E-Voting on Fenix

Fernando Marques
Intituto Superior Técnico
Lisboa
fernando.m.marques@tecnico.ulisboa.pt

ABSTRACT
This article considers the problem of implementing an e-voting system on Instituto Superior Técnico, integrating this to the already existing university information system Fenix. As such, the work presented works under the assumption that coercion resistance is not of primary concern. It is required that the system be able to integrate the already present paper voting protocol in order to provide to those who have less experience or less trust in the e-voting system a means in which they may exercise their right to vote. This article presents a hybrid e-voting system, H-Belenios, which is a combination of two existing protocols. This protocol is then integrated with the already existing authentication system present at Instituto Superior Técnico. It presents the two protocols that served as a baseline for the final solution, the specification of the hybrid protocol and some implementation details. In the end, it presents some results over the analysis done and the conclusion, which include some relevant future work.

KEYWORDS
e-voting, hybrid, Belenios, protocol

1 INTRODUCTION
Voting has been one of the main ways for decision making in the current times and has been for some time now. It is used by many to make decisions being these small and sometimes insignificant to very important ones which can impact the way society works, for example government election or referendums[19]. For a long time, society has used the paper voting system, in which ballots are sheets a paper where the voter casts their vote, put them in a ballot box and then someone counts them being the result publish later. This system has worked for ages and although it works, it also has some disadvantages, for example how can a voter check that their vote was actually count correctly?

Electronic voting, also denoted as e-voting, came in order to so try and solve some problems which existed with paper voting and although it has some advantages, it also has some disadvantages. The main advantages that can be seen from e-voting is the possibility of voting remotely (thus people who have low mobility or even those who could go the the voting booth due to some restriction are able to cast a vote), it reduces the time and effort of managing an election also cutting on staff cost, but most of all some protocols allow the voter to verify that their vote is being counted correctly. Although this may only seem to bring advantages, e-voting also brought some disadvantages. One of those is that vote coercion is easier in e-voting, another is that a protocol which is not implemented correctly may bring about security flaws that could impact the result of an election.

In the end, it is not correct to say if e-voting or the classical paper voting is better since both has their advantages and disadvantages but the truth is that e-voting has gain popularity over the year and better protocols have been made in order to tackle the disadvantages it has.

1.1 Objectives
E-voting has been used in many situations already, from governmental elections[19], the municipal elections[5] and even some university elections[4, 15]. Currently at IST elections are still done in the classical paper format. This has some difficulties in organizing the election in different campuses and some voters may not be on campus during the days of the election. There is also the problem of a cultural shift when changing to a full e-voting protocol. As such this article tackles the problem of implementing an e-voting system on a academic environment, in this case, IST where it will be connected with the already present informatics campus management system Fenix in order to provide authentication of users.

The main objective is to specify and implement a hybrid e-voting system to be used in an academic environment, in this concrete case in Instituto Superior Técnico.

1.2 Requirements
The requirements which were considered relevant for such a system can be divided in three main types:

1.2.1 Security Requirements. These are the requirements which have to do with the security of the system. Being a voting system, these are of primary concern. The main requirements which should be kept in mind are the following:

1. The votes must remain anonymous in the system, meaning that it must not be able to associate a vote with the person who voted.
2. The system must provide integrity of the ballots which are saved before the tallying.
3. The system should ensure a high availability and reliability.
4. The system should be end-to-end verifiable.
5. No person should have the full power to decrypt the election data.
6. Only eligible voters should be able to vote.
7. The system shall count all the votes and do so correctly.
8. No voter should be able to vote more than once.
9. All steps of the election will be available to audit by either the voters or the election administrator.

1.2.2 Functionality Requirements. These are the requirements which affect how the system will function and feel to the user. The requirements are:
1.2.3 Architectural requirements. These requirements are those that affect the architecture of the system. These can be due to hardware limitation, documentation limitation or other. The architecture requirements which were considered relevant are:

(1) The user must be able to vote in a computer with an internet connection, preferably a secure 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) The design and implementation shall be documented and available to the public.

1.3 Contributions
As such, in this thesis, a specification and implementation of hybrid e-voting protocol called H-Belenios, which uses as baseline protocols the current paper voting protocol used at Instituto Superior Técnico [16] and the Belenios e-voting protocol [13], is presented.

As such, we can summarize the main technical contributions as:

- A new hybrid voting system that integrates a modified version of the Belenios e-voting protocol (in which voting credentials are generated on demand) and a classical paper voting protocol.
- Accurate enunciation of the preservation of properties of the baseline protocols and their proofs.
- Proof of concept of applicability to an academic setting, by the design of an architecture for implementing the proposed solution that fits the implementation requirements of an existing organization.
- Implementation of a working prototype of the system.
- An usability evaluation in order to prove that users would adopt and use such a system.

A paper explaining the protocol specification was published in INForum 2017 Atas do Nonô Simpósio de Informática [10]. The code of the implementation and a wiki which will be developed as the implementation evolves will be available in the github project page [1].

1.4 Overview
Section 2 explains the protocol used for the creation of the hybrid solution. A brief explanation on how the protocols work is presented and some assumptions and security properties of these are also shown.

Following this section, a specification of our hybrid solution is shown. During this, each phase of the protocol is explained in detail. Section 4 follows were it is proven that the changes made on the baseline protocols don’t affect the correct function of these when working together in our hybrid solution.

The next section presents some implementation details, where the technologies used are described, some cryptographic details are shown and finally a more close relation between the implementation and the specification of the protocol is done.

In order to test the implementation usability tests were done and are presented in section 6. In this section, a reflection on the results on these tests is also done.

In the end of this article, a conclusion is presented on the overall work.

2 BASELINE PROTOCOLS
The solution to the objectives describe in 1.1 is the combination of two already existing protocols. The protocols that serve as a baseline for the solution are the already existing paper voting protocol which is used in Instituto Superior Técnico and the Belenios e-voting protocol[15]. This last one chosen due to its simplicity, good security and the low requirements in regards to server number. During this section, these protocol are described in some details, include some of the properties and assumptions these have.

2.1 Current Paper Voting Protocol
Currently at IST voting follows a specific protocol which is well documented. This protocol changes for some specific elections, being these changes also documented on the document[16].

The protocol used for paper voting at IST can be divided in to the following five sections:

Set up phase. Before the election begins, the members of the electoral commission, including the president, are designated. These members cannot be candidates for the election and include, in particular, one member per election candidate which serves as a representative.

On the day of the election, one or more voting tables are available for depositing votes. Each voting table has a group of pool workers (which may include one representative per candidate) designated to it.

Voting phase. In order to vote, the voter first shows his identity to the voting table president, who also verifies his voting right. The voting secretary notes on the electoral roll that this voter has voted and the president hands out a paper ballot. The voter goes to a voting booth, and fills in the ballot secretly by placing an ‘X’ next to the candidate of choice.

Any other type of mark on the ballot invalidates it. Finally, the voter hands in the folded ballot to a person present at the voting table who inserts it into the ballot box.

Tally phase. At the end of the election the pool workers at each table proceed to count the votes and their distribution. The minutes are then written, recording the information gathered from the votes. The partial results are then handed to the electoral commission who will add all the votes from different tables and calculate the tally. The results must be released 24 hours after the end of the voting phase, and the paper votes must be destroyed 30 days after the election has passed.

Audit phase. After the tally phase complaints can be made to the electoral commission with a time limit of one day after the results are published. Every pool worker also has the
option of complaining in the table report against decisions made by that table.

From this protocol the following security properties and assumptions can be extracted:

Assumptions:

(1) No one, including the pool workers, interferes with the paper votes in the ballot boxes.
(2) The pool workers perform the tally and publication of the results according to regulations.

Security Properties:

(1) Only and all eligible voters are able to vote.
(2) Each vote is kept confidential (regarding both existence of vote and vote content, for a given voter), up to aggregation of results.
(3) The regulations specify a correct procedure for counting exactly one vote per voter.
(4) It is possible to tally the votes, within a specified timeframe.

2.2 Belenios

Inspired by Helios and Helios-C there was also another system created named Belenios\(^1\)\[13\]. 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 does not use a threshold for the distributed shared key used by the trustees. The following can summarizes the Belenios protocol with the exception of the credential recovery process:

Set up phase. An election in Belenios starts with the creation of the credentials. The server administrator sends an unique universal identifier (uuid) to the credential authority. There, a list of credentials and their public parts is created. Each credential is sent to the respective voter, and the list of shuffled public parts of the credentials is generated and sent to the server administrator.

Then, each trustee generates her own public key and sends it to the server administrator, who verifies that the trustee has the secret part of it, and in the end combines all of the trustees keys in order to generate the election public key. With both the list of public credentials and the election public key, the server administrator is able to create the election. The protocol uses a public key encryption scheme with additive homomorphic properties.

Voting phase. In order to vote, the voter obtains the election parameters, creates a ballot which is formed from the encrypted votes and proofs (proof of membership and proof that the vote is formed correctly), the signature and the election uuid, and obtains a hash of the ballot. If all the data present in the ballot is valid then the voting server publishes it.

Tally phase. The election is terminated by the server administrator. The homomorphic properties of the encryption scheme are used to aggregate all the votes into an encrypted tally which is sent to each trustee to perform a partial decryption. Finally, the server administrator verifies the partial decryptions and aggregates them in order to create the decrypted tally, which is then published.

Audit phase. During the voting phase and tally phase, each voter is able to verify that their vote was cast correctly using the smart ballot tracker (hash of the serialization of the ballot) which was published on the bulletin board when the ballot was cast. It is also possible to verify that the encrypted tally was calculated correctly. The greater the number of voters performing this verification, then stronger is the confidence in the outcome of the election.

From this protocol the following security properties and assumptions can be extracted:

Assumptions:

(1) Not both the voting server and the credential authority are simultaneously corrupt.
(2) The client software does not leak information about the electronic votes.
(3) The trustees will not work together in a malicious way.

Security properties:

(1) Only and all eligible voters are able to vote.
(2) Each vote is kept confidential (regarding vote content), up to aggregation of results.
(3) The protocol specifies a correct procedure for counting exactly one vote per voter.
(4) Each voter is able to verify that their vote was cast correctly.
(5) Each voter is able to verify that the tally is correct.

Another version of Belenios was proposed, called BeleniosRF\[7\] which proposed a system that allowed for a more efficient coercion resistance.

3 H-Belenios

The protocol designed in order to solve the problem presented in this article is a hybrid protocol which is called H-Belenios. This protocol is based on the Belenios e-voting protocol and the Helios-C e-voting protocol. It then integrates with the current paper voting system present at IST in order to minimize the changes to this last so that users do not feel the difference if they refuse to use the electronic part. The protocol allows for users to revote through paper in case they have already cast an electronic vote since the paper vote always has precedence over the electronic.

The entities who play a role in the protocol are the voting server VS, the credential Authority CA, the server administrator SA, each eligible voter V, each trustee T and the each pool worker PW.

In order to better describe the protocol, it also necessary to specify the structure where data is saved: the EL, which is the list of all the elections, the list of eligible voter list and trustee list which are EVL and TL respectively, the table of electronic votes TEV, the list of paper voters and its paper version which are denoted by LPV and pLPV, the table of electronic final electronic votes TFEV and the smart bulletin board SBB.

The protocol makes use of the exponential El Gamal encryption scheme for encrypting and anonymizing the ballots and the Schnorr signature scheme in order to sign the ballots.

\(^1\)http://belenios.gforge.inria.fr/
An election process can be divided into four phases which are the set up phase, the voting phase and the tally phase. The fourth phase is the audit phase that occurs in parallel with some of the other phases. Each of these phases is described below in more detail.

3.1 Set up Phase
This is the first phase of the protocol where the election is prepared. Similarly as in Belenios, the SA must first send to the voting server the election parameters (denoted as eParam). These parameters are the name of the election, the description of the election, the start and end dates and if applicable the start and end times for the ballot box (these determine the time interval in which a voter is allowed to cast a vote during the voting phase). As the server receives this information it generates a uuid for the election, saves this information to EL, publishes this information on SBB (denoted as publish), and generates the cryptographic parameters necessary for the election (two primes p and q where $p - 1 \equiv 0 (mod q)$, and $g$ which is a generator for $Z_p$).

The SA must also send TL to the voting server in order to start the key generation process. After this T can generate their own El Gamal key pair. The private key is kept secret by the trustee and the public key is sent to the server with a proof of knowledge of discrete logarithm. Each of the keys is publish on SBB in order for everyone to verify that the aggregation is done correctly. After all T have sent their key shares to the server, the administrator can verify each of the zero knowledge proofs (denoted as validate) and aggregate the keys of the trustees generating the election public key which will then be used to encrypt the ballots. This key is sent to the server and published on SBB in order to be verified by anyone who wishes it.

For the paper part of the protocol, the set up phase happens as it did previously without any change.

3.2 Voting Phase
The second phase of the protocol is the voting phase which occurs in during the dates defined by SA while creating the election. During this phase any $v \in EVL$ can cast a vote either by paper, electronically or both (paper votes have precedence). In case the voter wishes to vote electronically he must first register himself with the credential authority. To do so, he sends a request to $v$ who then send to CA the identifier of $v$ and the cryptographic parameters generated during the set up phase. CA then generates the credentials which are a key pair to be used in the schnorr signature scheme. These credentials are then sent to $v$ by email and the public part of the credentials is sent to VS. After receiving his credentials, $v$ will have access to the ballot box. Here he must insert his private and public credentials, choose his answers for the election and creates the ballot. The ballot is composed of the encrypted answers (done with the election public key), a digital signature done with the private credential, individual zero knowledge proofs that each encrypted answer belongs to a finite and proof that the overall answer also belongs to a finite set. As the ballot is formed, a hash of the ballot is computed, which corresponds to the smart ballot tracker, and $v$ saves this. Finally $v$ can cast his ballot to VS, which verifies the zero knowledge proofs, saves the ballot to TEV and calculates the hash over the ballot and publishes it on SBB.

In case $v$ wishes to vote by paper he may do so as it currently occurs. He goes to a poll booth to vote, proves his identity to PW and votes by paper. The only difference happens at the end of this phase, in which PW must give to SA pLPV who then submits the electronic version LPV to VS.

3.3 Tally Phase
After the voting phase ends, then begins the tally phase. VS calculates TFEV which is the table of final electronic votes. On this table are present all of the votes of voters who didn’t cast a paper vote. VS is able to identify these from the public credentials which are associated with both a ballot and the voter. These votes are then aggregated using the additive homomorphic properties of exponential el Gamal. Following this, T must partial decrypt this aggregation and send it to VS with a zero knowledge proof of correct decryption. SA can verify these proofs and aggregate the partial decryption forming the electronic tally which he send to VS. SA also submits the tally of paper ballots to VS. After all the verifications are done VS can publish the results of SBB.

3.4 Audit Phase
The audit phase occurs in parallel to the phases described above and in a similar fashion as in Belenios. At the end of the election all the voters will be able to check that their smart ballot tracker is present in SBB. For those voters who voted both by paper and electronically, since the vote that is counted is the paper one, then their smart ballot tracker will not be present there, instead appears a note saying that they voted by paper.

A checksum is also added in order to verify that the total number of votes is equal to the number of valid smart ballot trackers and the number of votes done by paper (given by |finalResult| = |TFEV| + |LPV|). This enables to detect any mismatch between the number of counted paper votes and those that appear in the smart ballot tracker.

During the key generation process, the ballot casting and the decryption process, zero knowledge proofs are also used and verified in order to provide assurance of the data being submitted, which are the same as the ones used in Belenios.
Table 2: Voting phase specification for electronic voting

<table>
<thead>
<tr>
<th>#</th>
<th>Message Exchange</th>
<th>Voting Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \nu \rightarrow \text{VS} : Id_v ), request</td>
<td>( \text{EVL} := \text{EVL}[Id_v \mapsto P_{cv}] )</td>
</tr>
<tr>
<td>2</td>
<td>( \text{VS} \rightarrow \text{CA} : Id_v, p, q, g )</td>
<td>validate((P_{cv} \notin \text{TEV}))</td>
</tr>
<tr>
<td>3</td>
<td>( \text{CA} \rightarrow \text{VS} : P_{cv}, Id_v )</td>
<td>( \text{TEV} := \text{TEV} \cup {(P_{cv}, e\text{Ballot})} )</td>
</tr>
<tr>
<td>4</td>
<td>( \text{CA} \rightarrow \nu : P_{cv}, Sc_{cv} )</td>
<td>publish((\text{TEV}))</td>
</tr>
<tr>
<td>5</td>
<td>( \text{VS} \rightarrow \nu : e\text{Param}, p, q, g, P_{ke} )</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( \nu \rightarrow \text{VS} : e\text{Ballot} = {\text{ans}, s{\text{ans}}<em>Sc_v, \delta</em>{\text{ans}}, P_{cv}} )</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( \nu \rightarrow \text{VS} : \text{eParam} = p, q, \delta )</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( \text{VS} \rightarrow \nu : \text{eBallot} = {\text{ans}, s{\text{ans}}<em>Sc_v, \delta</em>{\text{ans}}, P_{cv}} )</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Voting phase specification for paper voting

<table>
<thead>
<tr>
<th>#</th>
<th>Message Exchange</th>
<th>Server &amp; Administrator</th>
<th>Voting Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \nu \rightarrow \text{PW} : Id_v )</td>
<td>( \text{pLPV} := \text{pLPV} \cup {Id_v} )</td>
<td>validate((Id_v \in \text{dom}(\text{EVL}) \setminus \text{pLPV}))</td>
</tr>
<tr>
<td>2</td>
<td>( \nu \rightarrow \text{PW} : p\text{Ballot} )</td>
<td>( \text{LPV} := \text{pLPV} )</td>
<td>( \text{BB} := \text{BB} \cup {p\text{Ballot}_id} )</td>
</tr>
<tr>
<td>3</td>
<td>( \nu \rightarrow \text{PW} : p\text{Ballot} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( \text{PW} \rightarrow \text{SA} : \text{pLPV} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( \nu \rightarrow \text{PW} : p\text{Ballot} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( \nu \rightarrow \text{PW} : \text{pRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>( \nu \rightarrow \text{PW} : \text{pRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>( \nu \rightarrow \text{PW} : \text{pRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>( \nu \rightarrow \text{PW} : \text{pRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>( \nu \rightarrow \text{PW} : \text{pRes} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Tally phase specification

<table>
<thead>
<tr>
<th>#</th>
<th>Message Exchange</th>
<th>Server &amp; Administrator</th>
<th>Voting Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \nu \rightarrow \text{PW} : Id_v )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( \nu \rightarrow \text{PW} : p\text{Ballot} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( \nu \rightarrow \text{PW} : \text{pRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>( \nu \rightarrow \text{PW} : \text{eRes} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 H-BELENIOS CORRECTNESS

Some relevant security properties for H-Belenios are now presented. First it is necessary to express what are the assumptions and definitions which are considered.

Assumptions: The union of those enunciated for the baseline protocols A (BP-A) and B (BP-B).

Definition 1. Final vote of an eligible voter

The final vote of an eligible voter is either the paper vote (if the voter cast a paper vote), the last vote cast electronically (if the voter did not cast a paper vote), or undefined (if the voter did not vote).

- Not defined, if the voter did not vote.
- The vote expressed on the paper vote, if the voter issued a paper vote.
- The last vote issued electronically, otherwise.

Property 1. Only and all eligible voters are able to vote.

Proof.
- All eligible voters can vote. If an eligible voter wants to vote by paper he can do so as in BP-A. If a voter wishes to vote electronically he can do so, as in BP-B, from any web browser so long as he registers to vote electronically and authenticates himself in the process.
- Only eligible voters can vote. As in BP-A, a voter has to prove his identity in order to vote by paper. If a voter wants
to vote electronically, he must register as such using a web browser, as in BP-B. To do so he must authenticate himself using his authentication credentials and to cast a vote he must be authenticated and have both credentials that are sent to him when he registers himself to vote.

Property 2 (Confidentiality). Each vote is kept confidential (regarding vote content), up to aggregation of results.

Proof. Confidentiality (regarding vote content) of votes up to publication of the results is guaranteed for paper votes in the same manner as for BP-A, and for electronic votes as for BP-B. The publication of LPV, as well as the construction of TFEV does not reveal vote content.

Property 3 (Integrity). The protocol specifies a correct procedure for counting exactly one vote per voter.

Proof. (1) The protocol specifies a correct procedure for building a list of paper voters LPV and a table of final electronic votes TFEV.

Indeed, from BP-A it is possible to conclude that LPV is correctly built, since this is not altered in any way. Each time a voter votes electronically, his ballot is added in TEV (line 9 in table 2). These ballots have a connection to the ids of the voters via the public credential. Finally, TFEV is constructed from TEV and LPV (line 1 in table 4) by removing from TEV the ballots cast from voters who have their ids in LPV. Given that LPV and TEV are correct, it is possible to conclude that TFEV is also correct.

(2) The protocol specifies a correct procedure for tallying tables LPV, TEV and TFEV.

(3) All and only final votes are counted.

- Only final votes are counted: According to line 10 of the table 4 the final result is composed from the decryption of eRes, which is the aggregated encrypted result, and pRes. pRes is the tally of the paper votes, calculated from the votes present in the physical ballot boxes as in BP-A. The tally of the BBs counts exactly one vote per voter, as in BP-A.

- From the definition of final vote it is possible to conclude that a final vote is either in paper or in electronic format, but in only one of these formats.

- If the final vote is in paper, then according BP-A, it is assumed that the vote is counted and that according to the correctness of TFEV, if an electronic vote from the same voter was cast, it will not be counted.

- If the final vote is electronic, from the correctness of TFEV it can be concluded that it will be counted.

- All final votes are counted: According to definition 1, a final vote is either in paper form or in electronic format. If the vote is in paper form then according BP-A, it is assumed that the vote is counted correctly. If the vote is electronic then, according to correctness of TFEV, it is be counted correctly.

- It is defined in the the specification, that the number of cast votes, paper and electronic, are added and compare to the number of paper voters and the number of final electronic votes which corresponds to the number of electronic voters who did not cast a paper vote. These results need to be equal in order for the election to be valid and to prove that each final vote was counted at most once.

(4) An eligible voter will have exactly one counted vote if and only if he has issued at least one vote.

(a) If an eligible voter issued a vote, he will have a final vote. Indeed, if he issued a paper vote, this will be his final vote. Otherwise, the vote is electronic, in which case it will also be a final vote.

(b) Each eligible voter will have at most one final vote. Indeed, as with BP-A, a voter cannot cast a paper vote more than once. By definition of final vote, even if a voter has both paper and electronic votes, then only the paper vote is the one that is considered for the election.

Property 4 (Auditability). It is possible to retally the paper votes, within a specified time frame. Furthermore, assuming that the tally of the paper votes is correct, every voter is able to verify that the final tally is correct.

Proof. The partial tallies (paper and electronic votes) can be verified as in BP-A and B. The combination of both enables to verify the final tally.

Coercion resistance: While BP-A is considered to be strongly coercion resistant, BP-B suffers from the fact that vote receipts can be produced. Although this is assumed to not be of primary concern in the context of this work, it is possible to observe that as a result of the integration of the protocol with a paper-based voting system, the new protocol improves on the level of coercion resistance that is provided by BP-B. Indeed, the voter now has a choice of also casting a paper ballot, which is preferred over the electronic ballot during the tally. As such there is no way to prove who he voted for, up to aggregation of the results.

Relaxing assumptions. The integrity properties of BP-A rely on strong assumptions of correctness of the actions of the poll workers. However, these assumptions can be relaxed in face of the new possibility of detecting a mismatch between the counted number of paper votes and the electronic votes published in the smart bulletin board. More precisely, it is no longer necessary to assume that fake ballots or votes cannot be added to or removed from the ballot boxes or tally, but only that they are not changed.

5 IMPLEMENTATION

The implementation was done in using the Python programming language and the Django Framework. These technologies since they allowed for a fast development of web applications which are secure and reliable. Django provides out of the box some features regarding security and authenticity which were considered useful.

https://www.djangoproject.com/
As the technologies used for the front-end, common web languages (HTML, JavaScript, CSS) were used. The code and a wiki which is going to be developed as the project advances will be available in the projects github page\(^3\).

5.1 Cryptographic Details

In this section some of the cryptographic details of the implementation are presented. Details are given on how the cryptographic parameters are generated, how the random number are generated in the front-end, the zero knowledge proofs and others.

5.1.1 Cryptographic Parameters Generation. As mentioned in section 3.1, the VS needs to generate some cryptographic parameters which are then used in the encryption and signature schemes. These parameters are of high importance because if they are generated incorrectly, it could lead to security issues, like information leak which could compromise voter privacy or ballot replay which could lead to incorrect election results.

These parameters are generated on the VS for each election that is created. The first parameters is a 160 bit prime number which is denoted as \( q \). To generate this parameter, first a random number is generated using the secrets module from the python standard library. This module provides random number generators which are said to be cryptographically secure, or in other words, are able to provide secure random numbers to be used in cryptographic schemes. After generating this value, it is test to check if it is prime. The system uses the Miller-Rabin primality test which is run 11 times (accuracy value).

After generating a value number for \( q \) it is time to generate the value \( p \). This value is closely related to \( q \) since we can calculate \( p \) using \( p = q \times 2 + 1 \). After having also generated \( p \) and verified that it is indeed prime it is finally time to generate \( g \). This value is a generator for the of the cyclic group with \( p \). In order to generate this number, first a primitive root of \( p \) is discovered. This is a value that verifies \( a^{p−1} \equiv 1 \pmod{p} \) where \( f \) is a factor of \( p − 1 \). After having calculated a primitive root, it is possible to calculate the last parameter as \( g = a^{\frac{p−1}{f}} \pmod{p} \) where \( a \) is the primitive root. These cryptographic parameters are then used in the Exponential El Gamal and the Schnorr signature scheme.

5.1.2 Random Number Generation on the Front-End. During the encryption process and signing processes, random number generation is also necessary. This brought a problem since these operations occurred in the front-end and JavaScript doesn’t work well with very big numbers since it loses precision due to storing them as 64-bit floating points. As such the Big Integer library from Tom Wu was used in order to do these operations. Another problem with JavaScript is that it is not prepared to do cryptographic operations and as such, it does not provide a built in random number generator which is considered cryptographically secure. In order to generate the random numbers, the Web Crypto API [2] was used since it provided a secure way to generate said numbers and is present in most modern browsers.

For the generation of the Schnorr signatures, their verification and the creation of the Smart Ballot Trackers used for the verification of the integrity of the ballot, SHA256 hashes were used. In order to implement generate these, the Stanford JavaScript Crypto Library was used since it provided an implementation of these hashes. Although it also provides a random number generator, this needs to be feed entropy from external sources, like key presses and cursor movement. This poses a problem cause it may not be random enough to provide good entropy. Another problem is that since these cryptographic operations are CPU intensive, they block the UI and as such were executed on different threads using web worker technology (via VkThread plugin). These web workers are run in a separate environment and as such these environments does not have access to the UI thread which holds the values necessary for generating entropy.

5.1.3 Zero Knowledge Proofs. Zero knowledge proofs were used in order to prove that certain operations were done correctly without disclosing sensitive information regarding the operations itself. The zero knowledge proofs used were:

- Proof of knowledge of discrete logarithm, which is used to prove to the SA that the trustee has the secret key for the public key share he submitted.
- Proof of correct decryption, that is used to prove to the SA that the trustee knows the secret key for decryption a message encrypted with his public key(or part of it in this case).
- Proof that a discrete logarithm belongs to a finite set, which is used to prove to the VS that the ballot cast by a voter is valid. This proof is run for each individual answer of a question(individual proofs) and to the group of answers to a question(overall proof).

More information on these proofs is available on Some ZK security proofs for Belenios by Pierrick Gaudry [12].

5.2 Specification Fulfilment

Through this section, some implementation details will be presented and compared side by side to the specification of the protocol present at section 3. As such this section will be divided in subsections, each representing the phases of the protocol, and each subsection will deal with the main details regarding that phase.

5.2.1 Set Up Phase. On the start of this phase the SA inputs the election data to VS. This last calculates the cryptographic parameters as shown in section 5.1.1 and saves this information in a Election object (lines 1-2 in table 1). A list of all the Election objects stored in the database is the equivalent of the specifications EL. Since the specification follows one election, let’s assume that the uuid generated for this election is denoted by electionId. When an election is created, CA is also informed of the election and receives its id and the end date so that it can delete all the information regarding it 30 days after the end date defined. It is also during this phase that the trustee can input the list of all trustees and the list of all eligible voters (lines 4 and 11 of table 1). Each trustee creates a Trustee object which has a connection to the Election object created earlier. The same can be said for each voter, except they are created as Voter objects. As such TL can be defined as the

\(^3\)https://github.com/imarques94/E-Voting-on-Fenix
list of Trustee objects which have the foreign key for the Election object with key electionId and EVL can be defined the same way, except with Voter objects instead of Trustee objects.

The trustees perform the key share generation process during this time. Each public key share and it’s proof is stored on their objects, published (lines 5–6 of table 1) and retrieved by the SA in order to perform the verification of the proofs (lines 7–8 of table 1) and aggregation of the election public key. This public key is stored in the Election object and is also published (lines 9–10 of table 1).

The structures present in this phase can be described as the following sql queries:

1. **EL** = SELECT * FROM Election;
2. **TL** = SELECT * FROM Trustee WHERE election = electionId;
3. **EVL** = SELECT * FROM Voter WHERE election = electionId;

### 5.2.2 Voting Phase

During this phase the voter can vote electronically, by paper or both forms. The last ballot counted is always the paper one if both are present. If the voter votes by paper, his id will be submitted to the system as a paper voter (line 8 from table 3). As such, a query will be done in order to find his Voter object and the flag which says that he voted by paper will be set to true. The sql equivalent is the following assuming istXXXXX is the identifier of the voter:

```sql
UPDATE VOTER SET paperVoter=TRUE WHERE identifier = 'istXXXXX' AND election=electionId;
```

This creates the structure **LPV** which in sql can be denoted as following:

1. **LPV** = SELECT * VOTER WHERE paperVoter=TRUE AND election=electionId;

If the voter voted electronically, he must first request a credential (line 1 from table 2). This request is sent from VS to CA (line 2 from table 2) who sends both credentials to the user via email (line 5 from table 2) and send the public credential to VS which is then stored on the Voter object (lines 3-4 from table 2).

After receiving their credentials, the voter is able to vote. All the information needed to create the ballot is loaded before hand so that the voter is able to create it even offline. Each selected answer is chosen as value 1 and each non-selected answer as value zero. After choosing a option for every question, the ballot is encrypted, proofs are calculated, a signature is formed and the smart ballot tracker is calculated (which the voter should keep). The ballot is then sent to VS (lines 7 from table 2). When receiving the ballot, VS checks that the public credential it received is the same as the voters, checks if there is a ballot with the same public credential already present or not and checks the proofs and the signature on the ballot. If everything is correct, then the ballot is stored in a Ballot object which has a connection to the Election object (lines 8–10 from table 2) and VS creates a Smart Ballot Tracker for that ballot. A list of all the ballots objects that correspond to a specific election is the equivalent of the specifications **TEV**. The sql equivalent of this structure is:

1. **TEV** = SELECT * FROM Ballot WHERE election = electionId;

### 5.2.3 Tally Phase

During the tally, the VS creates the **TFEV** which is a list of all electronic ballots without those of paper voters (line 1 from table 4). To do this, the system first gets the list of all public credentials from the Voter objects which have the paperVoter flag set to True. After getting all those public credentials, the system queries all Ballot objects and removes from that list all those whose credentials are included in the list calculated earlier. This is done each time it is needed to access this list. The sql equivalent would be something similar to this:

```sql
TFEV = SELECT * FROM Ballot WHERE publicCredential NOT IN ( 
SELECT publicCredential FROM Voter WHERE paperVoter = True 
);
```

After the SA inputs the paper voters he will also need to input the paper results which are stored in the Election object (lines 3 and 9 from table 4). He also requests to the VS to aggregate **TFEV** which is done using additive homomorphic aggregation (line 2 from table 4). As such for each ballot, each answer for each question is aggregated to the equivalent of the next ballot, thus calculating the encrypted aggregate tally. This is also stored in the Election object.

Each trustee then fetches this aggregated encrypted tally and calculates the partial decryptions and proofs for these (lines 4–5 from table 4). These are stored on the Trustee object which the SA accesses in order to perform the validation of the proofs and aggregation of the partial decryptions, which gives the electronic tally (lines 5–6 from table 4). The electronic tally and paper tally are then added by the SA and sent to VS to be published. This are stored once more in the Election object (lines 10–12 from table 4).

### 5.2.4 Audit Phase

The audit phase is a phase that occurs in parallel with the previous phases. The main ideas to retain from this phase are:

- **Voter** can check their Smart Ballot Trackers in the bulletin board. One Smart Ballot Tracker is created on the front-end and the other on the VS. Since these are the equivalent of hashing the ballot, if both these Smart Ballot Trackers which were generated on different places are equal, then it is possible to conclude that the value that generated (the ballot) is also equal. This allows to prove that the ballot was cast as cast.

- At the end of the election, everyone will be able to export the election data in order to verify that the election occurred correctly. This creates a json from the pertinent information which is stored in the objects present in the database.

- The voter can perform the audit of their ballot before casting it. Since this requires to reveal the randomness used during the encryption process, the ballot needs to be generated with another randomness to be cast again. When audited, the ballot is check to view that the answers were the ones the voter wanted, the encryption was well done, the proofs were calculated correctly and that the signature over the ballot is correct. This allows to prove that the ballot is cast as intended since the script performing the auditing is independent from the script performing the generation of the ballot.
6 USABILITY TEST
In order to evaluate the usability of the system, user testing was done. The servers were both deployed on the same machine running a developer server for each of the applications. Since the evaluation was run inside a closed network no secure channels were used. The user group who participated in this evaluation consisted of 15 individuals whose ages ranged from 20 years old to 55. This was so that the evaluation would received different feedback from different age groups. An user was chosen in order to serve as a trustee for the election. This user had an explanation on how the protocol worked and what was his role. The rest of the users were informed on how the system worked in order for them to have an understanding of what was going to happen.

The election had the following questions to which the users had to answer:

**Which of these movies would you like to see?**
- Dark Tower
- It
- Other

**Which of these books do you like best?**
- Lord of the Rings
- Moby Dick
- 1984
- None of the above

In order to simulate a real world scenario, it was informed that the votes present in table 5 and 6 would be inserted into the the system. This would simulate a scenario in which both paper and electronic voting was occurring at the same time. The total number of simulated paper voters was 12, thus making the total number of voters in the system 27.

**Table 5: Simulated paper votes for question “Which of these movies would you like to see?”**

<table>
<thead>
<tr>
<th>Which of these movies would you like to see?</th>
<th>Simulated Paper Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark Tower</td>
<td>3</td>
</tr>
<tr>
<td>It</td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table 6: Simulated Paper Votes for question “Which of these books do you like best?”**

<table>
<thead>
<tr>
<th>Which of these books do you like best?</th>
<th>Simulated Paper Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lord of the Rings</td>
<td>5</td>
</tr>
<tr>
<td>Moby Dick</td>
<td>3</td>
</tr>
<tr>
<td>1984</td>
<td>2</td>
</tr>
<tr>
<td>None of the above</td>
<td>2</td>
</tr>
</tbody>
</table>

As each user executed the task of voting, they would share their answer being this recorded in order to compare the recorded results with the final election results calculated by the system. In the end of the election the following questions were posed on the users:

- Did you audit a ballot?
- Did you verify your ballot was cast correctly?
- Did the system feel fluid?
- What would you improve?
- Would you use the system instead of the classical paper voting?

For the user who had the job as a trustee, it was also asked if he tough the job would be intuitive.

6.1 Discussion
This section presents a reflection on the results presented for the evaluation present in section 6.

Firstly, it is possible to see that the protocol is working correctly from the results present in tables 7, 8, 9 and 10. On both questions a total number of answers done were 27, which corresponds to the 15 votes from the users who participated on the evaluation and the 12 simulated users. It is also possible to verify that the protocol takes priority on paper votes over electronic votes from the user who cast both votes. If both votes had been counted then the results present would have 12 votes for “Dark Tower” in the question “Which of these movies would you like to see?” and would have 10 votes for “None of the above” which would make a total of 28 votes for each of the questions.

**Table 7: Recorded Electronic Votes for question “Which of these movies would you like to see?”**

<table>
<thead>
<tr>
<th>Which of these movies would you like to see?</th>
<th>Recorded Electronic Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark Tower</td>
<td>9</td>
</tr>
<tr>
<td>It</td>
<td>2</td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table 8: Recorded Electronic Votes for question “Which of these books do you like best?”**

<table>
<thead>
<tr>
<th>Which of these books do you like best?</th>
<th>Recorded Electronic Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lord of the Rings</td>
<td>4</td>
</tr>
<tr>
<td>Moby Dick</td>
<td>0</td>
</tr>
<tr>
<td>1984</td>
<td>3</td>
</tr>
<tr>
<td>None of the above</td>
<td>8</td>
</tr>
</tbody>
</table>

**Table 9: Tally for question “Which of these movies would you like to see?”**

<table>
<thead>
<tr>
<th>Which of these movies would you like to see?</th>
<th>Tally</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark Tower</td>
<td>11</td>
</tr>
<tr>
<td>It</td>
<td>7</td>
</tr>
<tr>
<td>Other</td>
<td>9</td>
</tr>
</tbody>
</table>
Table 10: Tally for question "Which of these books do you like best?"

<table>
<thead>
<tr>
<th>Answer</th>
<th>Tally</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lord of the Rings</td>
<td>9</td>
</tr>
<tr>
<td>Moby Dick</td>
<td>4</td>
</tr>
<tr>
<td>1984</td>
<td>5</td>
</tr>
<tr>
<td>None of the above</td>
<td>9</td>
</tr>
</tbody>
</table>

From the results of the questions done after the election it was possible to verify that only 20% of the participants audited their ballots. This can be due to this being an optional part of the test and as such, in a real world scenario, a larger part of the user may decide to verify their ballots. The same can be said for the verification of correct cast of the ballot. Either way, it would be a good idea to reinforce the idea that these steps are important to increase the trust of the final results of the election. One way to do this is to, in the UI, present the importance of those optional steps to the voters. This approach can also be used in order to emphasise the importance on the trustee key share and the voter credentials.

In terms of improvements requested, the UI was a main aspect. The current UI is very minimalistic and as such in future version should contain more information for the user and be more intuitive. Important pitfalls to avoid are some that were presented by Karayumak et al. in "Usability Analysis of Helios - An Open Source Verifiable Remote Electronic Voting System", for example inconsistencies in the system in regards to wording and use of buttons and links. Another problem which was mentioned in the paper and also revealed during the user test is the use of copy paste mechanism during the ballot auditing. This can be prone to error for less experienced users who, due to lack of technical skill or simple distraction, not copy the whole ballot json or even insert a foreign character which would case the auditing to fail. One proposed solution would be to allow the user to download a file with the information and then upload it to the server for auditing, but a study would firstly need to be conducted on such approach due to the possibility of leaking information. The last improvement mentioned by the users was the possibility of credentials recovery. This is already implemented in the original Belenios protocol and should be able to be ported over to H-Belenios. One important detail to take into account would be the necessity to delete the previous credential and its association with the user id and possible previous ballot so that the relation of one to one from voter to credential is maintained.

Finally, and one of the most important questions, was whether the users would use such a system to cast a vote. 60% of the users answered that they would use such system while 20% said they would prefer only the paper voting. That leaves 20% who said they would not mind the feature since it could be useful. That gives the system a total of about 80% of approval. Since this is a small test group, these results may still vary a lot, but for now it gives a good perspective in term of the possibility of implementing such system.

7 RELATED WORK

Internet voting protocols. The electronic part of the hybrid e-voting protocol is based on Belenios. Belenios is a purely e-voting protocol which is in turn based on Helios [3] and Helios-C [8].

Helios was created by Ben Adida in order to allow voters to audit their ballots before casting them. Various versions of Helios have been released, incorporating new security features such as the replacement of mix-networks [6], used in the anonymization of the ballots, for the use of homomorphic encryption using the El Gamal encryption scheme [9]. One of the main differences between Helios protocol and Belenios is that the later uses a credential system in order to avoid ballot stuffing. Helios-C is an improvement over the Helios system, proposed by Veronique Cortier et al., which adds the idea of credentials. This protocol uses an El Gamal encryption scheme with homomorphic properties for anonymization and a Schnorr signature scheme [18] for ballot authenticity. The main difference between Belenios and Helios-C is that the former does not have threshold support for key generation.

Hybrid voting protocols. TrustVote [14] is a hybrid e-voting system which includes a paper-based and an electronic-based voting components, designed to be used by governments and organizations. The e-voting component is based on FOO92 [11], being one of the main improvements the introduction of threshold blind signatures. This system also allows the possibility of revoking a previous electronic vote. In order to integrate the paper voting with their e-voting system, paper voting is allowed during a new period that starts after the electronic vote casting ended. If the voter had cast an electronic vote, then during the paper voting he may chose to revoke it. In order to revoke an electronic vote, the voter must present his voter secret which is used in order to identify his e-vote. The tally is then computed with the sum of all electronic votes minus the sum of all revoked electronic votes and finally adding the sum of all paper votes. Unlike our system, paper voting and e-voting phases are thus kept separate in TrustVote. It is also necessary to submit additional documents in the act of paper voting.

8 CONCLUSIONS

This document presents a hybrid voting protocol that integrates an already existing e-voting protocol with the paper voting protocol which is already used. The cultural shift which is required in order to adopt a full e-voting protocol is reduced by minimizing the number of changes done on the paper voting protocol. This document also presents details regarding the implementation of said protocol and the results that were received from an usability test done with various users. From these results it was possible to verify that users were willing to adopt this protocol but that it still needed improvements in some areas. It is also important to note that although the solution presented was created with the concrete case of using it in IST, it is possible to adapt it in order to be used in other academic environments with similar requirements.

In conclusion, it was proven that work still needs to be done in order this system to be ready to be deployed on a real world scenario, but the results received were positive and the possibility of bringing e-voting to IST is nearer.
REFERENCES


