2002. Available at `https://people.ksp.sk/~zuzka/elevote.pdf`.

Appendix A - Source Code

Source code of our implementation of the election system can be found on the CD attached to this thesis. It is also available on the Internet and can be accessed on `github.com/adamcho14/SEVAS.git`

Necessary information about the code is provided in the `README.md` and the `CONTENTS.md` files.