E-Voting on Fenix

Fernando Marques
fernando.m.marques@tecnico.ulisboa.pt

Instituto Superior Técnico
(Advisor: Professor Ana Matos
Advisor: Professor Jan Cederquist)

Abstract. An electronic voting system is a system which is used to cast and count votes in an electronic format. Some of the main advantages of e-voting are the improved convenience of the voting act, as well as the minimization of the cost and time it takes to run an election. These systems have become more secure and reliable by allowing voters to verify that their votes have been counted and some even allow to verify that all steps of an election occurred correctly while reducing the trust assumptions on which the security properties of the system rest. Nowadays voting in Instituto Superior Técnico still occurs in paper format. This work aims to propose and implement a secure and verifiable e-voting system that can be used with the authentication mechanism used in Fenix for conducting elections in Instituto Superior Técnico, although it will have portability in mind in order for adoption by other organizations with similar requirements. This report reviews some solutions that already exist for secure and verifiable e-voting and analyses their applicability to the problem at hand. It also proposes an adaptation of the system in order to work in parallel with paper voting and delineates its implementation.

Table of Contents

1 Introduction

Electronic voting, also known as e-voting is the act of voting using electronic means. E-voting is used in order to minimize the cost of the election, this being monetary cost due to the lower number of staff necessary during the election and the time that is used in the management. E-voting can also be used to provide people an alternative way to vote when they cannot be at the voting place due to illness, disability or other factors.

Some electronic voting systems still work similarly to classic voting in which the voter needs to go to a specific predetermined location to cast their vote. These systems are considered **In person e-voting** and they normally imply the use of hardware present at the scene (for example scanners which are used to scan the vote) and/or the use of special ballots [1–3].

In order to avoid the trouble of the voter having to go to the election location and as such facilitating their participation in the election **remote e-voting** systems started to appear. In these systems as long as a person has internet access they can access a portal and vote for a specific election. There are even cases of **In person e-voting** systems which were added modules in order to adapt to remote usage [2].

Nowadays e-voting is used for various types of elections from country-wide elections [4], municipal elections [5, 6] and even university staff elections [7] but in Instituto Superior Técnico,as in many universities, voting is still restricted to the classical paper form in which a person goes to a poll both, receives the paper ballot, votes and then the ballots are counted manually. In the case of the student representatives elections, the voting is done electronically but there are no security assurances in the current system.

This report proposes a solution to improve how elections are done at Instituto Superior Técnico. For this we present an implementation of a secure and verifiable remote e-voting platform, similar to that used in Helios-C [8] and Belenios [9], which can be accessed using the organizations Intranet platform (Fenix, in case of IST) credentials. An automatic voting verification, will also be added to the system, that will check if a vote was counted or not, similar to the one used in sElect [10], with the intention of increasing the number of votes verified. Lastly the system will be used in parallel to the classical paper ballot voting scheme and as such needs to be adapted to work correctly. Full documentation for the solution will also be released with it in order to maximize its receptiveness among the organization's voting community.

1.1 Goals

This work addresses the problem of developing and implementing a secure and verifiable e-voting platform to work with the Fenix authentication system. The project will produce a specification of the system produced, an easy to use e-voting system which the voter might access using Fenix and vote in a secure a verifiable way on a particular election, and an evaluation done on the system implemented. The goal of the work is to design a platform that allows every user of Fenix to vote in a particular election and that should follow the following requirements:

1.1.1 Security Requirements

1. The voter will have to authenticate himself using Fenix.
2. The votes must remain anonymous in the system, meaning that the system must not be able to associate a vote with the person who voted.
3. The system must provide integrity of the ballots which are saved before the tallying.
4. The system should ensure a high availability and reliability.
5. No one should be able to determine how and who an individual voted.
6. System operations should be logged (except for the votes) in order to facilitate verifiability.
7. The system should be end-to-end verifiable.

8. Election keys shall be distributed in order to avoid a single all powerful entity.
9. Only eligible voters should be able to vote.
10. The system shall count all the votes and do so correctly.
11. No voter should be able to vote more than once.
12. All steps of the election will be available to audit. Some will be available to audit only by the voters while others will be available to audit by everyone.

1.1.2 Functionality Requirements

1. Voting may be configured in order for it to be only available during certain time intervals.
2. The user interface should be easy to use and seamlessly merge with the one used by Fenix.
3. Step by step instructions should be given during the voting process in order to facilitate voting by less experienced voters.
4. Every voter shall be given the choice between paper voting and electronic voting.
5. The voter shall be able to cast a blank vote.

1.1.3 Architectural requirements

1. The user must be able to vote in a computer with an internet connection, preferably a trusted computer.
2. No other devices should be necessary for the voter to vote.
3. The system should be portable in order to simplify integration and adaptation to other organizations.
4. The voting system should be run on a limited amount of servers, with the maximum number being two.
5. All the design, implementation and testing shall be well documented and available to the public.

1.2 Overview of the report

The report is then structured as follows: Section 2 were we describe some related works that are already published. This section is divided in five main subsections: one describing e-voting in general, one for integrity, one for confidentiality and one where we describe in a bit more detail the main systems to be used in our solution. Finally the last subsection will have a brief analysis of all the systems viewed and some advantages and disadvantages of each one. Section 3 gives an overview of the solution with the architecture and how the protocol will work being the evaluation details presented in Section 4. Finally we present the schedule of the work in Section 5 and finish the report with a conclusion present at Section 6.

2 Related Work

In this section we start by giving the idea of how an e-voting system is organized and the main properties these systems should have. We present the related work according to which goal is prioritized, giving closer attention to the security requirements mentioned in subsection 1.1. We then describe in more detail the systems that offer security guarantees that most closely meet the security requirements of this work. Finally we give an overview of all the systems discussed, in which the main advantages and disadvantages of each system is pointed.

2.1 Preliminary concepts

As seen in the beginning of the report, e-voting has gain a lot of popularity over the years. Most of the current e-voting systems use the following common definitions which are important to understand how these systems work:

- **Voter** is the entity which casts the vote. This entity can the either honest or dishonest. Honest voters are authorized to cast the vote and do so according to the rules of the election. On the other hand dishonest voters might not be authorized to cast or may be breaking the rules of the election.
- **Candidate** corresponds to the entity to which the voters can vote.
- **Adversary** is an entity whose main objective is to disrupt the normal occurrences of the election. This entity may try to access votes, coerce voters or even cast unauthorized votes.
- **Election authority** is the entity who manages the election.
- **Ballot** corresponds to the sealed vote. Like mentioned previously these can be paper or can be a digital entity.
- **Bulletin Board** is a component where information is published. This information can range from a list of authorized voters to the election results and in some cases can even have the votes that were cast.

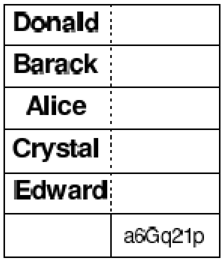
Since e-voting deals with sensitive information regarding elections, it needs some security requirements being the most important integrity and confidentiality. Integrity guarantees that the vote of a honest voter is counted and that an attempt to corrupt this must be detected. Confidentiality assures voters privacy and as such mitigates vote selling and vote coercion (although it is impossible to eliminate by 100% vote coercion).

These requirements are usually difficult to implement because the stronger one of these requirements is, the weaker the other one is going to be. For example integrity can be obtained by showing who voted for who, but this destroys privacy. On the other hand if the ballots are 100% secret then confidentiality is assured but integrity might fail [11].

2.2 Prioritizing Integrity

One of the most important properties of integrity in electronic voting is verifiability. This property allows for the verification of the system's correct behaviour and as such gains the trust of the voters. In e-voting the verifiability may be achieved by having the voter check the bulletin board to see if his vote was counted or by allowing anyone to verify if the votes of the bulletin board are counted correctly. These two types of verifiability are called **individual verifiability** and **universal verifiability** accordingly. There is also the possibility to allow the verification of every stage of the voting process in order to check if all the stages are functioning correctly. This is called **end-to-end verifiability** and can be either individual, universal or a mixture of the two where some stages are only verifiable by the election authority and the other stages are verifiable by everyone [3]. One main advantage of end-to-end verifiability is that it becomes easier to diagnose which component caused the malfunction to the system.

Prêt À Voter [3] implements verifiability by the use of special ballots. These ballots are divided in half where the left half has the candidates list in random order (different for each ballot) and the right half is the place where the vote is cast with an "X". This right side also has encrypted information called "onion" which enables the system to reconstruct the order of the candidate list (figure 1). This information is encrypted in a way that no single party can decrypt this information. The voter then casts the vote and keeps the "onion" as a receipt. When the results are published, the voter can then check if the vote and onion appear on the bulletin board correctly. If not the receipt can be used to challenge the election. The voter can also choose to audit the ballots before casting the vote. This is done by removing the left side of the ballot and asking the system to give the candidate list based on the "onion". The system also publishes enough information so that anyone can verify every stage of the election.



Donald	
Barack	
Alice	
Crystal	
Edward	
	a6Gq21p

Fig. 1. Prêt à Voter Ballot from "The Prêt à Voter Verifiable Election System" [3]

Some systems implement code voting [12] which corresponds to the use of special codes for the voter to verify that their code was valid. These codes can be associated to candidates or can be confirmation codes that are previous distributed on code sheets.

Scantegrity II [1] makes use of ballots with codes which can only be seen using a special ink from a decoder pen. This system is to be used in a polling both and as such makes assumptions that the ballots are not accessible from the outside. Each ballot is composed of a main body with a list of candidates, which have hidden confirmation codes next to each other, a machine readable serial number and two chits that can be detached from the vote. The left chit has a hidden serial number for the ballot and a space where the voter can write the confirmation code while the right chit only has a hidden serial number. Both of these serial numbers are different from each other. Each confirmation code is cryptographically linked to the candidate for each ballot and this is secure with a key that is shared among trustees. As such it is necessary for a number of trustees to collude in order to decrypt the link. When the voter goes to vote he uses the decoder pen and marks the oval space next to the candidate of choice. Doing so turns the oval space dark revealing the confirmation code for a few minutes until it also turns dark. The voter can spoil the ballot by giving it to a poll worker who without seeing the rest of the ballot removes the right chit and destroys the rest of the ballot. The right chit is then used to count the total of ballots that should be equal to those that were spoiled, audited and tallied. If the voter wishes to cast the ballot, then he gives it to a worker who scans the main body and reveals the serial of the chits. The voter can then leave with the chit. The voter can view on a website if the confirmation code that appears linked to the serial number of his ballot is the same as the one recorded. For auditing a ballot one of the chits is given at random to a worker and another of the chits is kept with the voter. The ballot is then fully marked and cast allowing for the voter to check in the website that all the confirmation codes there are correct. Finally anyone can verify the data and tally with the use of software acquired from a trusted source or written by the voter (information for writing the software is provided). In case of those organizing the election they need to appoint an independent auditor to verify the system.

Remotegrity [2] is an extension to the Scantegrity II system in order to use it remotely. This uses ballots similar to the Scantegrity II ballots but with the codes visible. The ballot is sent by mail with another card that contains one authentication serial, four hidden authentication codes, one acknowledgement code and one hidden lock code (codes are hidden behind a scratch-off surface). The voter inputs the ballot and authorization serials and the system checks if these are not duplicates. The voter then inputs the vote code from the Scantegrity card and one of the four hidden authorization codes. The system checks the code and if it is correct and was not previously used then it sends back the acknowledgement code. The voter checks if the acknowledgement code is correct and casts the vote using the lock code. Since the codes are hidden and are randomly assigned there is a low probability of guessing the correct chosen code. In case the client is corrupt and tried to change the vote code then it would have to guess the code from where the system would not reply the acknowledgement code if it were incorrect. As such the client would only have one try to guess the vote code before the voter could verify that the client was corrupt. In this case the voter would not input the lock code. In case of the system being corrupt then the system would have to use an unused authentication code. As such that can be verified by the voter if he presents the card with the codes still sealed.

VeryVote [13] is a code voting system which uses an adapted version of MarkPledge cryptographic receipts techniques [14] to achieve end-to-end verifiability. This technique has the objective of proving that a particular encryption is the correct encryption of the vote for that candidate. It uses an encoding that is a sequence of numbers where the number 1 corresponds to the selected candidate and 0 for the rest of the candidates. Then for each bit it is calculated a sequence of Ballot Mark Pairs (BMP) which are two El Gamal encryptions [15] of a 1 or a 0. If the bit being encrypted is a 1 then both encryptions of the BMP must be of the same number else both encryptions of the BMP need to be of a different number. To verify this encryption the verifier starts by committing a random challenge. The prover then sends the bit encoding and a bit string (ChoosenString) where each bit of the string corresponds to the bit encrypted in both elements of the BMP. The verifier sends to the prover the challenge which must be equal to the one it committed previously. For each bit in the challenge the prover reveals the randomness used to encrypt each BMP on the same index where bit 0 corresponds to the left BMP and bit 1 corresponds to the right BMP. This allows to decrypt the bits of the BMP and arrange them to a string which can be compared by the verifier to the ChoosenString given in the beginning by the prover. As such it proves that the encryption is well done.

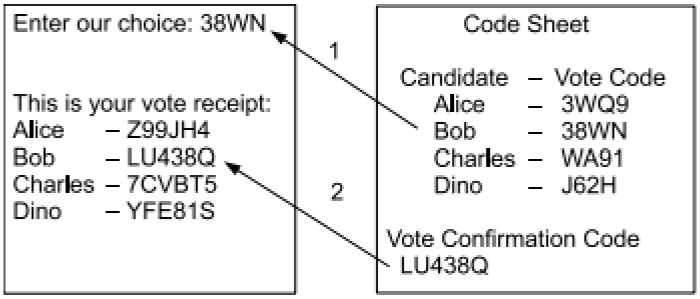


Fig. 2. In 1 the voter chooses a candidate and inserts his Vote Code. In step two, the voter can verify that the receipt he got has the confirmation code that is present in the code sheet associated with his candidate of choice. [13]

The candidate choice is done using the codes which are present in a code sheet which is sent by a third party, for example mail. This code sheet has a list of candidates, a code for each candidate and a confirmation code. The voter will then choose a candidate and insert the code that is linked to the candidate and will also commit a hash. The Voting Machine will then encrypt the vote and display a ChoosenString. The voter inserts the challenge and the voting machine computes a proof and outputs a receipt. Finally the voter only has to verify that the confirmation code appears next to the candidate he chose in the receipt (figure 2). As for other cryptographic protocols used, the election key are shared between a number of trustees using a threshold encryption scheme with a variation of the El Gamal encryption described by Cramer et al. [16]. For minimization it uses a verifiable re-encryption mix networks like the one described by Neff [14]. One of the main disadvantages in this protocol is that since it is the Election Server that makes the code sheets and since it knows who has cast the vote then it can make associate a code sheet to a voter on the election day.

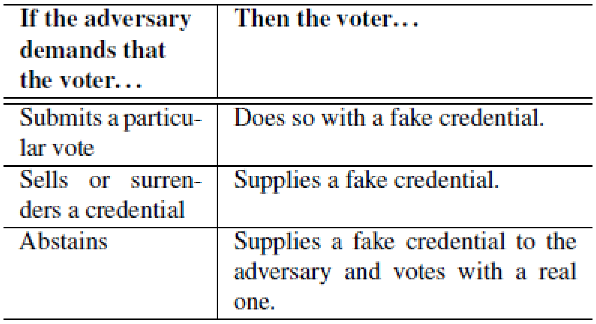
2.3 Prioritizing Confidentiality

Like in normal elections, e-voting requires confidentiality. In other words, the voter needs to have privacy so that no third party can know who they voted for. As such to promote this anonymity most of the systems implement mix networks [17]. The main objective of mix networks is to hide the link between the source and destination of the communication. As such it becomes harder to check which voter a certain ballot corresponds to. This protocol transmits a message through a number of mix servers which encrypt it, append the address of the current server and then forward it to the next mix server (this one normally being random). As such one only knows where the message previous was sent from, but never knows the origin of the message unless all servers are compromised.
Another important aspect of confidentiality is to mitigate vote selling and vote coercion although these two actions are hard to accomplish especially in remote e-voting systems.

Civitas [11] is coercion resistance e-voting system. In this system there are five main agents:

- the voters;
- the supervisor that administers the election;
- the registrar that authorizes voters;
- the registrar tellers that generate the credentials used by the voters;
- the tabulation tellers that tally the votes.

Each of these agents use a log system with a publicly available write only storage. Civitas resists coercion with the use of credentials. These credentials are divided in public credentials which the registrar tellers publish on the bulletin board and the private credentials which are given to the voter by the tellers when he authenticates himself. Each teller only has a share of the private credentials being necessary for all the tellers to work together in order to give the full private credential. When voting the voter has to input the private credential. In case of coercion, the voter may choose to input a fake credential. These fake credentials can be generated by the voters using a faking algorithm that can be run locally.



If the adversary demands that the voter...	**Then the voter...**
Submits a particular vote	Does so with a fake credential.
Sells or surrenders a credential	Supplies a fake credential.
Abstains	Supplies a fake credential to the adversary and votes with a real one.

Fig. 3. Usage of fake credentials from “Civitas: Toward a Secure Voting System” [11]

The vote is then submitted with the private credential and a proof that the vote is well formed to some or all ballot boxes. The system has the option of allowing revoting. If this option is enabled then the new vote must have a proof of the credential and the choice of the previous vote that is to be replaced. In case the option is disabled then the duplicates votes are discarded. Civitas also allows for multiple ballot formats, although write-in votes are not coercion resistant. The vote then passes through a tabulation phase where:

- the proofs are verified;
- duplicates are eliminated;
- the votes are anonymized by the use of a mix network;
- the unauthorized votes are eliminated (votes with invalid credentials);
- the tally is publicly computed.

All tabulation tellers posts proofs of the computations made allowing this protocol to be verifiable. Each tabulation teller also check the proofs as tabulation occurs and stops if a proof does not pass.

Another system which implementation whose main focus is to maintain the users privacy is **EVIV** [18]. EVIV is an end-to-end verifiable internet voting protocol. The main feature of this system is that it allows the voter to vote from any computer while at the same time guaranteeing the voters privacy. This is achieved by the use of an external device (smartcard, an USB device or even a secure element in a smartphone) which is called the voters security token (VST) which contains an unique cryptographic key pair and that is used to encrypt the voters ballot. For verifiability it is used the combination between code voting and the cryptographic voter’s verifiable vote encryption technique which is used in MarkPledge 3 [19].

The protocol is divided in four phases(figure 4):

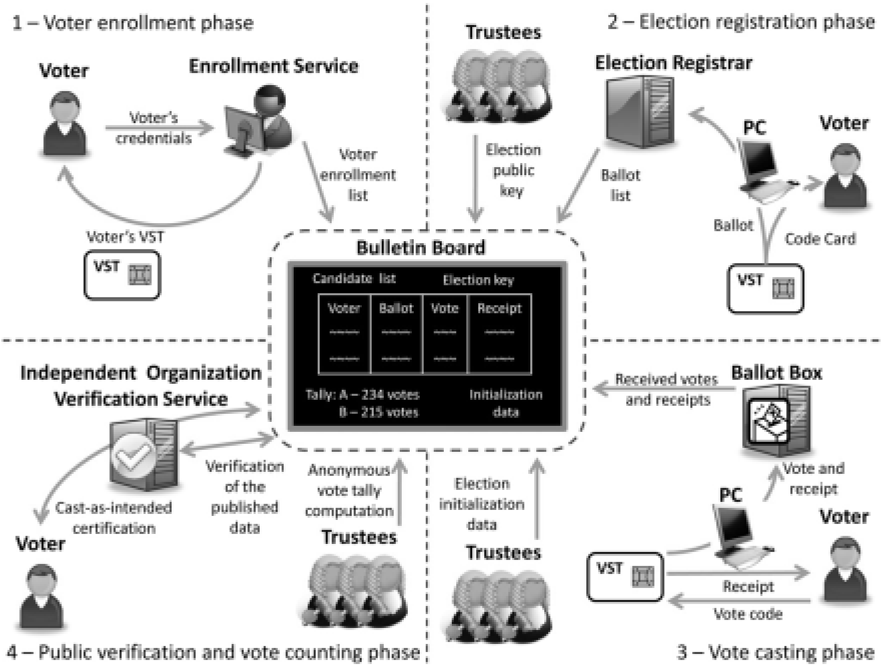


Fig. 4. Overview of the EVIV voting system [18]

- The voter enrolment phase;
- The registration phase;
- The vote casting phase;
- The public verification and tally phase.

The protocol starts with the voter enrolment phase, which is an offline enrolment of the voter. During this phase the voter identifies himself to the Electoral Commission and receives the VST with his private key and a signature by the Electoral Commission. At the end of this phase all the certificates from the voters are published on the bulletin board. Sometimes before the starting the election begins the registration phase. During this time the election parameters are set and the voter registers a ballot. It starts by the Electoral Commission publishing the candidate list and the election parameters. An El Gamal shared threshold key is also computed and distributed by the trustees being the data from this computation verified by the Electoral Commission. The voter can also decide to register for the election during this phase. For this the voter must connect the VST to a computer who will then connect to the Election Registrar using a Secure Socket Layer/Transport Layer Security (SSL/TLS) [20, 21] connection. The VST receives the candidate list and the election public key. The VST then creates the ballot which is comprised of candidate encryptions in random order and vote Validity proofs which are created using the MarkPledge vote encryption function. It then signs the ballot and send it to the Election Registrar which verifies the ballot and if it is successful then it publishes it on the Bulletin Board. The VST then creates a code card which contains random codes (one for each candidate), and a confirmation code which can be printed or noted down. Finally all ballots published are verified by the Electoral Commission and a signature over the list of ballots is created. The vote casting can be performed on any computer without the risk of the computer changing the voters vote. Even so the voter must protect his own privacy for example, keep the code card secret. This phase starts by the issuing a random election challenge which is used for the system to be end-to-end verifiable. This is done by the with a distributed random number generator used by the trustees which is then validated by the Election Commission. The election commission then generates a hash of the number, the electoral roll and the ballot list which is the election challenge. At this point the voters can start voting by connecting to the Ballot Box using a SSL/TLS connection done by the client application. The voter receives the challenge and the list of candidates and inputs the vote code for the chosen candidate. The choice and the election challenge is sent to the VST who checks if the vote code is valid, and computes the vote and the ballot challenge from the election challenge. The VST also computes a receipt where each candidate has a verification code and the chosen candidate has the verification code equal to that of the confirmation code on the voters code card. The proofs of the creation of the verification codes are concatenated to create a receipt validity. After the confirmation of the voter that the receipt is correct, the signed vote (which has the ballot challenge), the verification receipt and the receipt validity is sent to the Ballot Box who verifies the vote and the receipt validity. If everything is correct then it sends it to the Bulletin Board where all receipt validity data is verified and all votes and receipts are validated by a signature over the list of all the voter-ballot-vote-receipt associations. In the last phase Independent Organizations start by verifying the election public data and commit that verification by signing the list of all the voter-ballot-vote-receipt associations. The voter can also verify their vote by the use of a verification service from an Independent Organization who will give a copy of the receipt to the voter. The tally is finally computed on the votes that were not protested by the use of a homomorphic aggregation of the votes by the Electoral Commission. Each Independent Organization can also verify the vote aggregation and tally decryption proofs.

2.4 Closest Systems

In this section we are going to analyse in depth two systems that implement secure electronic voting. These systems are sElec [10] and Helios [22, 23, 8]. These are remote electronic voting systems which feature end-to-end verifiability and are supposed to be implemented in environments with low risk of coercion.

2.4.1 sElect [10] is a lightweight verifiable remote voting system. This system has two main ways of verifying votes, human verification and automated verification.

The systems is composed of the bulletin board, the voters, the voters supporting device which is normally the browser and platform where this is running on, the authentication server, a number of mix servers and the voting authority.

There are also some assumptions made on the system. One of these is that the channels between the voter supporting devices and the authentication server are authenticated. As such we can assume that only authorized voters can cast ballots. The other assumption made is that for each voter supporting device there exists one authenticated channel and one anonymous channel to the bulletin board.

The protocol has the following phases which we are going to describe in more detail:

- Setup phase
- Voting phase
- Mixing phase
- Verification phase

During the setup phase the election parameters are posted on the bulletin board by the voting authority. Every server generates their keys which are then used for signing messages. The mix servers also generate another pair of keys to use in their encryption method. The public keys are then posted on the bulletin board.

During the voting phase is when the voters cast theirs votes. During this phase there are also generated the verification codes used in the human verifiability which are discussed further ahead. This verification code is a concatenation of a nonce given by the user (it is important the nonces chosen by different user have a small probability of being the same) and a nonce generated by the voter supporting device. After the verification code is formed the voter supporting device encrypts the vote with the verification code and sends it to the authentication server and if this ballot is valid then the server sends and acknowledgement. In the case of revoting the authentication server sends the old acknowledgement. If the authentication server does not accept the first vote, the voter supporting device tries again and if it does not succeed then it files a complaint. After this phase two lists are published on the bulletin board. One with the valid ballots and another with the identifiers of all voters who have cast a valid ballot.

Next is the mixing phase. Like a normal mix network, each mix server will process the ciphertext list which contains the ballots. During this it will check that the input it received is in lexicographic order, has the correct format, is signed and that it does not contain duplicates. If one of these conditions does not occur it sends a complaint to the bulletin board and stops. This step is called input validation. The mix servers also have a processing step where they decrypt the input with their private key, remove the duplicates and order the result. In case the decryption produces an unexpected result or cannot be decrypted then that entry is discarded. The result is then signed and posted on the bulletin board.

Finally is the verification phase. It is on this phase that the two types of verification occur. In human verification or voter verification, the voter simply checks that the voter nonce appears next to the choice. In case the voter did not vote, then they can check the list in order to make sure that their identifier is not there. In the automated verification or the voter supporting device verification, it is the voter supporting device that verifies the vote and it is triggered when the voter checks the result of the election. This type of verification depends on the honesty of the device. The voter supporting device first checks if the original submitted plaintext is present in the results published. If not the device checks if the ballot is present in the list delivered to the authentication server. If it is not there then the voter supporting device publishes the acknowledgement receive by the authorization server thus proving that the culprit is the server. In case the ballot is present in the authentication server then the device checks if the ballot (now encrypted) is present on the result list of the first mix server. If it is then it then checks the second mix server and does the same to the remaining mix servers. In case the ballot is not on one of the lists then the voter supporting device anonymously (in order not to tell how the voter voted) sends a message to the bulletin board which demonstrated that the mix server in question did not function correctly. It is said that the voter accepts the election if the voter nor the voter supporting device complains about the results.

2.4.2 Helios [22] is an open audit web based voting system meaning that anyone, even a person who cannot vote for that election can audit the election and verify that everything is running accordingly. This voting system is intended for use in low coercion elections and as such it does not offer much coercion resistance.

The first version of Helios is based on the Benaloh vote-casting approach [24]. Privacy is guaranteed only if you trust the server, but on the other hand integrity is guaranteed even if the server is corrupt since anyone can verify the election. Helios is composed of three main components:

- A Ballot Preparation System
- A Bulletin Board
- A Sako-Kilian mix-net [25] based on the El Gamal re-encryption.

The ballot preparation system is the component that allows for the encryption of the votes. It records the voters choices in the election and encrypts these, committing the ballot by displaying the hash of the ciphertext created. In case the voter wishes to audit that ballot the ballot preparation system displays the ciphertext and the randomness used during the encryption. The ballot preparation system then ask the voter to generate a new encryption for the choices. When the voter chooses to seal the choices the ballot preparation system discards the plaintext and the randomness and asks for the voter to authenticate himself. In case the authentication is successful then the encrypted vote is recorded as the voters vote.

On the Bulletin board of the Helios System, the votes are displayed next to a voter identifier be it an username or an identification number. All of the processing data is also available to everyone on the bulletin board in order for them to download and verify the election.

For anonymization Helios uses a Sako-Kilian mix-net based on the El Gamal re-encryption. Each server of the mix-net receives a number of inputs and re-encrypts them. Each server also creates a "shadow mix" which is used to prove that the mix encrypted the input correctly. A verifier can then ask the shadow mix for the encryption factors and permutations which a dishonest mix server only has 50% chance of guessing correctly. In order to increase integrity, each mix server has more than one shadow mixes thus decreasing the probability of guessing. Since the protocol above is complex it is transformed using the Fiat-Shamir heuristic [26] in which the challenge bits are computed as the hash of all shadow mixes. The decryption of the El Gamal encryption is using the Chaum-Pedersen protocol [27]. The result of these proofs is then posted on the Bulletin Board.

The overall process of the protocol can be described as the following:
The voter can create as many ballots as he wants and verify these. When he is satisfied with the audits, the vote is encrypted with a new randomness and he authenticates himself which casts his vote. The bulletin board posts the encrypted vote next to the voters name to the public. As the election ends, the votes are shuffled and the shuffling process can be verified. After some time the votes are decrypted and a tally is calculated. The results and all the data is finally published on the bulletin board.

Still some security problems arise with this implementation of Helios:

- Helios can cast votes for users by changing the ciphertext of a voter or authenticating himself as one of the voter since he knows the usernames and password of the voters.
- Helios may also show the voter a corrupt ballot thus leading the voter to thinking that he is voting for a candidate when in reality he is voting for another person.
- A dishonest voter may replay the vote another voter.

Some of these problems problems are mitigated with the use of the auditing tools available to the public thus decreasing the probability of these types of error not being detected.

In version 2 of Helios [23] the protocol has some differences. In this version of the protocol there is more than one trustee which generate a public key pair (based on the additive property and distributed encryption of El Gamal [27, 16, 28]) and an election officer who publishes the election parameters on the bulletin board. These parameters are the public part of the trustees' key, the proof of correct construction, the candidate list, the list of authorized voters and the election fingerprint which is a hash of all the previous parameters. The protocol then runs as follow:

1. The voter gets the election parameters and calculates the fingerprint which he can check with the one available at the bulletin board
2. The voter votes creating a ballot encrypted with the trustees' public key and proof that the vote is permitted. This ballot is displayed to the voter.
3. The voter can audit the vote like in the previous version.

4. When the vote is to be cast, it is sent to the election office who authenticates the voter, checks if the voter is authorized to vote and checks the proof and publishes the vote like in the previous version.
5. Anyone can verify the ballots in the bulletin board are permitted votes.
6. After some time the election officer combines the ballots still encrypted and publishes the tally which is also encrypted. This tally can also be verified.
7. Each trustee decrypts part of the tally and publishes proof of correct decryption.
8. Finally the election officer decrypts the tally and publishes the result.

The third version of Helios is only as extension of the previous where there were added some practical features. Still this implementation has a major flaw. The information displayed on the bulletin board corresponds to the voter id, the votes ciphertext and two signature of knowledges. The problem is that an adversary might submit a ballot with the same ciphertext and signatures as another voter. Since the signatures are valid the system will count this vote as legitimate. If we take the simplified case in which there are only three voters and two of them are honest, the third voter could choose one of the previous voters vote and recast it as his own and thus he would know who that voter voted for breaking the vote privacy property. To avoid easy detection the adversary might even change the ballot without changing the vote by exploiting the malleability of ballots.

Some of the solutions proposed for this problem are:

- Ciphertexts and signatures of knowledge have unique representations.
- Ballots should not contain the same ciphertext, and those that contain the same ciphertext should be rejected.
- Add the identity of the voter could be used in the creation of the signature of knowledge.
- Votes could be bound to the private key of the voter.

An implementation to solve this problem was made with the name of Helios-C [8] which stands for Helios with Credentials. It is composed of five main components (figure 5):

- The election administrator who publishes the eligible voters, the list of candidates and the result of the tally.
- The registrar who creates and distributes the credentials of the voters.
- The trustees who tally and publish the final result.
- The voters.
- The bulletin box where information is published and processed.

This implementation uses an El Gamal encryption in order to utilize the homomorphic properties present in it, uses the Schnorr signature scheme [29] and non interactive zero knowledge proofs to prove the correctness of the algorithms. It also consists of eight algorithms which are Setup, Credential, Vote, Validate, Box,VerifyVote, Tally and Verify.

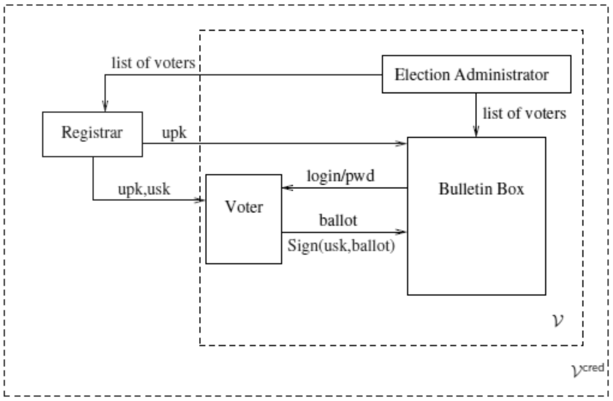


Fig. 5. Example of verifiable system with credentials from "Election Verifiability for Helios under Weaker Trust Assumptions" [8]

During the setup phase the public and private keys for the election are created and the list of credentials is also initialized empty. Then for each voter, a public and private key is also generated and adds the public key to the list of credentials thus ending the Credential phase.

In the vote phase the vote is encrypted using the election public key and a proof is computed to prove that the vote is valid. A signature is then made on the ciphertext and proof using the voter secret key. The ballot is then submitted as the voters public key, the encrypted vote and proof and the signature of the two previous parameters.

The validate phase is used to check if the vote is valid. It first checks if the public key is in the credentials list, the proof created during the voting phase is verified and so is the signature. The Box phase is when the vote is submitted in the bulletin board. This only happens if the voter authenticated successfully with an id. In case a vote is already present with the same id or public key as the vote that is about to be posted then the previous vote is deleted and the new vote is posted. If there is no such vote then the new vote is simply posted on the bulletin board. The votes are discarded in case the public key of the voter is not on the credentials list, if the vote did not pass the validate algorithm or if the ciphertext of the vote is the same as the previous one. The VerifyVote is used to check if the vote is present in the bulletin board.

In the tallying phase, the validation algorithm is passed on every ballot again and the voters public credential is also checked to see if it is in the credentials list. If this fails the election outputs an empty result. After this validation, the tally is computed relying on the homomorphic properties of the El Gamal encryption where the result should be in an interval between zero and the number of legitimate voters. Finally a proof is created. These results are validated in the Verify algorithm which does the validation of the previous phase again, creates the ciphertext and check the validity of the proof created before.

Inspired by Helios and Helios-C there was also another system created named Belenios[1][9]. This system uses the simplicity of the Helios system and adds the credentials of Helios-C, with the difference from the last one in that it doesn't use a threshold for the distributed shared key used by the trustees.

2.5 Analysis

We will now have a brief overview of all the protocols discussed and analyse the advantages and disadvantages of these. In the case of Scantegrity II and Pret à Voter, they rely on the person being present at the location where the voting is taking place. For anonymization both these protocols make use of mix network, which require a number of servers and good management of the distributed resources. Although these systems have these disadvantages they provide an easy way to audit ballots and to verify that the vote was counted. Remotegrity being an extension to Scantegrity II would force the implementation of all the Scantegrity II backend and as such force the use of mix networks whose disadvantages are mentioned above. It also uses the Scantegrity II ballots and another special code sheet which must be delivered by third party which could compromise the voter privacy in case someone got access to these beforehand.

EVIV provides a great way of maintaining user privacy but at the cost of the user having to carry a device to maintain the token used for the encryption process of the ballot and vote. It has the advantage of using code voting which increases the usability of the system during the verification phase of the protocol as it is easier for the voter to simply check if the correct code is presented to him or not. Similarly VeryVote uses the same idea of code voting used in EVIV but the code sheet must be delivered trough a third party means. This, like in the case of Remotegrity, can compromise user privacy. Another problem with this system is that the entity who creates the code sheets is the election server and as such, during the election it has enough information in order to associate a code sheet to a voter and as such it could discover who voted for whom.

Civitas is one of the first protocols to provide a remote e-voting protocol which is coercion resistant. For this it uses the idea of fake credentials that the voter can create with an algorithm which is given.

The protocol sElect introduced the idea of an automated verification system when the user would check the results of the election. This increased the percentage of votes that were verified since previously not all users would verify that their vote was counted. Although this automatic verification brings advantages it also makes the need of the user trusting the device that is verifying the vote, in this case the voters computer.

Helios presents various advantages and disadvantages depending on the version in question. The first version of Helios provided an open-audit system with the use of a single server, but it also had problems in which the voter had to trust the computer he was voting from and the Helios server. It was also possible to do ballot replay attacks where a dishonest voter could cast the same ballot

[1] http://belenios.gforge.inria.fr/

Table 1. Summary of Advantages/Disadvantages of different protocols

Protocol	Advantage	Disadvantage
Scantegrity II	Distributed shared key for election key End-to-End verifiable	Presencial voting Use of mix networks
Remotegrity	Remote RemoteCode Voting to assure correct ballot is sent End-to-End verifiable	Uses paper ballots sent by third party Extension of Scantegrity
Prêt À Voter	Easy vote verification End-to-End verifiable	Presencial voting Use of mix networks
Civitas	Coercion resistant with use of fake credentials	Use mix networks
VeryVote	Code voting End-to-End verifiable	Code sheet needs to be sent by a third party Use mix networks The server can know who voted for whom
EVIV	Protects user privacy in unsafe computer Uses code voting End-to-End verifiable	Use of external device to maintain token used in encryption
sElect	Automated Verification System End-to-End Verifiable	Use of mix networks User must trust the computer
Helios v1	Only needs a single server End-to-End Verifiable	Use of mix networks User must trust Helios server and computer Possibility of ballot stuffing Possibility of ballot replay attack
Helios v2/3	Same advantages as Helios v1 Use of homomorphic encryption instead of mix networks Distributed election key among trustees	User must trust Helios server and computer Possibility of ballot stuffing Possibility of ballot replay attack
Helios-C	End-to-End verifiable Can be verifiable if only one server is dishonest Solves replay attacks and ballot stuffing	Uses more than one server

as another voter, or for the Helios server to cast ballots to itself. It also made use of mix networks which were then dropped in the next versions. The second and third version started using a homomorphic encryption scheme, thus allowing anonymization without the use of extra servers and also started using distributed shared keys for the election key which made it impossible for only one person to decrypt the ballots. Although these issues were addressed there were still problems from the previous version that carried on, like the ballot stuffing. These versions of Helios also suffered from usability issues that are discusses by Fatih Karayumak et al. [30]. From the ballot stuffing problem Helios-C was created which introduced the credentials idea to the Helios system. This solution solved the ballot replay attacks and the ballot stuffing problem from previous versions of Helios and removed the single point of failure the system had, which was the Helios server being dishonest. The downside was the necessity of another server to create credentials for each voter.

3 Solution

In this section we are going to present the solution to the problem presented in section 1. We start by providing a detailed description of the protocol and then provide a overview of how the system is organized.

3.1 Protocol

An election will be divided in several phases. The first phase is the **setup phase**. During this phase the election administrator will setup the election with the help of the trustees. The election key will be created during this time using Exponential El Gamal encryption scheme and the election secret key is shared among all the trustees using a shared threshold in order to avoid a single all powerful entity, and as such, for a decryption to occurs it is necessary a number of trustees to collude. The public election key is published on the bulletin board. During this time it is also uploaded to the system a comma-separated values (CSV) file with information regarding the various eligible voters. This information must contain at least the voter id, which corresponds to the Fenix id, the voter name, the voters email address and the questions which this voter is authorized to answer.

After this phase starts the **registration phase** in which an email with a link is sent to the voters in order for these to confirm if they want to vote electronically. By default the voter will not vote electronically and as such he must confirm if he does. If such confirmation is done the voter is moved to another electoral register the is only for electronic voters. Credentials are also generated during this phase using a signature scheme similar to the used in Belenios and are sent to the voter by mail. In the end of the phase, there will be two electoral registers, one for voters who will vote by paper and another for those who choose to vote electronically who have also received the necessary credentials. The registrar who creates the credentials will not keep the secret credentials and the link between the public credentials and the voter is lost.

During the **vote phase**, the voter can access the link with the credentials sent to his mail and vote on the question he is authorized to answer. The votes are encrypted with the election public key and signed using the credentials secret key. A Non Interactive Zero Knowledge proof is also computed to prove that the vote is a valid vote. This is all aggregated in a ballot, with the public credential, which is sent to the bulletin board. A hash of the ballot is created so and used as a fingerprint which the user can then use to verify in the bulletin board that his vote is cast. As the ballot is received the server checks if the ballot is valid, verifies all the proofs and checks if the voter had already voted or not by verifying if the public credential had already been used. If so then the vote is discarded. The voter will receive a fingerprint of the ballot in order for them to check on the bulletin board that the ballot was received as it should have been. There will also be implemented a automatic verification system similar to that used by sElect in order to greatly increase the number of votes that are verified in the election. For the voters who are going to vote by paper, the process shall remain

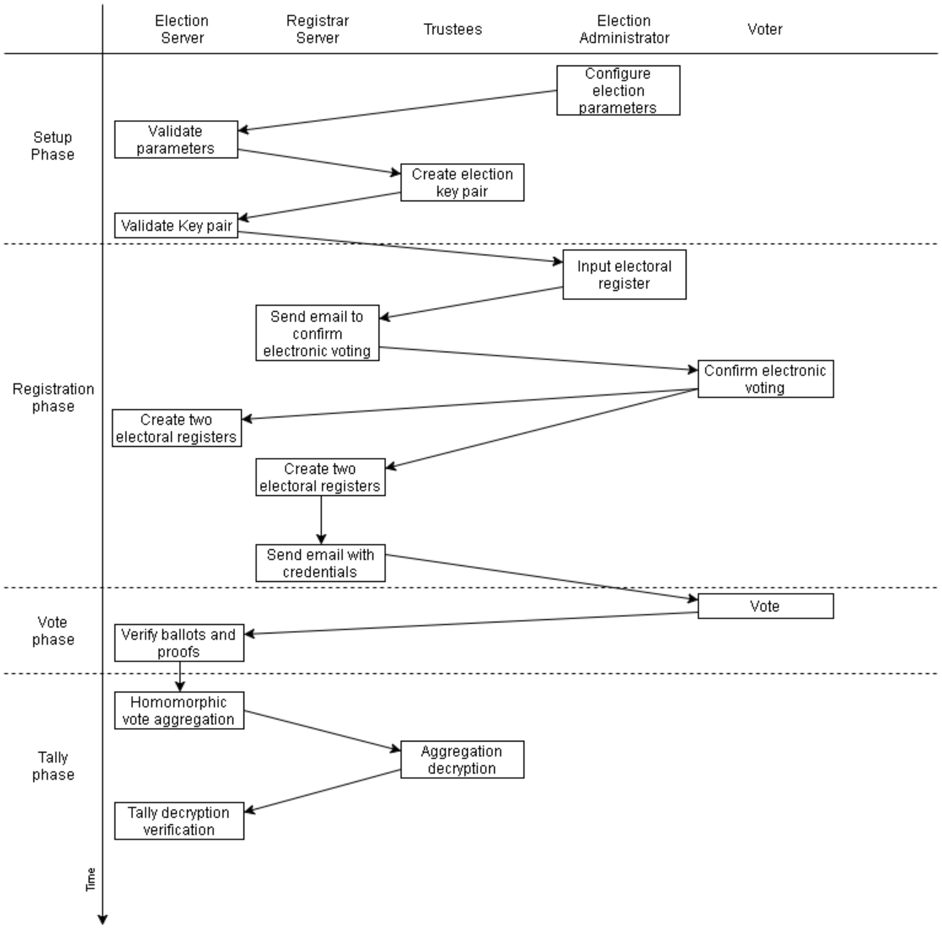

Election Server
Registrar Server
Trustees
Election Administrator
Voter
Setup Phase
Configure election parameters
Validate parameters
Create election key pair
Validate Key pair
Registration phase
Input electoral register
Send email to confirm electronic voting
Confirm electronic voting
Create two electoral registers
Create two electoral registers
Send email with credentials
Vote phase
Vote
Verify ballots and proofs
Tally phase
Homomorphic vote aggregation
Aggregation decryption
Tally decryption verification
Time

Fig. 6. Simplified schematic of the protocol that is going to be used.

the same. The poll worker who takes notes of the voters who voted will have an interface in the bulletin board where he can check if a voter has voted or not.

As the **tally phase** occurs the ballots are once again validated. Then using the additive homomorphic properties of the Exponential El Gamal Encryption Scheme the tally ciphertext is computed. The trustees then work together in order to decrypt the final tally. During all of the cryptography operations proofs are issued and data is logged in order to verify every process in the election.

A possible extension to the protocol is the addition of receipt freeness. An election protocol is considered receipt free if a voter cannot prove to another person that he voted in a particular way. This is similar to the privacy property but works in the case in which the voter is willingly working with the attacker. For example, Helios is only considered receipt free if the voter does not reveal the randomness used on the ballot. One possible way to extend our solution is to use a similar method that was done in BeleniosRF [31].

3.2 Overview of the architecture

The solution to the problem is based on the Belenios and Helios-C systems. These systems are e-voting systems which make use of credentials and as such the architecture is similar to that of a generic credential e-voting system.

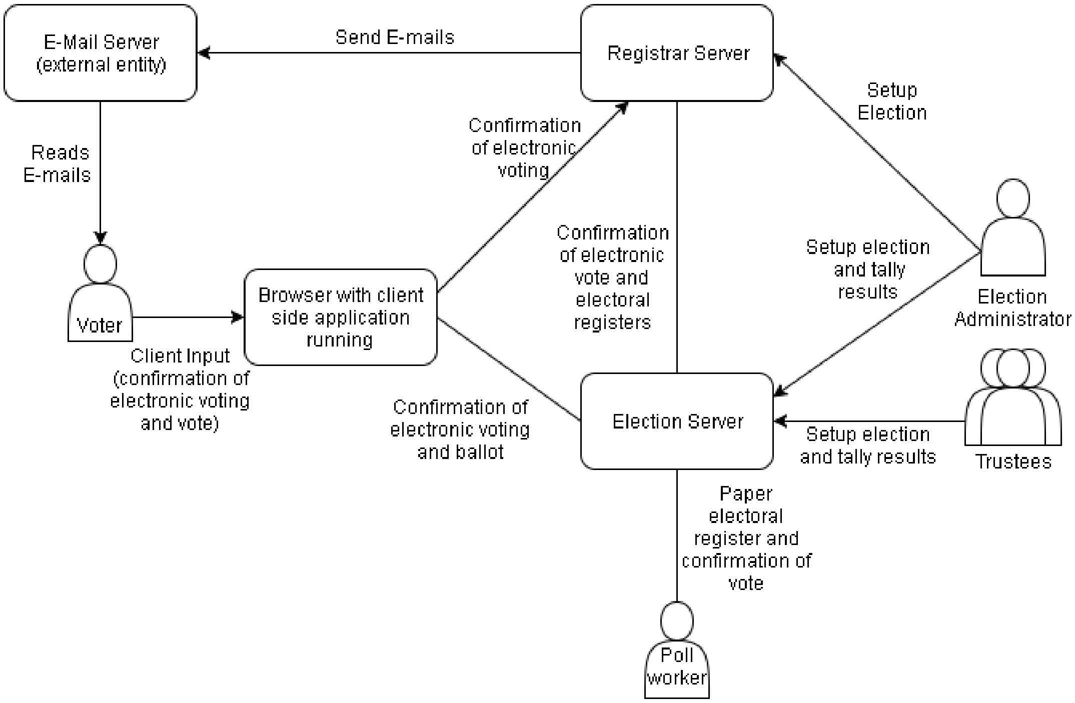

E-Mail Server (external entity)
Send E-mails
Registrar Server
Reads E-mails
Confirmation of electronic voting
Setup Election
Confirmation of electronic vote and electoral registers
Setup election and tally results
Voter
Browser with client side application running
Election Administrator
Client Input (confirmation of electronic voting and vote)
Confirmation of electronic voting and ballot
Election Server
Setup election and tally results
Trustees
Paper electoral register and confirmation of vote
Poll worker

Fig. 7. Simplified architecture of the solution and some information flow

As seen in figure 7, the architecture can be divided in three main components:

- The registrar server;
- The election server;
- The voting client.

The Registrar server is the server with the main focus of creating the voter credentials. It will divide the electoral register in two and create credentials for each voter that is going to vote electronically. It communicates with the Election server in order to pass information of the electoral registers. The Election Server is where the main computations of the system will occur. It will have a public bulletin board which the voters can access and will be the server which is responsible for storing the votes and counting these. Finally the voting client is an application which will run client side on the voters browser in order to give the voter a simple to use cast votes.

There are also four main entities in the system which are:

- The voter is the person who is casting the vote;
- The Election Administrator is the person who manages the election and its parameters;
- The trustees are a group of people who are trusted and share the election secret key;
- Poll Worker is the person who is working on the poll both for the paper ballot voting.

The connection between the different agents is going to be made using a SSL/TLS connection in order to provide message freshness and privacy during communication. The bulletin board will have a RestAPI in order for the elections to be configured remotely. There is also going to be a web client created using web languages (HTML, CSS and Javascript) and the servers will be done using the Java programming language. Authentication of the users will be done with the use of Oauth2 in order to connect to the Fenix platform.

4 Evaluation

In this section we will discuss the evaluation done to the project. There is going to be a practical evaluation and if there might also be additional evaluation made to the protocol due to the changes imposed by the mixture of paper voting.

4.1 Functionality Evaluation

For the practical evaluation a mock election is going to be made. This mock election will have a controlled environment where the users will have the option of choosing an arbitrary option to some questions and then are going to be asked to fill out a survey with the answers to those questions and some usability answers as well. The answers given in the survey are then used to compare to the election results to guarantee that all the answers were counted (this would not happen in a real life election) in order to evaluate the correctness of the system. Usability answers in the survey will be used in order to improve user experience with the overall system. For the usability we will also measure the time an user takes to vote, the number of steps it takes to cast a vote. The time it takes to verify a vote is also going to be taken into account and the simplicity to do so.

There will also be a theoretical evaluation on the minimum number of users that need to verify their votes in order for the election results to be reliable.

4.2 Security Evaluation

4.2.1 Protocol Adjustment Evaluation Due to the addition of paper ballots and the automated verification system to the protocol there might be some loop holes where attacks might be made on the system. To this end a theoretical evaluation on the differences of the system should be made in order to verify that the system remains secure according to the requirements specified for this work.

4.2.2 Implementation A code audit will be done as the development occurs and at the end of the development in each of the components in order to verify that the system is according to the requirements present in subsection 1.1.

4.3 Portability Evaluation

Due to time constraints, portability evaluation is outside of the scope of this work. However the system and documentation will be made available, in an effort to expand its adoption to other organizations.

5 Scheduling of Future Work

Future work is scheduled as follows:

- Feb 1 - Apr 30: Development of the system architecture;
- May 1 - May 15: Practical Evaluation of the system and results analysis;
- May 15 - May 31: Usability improvements and other improvements from the evaluation results;
- Jun 1 - Jul 1: Write the dissertation;
- Jul 2: Delivery of the dissertation.

During the development of the system architecture, the theoretical evaluation of the protocol adjustments will be made in parallel in order to minimize the design problems and correction costs.

6 Conclusions

E-voting systems are more used nowadays and facilitate the management of the election and the act of voting for each voter. As being systems that deals with critical information they have strong security requirements specially confidentiality and integrity. As seen in the report, we analysed some system which dealt more with confidentiality like Civitas and EVIV while others deal more with integrity like is the case with Prêt à Voter.

Having the systems analysed in mind we propose the implementation similar to Helios-C and Belenios in Instituto Superior Técnico in order to facilitate elections in an academic environment. The main challenges are the implementation and maintaining integrity and confidentiality of the protocols while also allowing the classic paper voting and by using the authentication system present in Fenix.

Finally although theoretically the protocol has good properties, attention should be given during the implementation to avoid possible code errors that might break those properties and attention should also be given to the additions on the protocol as to not make it less usable and insert possible security flaws.

It will also be important to develop the project with portability in mind in order for easy adoption and integration in other organizations with a similar environment.

Acknowledgments I would like to thank professor Ana Almeida Matos and professor Jan Cederquist for all the help they have given me during the semester with the thesis project. I would also like to thank professor Carlos Ribeiro, professor Paulo Mateus and Luís Guerra e Silva for the meetings and help in the research for the thesis. Finally I would also like to thank my friends and family for always being there for me and being supportive.

References

1. Chaum, D., Carback, R.T., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity ii: End-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Transactions on Information Forensics and Security **4**(4) (2009) 611–627
2. Zagórski, F., Carback, R.T., Chaum, D., Clark, J., Essex, A., Vora, P.L.: Remotegrity: Design and use of an end-to-end verifiable remote voting system. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7954 LNCS** (2013) 441–457
3. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: The Prêt à Voter Verifiable Election System. 1–13
4. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security Analysis of the Estonian Internet Voting System. Proceedings of the 21st ACM Conference on Computer and Communications Security (May) (2014) 12
5. Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II municipal election at Takoma Park: the first E2E binding governmental election with ballot privacy. Proceedings of the 19th USENIX conference on Security (2010) 19–35
6. Ministry of Local Government and Modernisation: The e-vote trial. https://www.regjeringen.no/en/historical-archive/Stoltenbergs-2nd-Government/Ministry-of-Local-Government-and-Regiona/tema-og-redaksjonelt-innhold/kampanjesider/e-vote-trial/id597658/ Accessed: 2016-12-23.
7. Adida, B., Marneffe, O.D., Pereira, O., Quisquater, J.J.: Electing a university president using open-audit voting: analysis of real-world use of Helios. . . . on Electronic voting . . . (i) (2009) 1–15
8. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election verifiability for helios under weaker trust assumptions. Lecture Notes in Computer Science (including

subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **8713 LNCS**(PART 2) (2014) 327–344

9. Glondu, S.: Belenios specification. (2013) 1–8
10. Kusters, R., Müller, J., Scapin, E., Truderung, T.: SElect: A lightweight verifiable remote voting system. Proceedings - IEEE Computer Security Foundations Symposium **2016-August** (2016)
11. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. Proceedings - IEEE Symposium on Security and Privacy **7875**(May 2007) (2008) 354–368
12. Oppliger, R.: How to Address the Secure Platform Problem for Remote Internet Voting. {SIS'02}, 5th Conference on "Sicherheit in Informationssystemen" (2002) 153–173
13. Joaquim, R., Ribeiro, C., Ferreira, P.: VeryVote: A voter verifiable code voting system. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **5767 LNCS** (2009) 106–121
14. Neff, C.: Practical high certainty intent verification for encrypted votes. VoteHere document (2004)
15. Gamal., T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms (1985)
16. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. European transactions on Telecommunications **8**(5) (1997) 481–490
17. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2) (1981) 84–90
18. Joaquim, R., Ferreira, P., Ribeiro, C.: EVIV: An end-to-end verifiable Internet voting system. Computers and Security **32** (2013) 170–191
19. Joaquim, R., Ribeiro, C.: An Eficient and Highly Sound Voter Verification Technique and its Implementation. (103) (2012) 1–16
20. Bhiogade, M.: Secure Socket Layer. Computer Science and Information Technology Education Conference (June) (2002) 85–90
21. Dierks, T.: The transport layer security (tls) protocol version 1.2. (2008)
22. Adida, B.: Helios : Web-based Open-Audit Voting. 335–348
23. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. Journal of Computer Security **21**(1) (2013) 89–148
24. Benaloh, J., Benaloh, J.: Simple verifiable elections. Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop (2006) 5–5
25. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth- (1995)
26. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. Procedings of Crypto (1986) 186–194
27. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology (1993) 89–105
28. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Annual International Cryptology Conference, Springer (1994) 174–187
29. Schnorr, C.P.: E cient Signature Generation by Smart Cards. Coexistence (March) (1991) 1–22

30. Karayumak, F., Olembo, M.M., Kauer, M., Volkamer, M.: Usability Analysis of Helios: An Open Source Verifiable Remote Electronic Voting System. Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections (2011) 5
31. Cortier, V., Fuchsbauer, G., Galindo, D.: Beleniosrf: A strongly receipt-free electronic voting scheme. IACR Cryptology ePrint Archive **2015** (2015) 629