TÉCNICO LISBOA

# Analysis and Implementation of a Suitable E-Voting Solution for Universidade de Lisboa

**Eduardo Alexandre Silva da Costa**

Thesis to obtain the Master of Science Degree in

**Electrical and Computer Engineering**

Supervisor: Prof. Carlos Nuno da Cruz Ribeiro

**Examination Committee**

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Carlos Nuno da Cruz Ribeiro
Member of the Committee: Prof. Hugo Alexandre Tavares Miranda

**July 2021**

*I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.*

# Acknowledgments

My parents, Helena and Gonçalo, for their support over all these years by providing me encouragement, motivation and means to allow me to finish my degree.

My family, namely my grandparents José, Carmo and Iria for always being there for me and helping me to grow and evolve as a person.

My friends Miguel, Filipe, Claudionor, Nuno, João, André and everyone else for all the good times that we've shared together and more importantly for our friendship - may it outlast us.

My sincere thanks to João Sebastião and Eduardo Melo for all the team projects that we've spent countless hours working together, for every study session before the exams and for always being there for me during all these years.

Everyone that took some of its time to participate in the field test with a special mention to Hugo Miranda and Fernando Silva, which feedback was invaluable for the development of this project.

A special thanks to my supervisor Carlos Ribeiro for guiding and helping me whenever I felt lost, for always providing me insightful knowledge on the various topics addressed in this document and specially for all the effort made to make this project real.

My girlfriend Joana, who was kind and gentle enough to review this document, for all the support, strength and patience given that allowed me to keep focused and never lose track of my true priorities, my dearest thank you.

To everyone else that directly or indirectly contributed to this work, thank you.

# Abstract

This thesis consists of the analysis and implementation of a suitable e-voting system for Universidade de Lisboa. It is required that this system satisfies most of the voting system properties such as integrity, privacy and verifiability while also providing a simple interface to allow inexperienced users with no technical knowledge to use it correctly and confidently. It is also acceptable that the system does not guarantee non-coercibility since most of the elections that are expected to be performed in it have a low-coercive risk. This implementation is based on a version of Helios: a voting system that allows any willing observer to audit the entire process of an election and has already been used several times in real-world elections. The thesis then details modifications that were made in Helios in order to comply with the requirements specified by the Universidade de Lisboa. After adapting and customizing the system, a field test with real voters was orchestrated together with a form in order to evaluate and detect problems and adversities related to security, authentication, usability and accessibility that may have gone unnoticed. As such, further modifications are performed in the system to mitigate any detected problem and every unresolved issue is properly documented. This project has already been used in a real-world election which details can be found in this thesis. Finally, the conclusion and the current system limitations are presented, as well as future work to further improve the system.

# Keywords

e-voting; elections; Helios; cryptosystems

# Resumo

Esta dissertação consiste na análise e implementação de um sistema de e-voting adequado para a Universidade de Lisboa. É necessário que este sistema de e-voting satisfaça a maioria das propriedades dos sistemas de votação como integridade, privacidade e verificabilidade. Deve ao mesmo tempo fornecer uma interface simples para permitir que usuários inexperientes e sem conhecimentos técnicos possam usá-lo de forma correta e com confiança. Também é aceitável que o sistema não garanta a não-coerção, uma vez que a maioria das eleições que é expectável que sejam realizadas neste apresentam um baixo risco de coerção. Esta implementação é baseada numa versão do Helios: um sistema de votação que permite a qualquer observador disposto auditar todo o processo de uma eleição, já tendo sido usada várias vezes em eleições no mundo real. A tese detalha as modificações que foram feitas no Helios de modo a cumprir os requisitos especificados pela Universidade de Lisboa. Após a adaptação e personalização do sistema, foi orquestrado um teste de campo com eleitores reais em conjunto com um formulário para avaliar e detectar problemas e dificuldades relacionados com segurança, autenticação, usabilidade e acessibilidade que possam ter passado despercebidos. Como tal, mais modificações são realizadas no sistema para mitigar qualquer problema detectado, e todos os problemas por resolver encontram-se devidamente registados. Este projeto já foi usado numa eleição no mundo real cujos detalhes se encontram discriminados nesta tese. Por fim são apresentadas as conclusões e as limitações atuais do sistema, assim como trabalhos futuros para melhorá-lo ainda mais.

# Palavras Chave

e-voting; eleições; Helios; criptosistemas

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **AAUL** | Associação Académica da Universidade de Lisboa |
| **BMP** | Ballot Mark Pairs |
| **CN** | commonName |
| **DN** | Distinguished Name |
| **E2E** | End-to-end |
| **EPPN** | eduPersonPrincipalName |
| **EVIV** | End-to-end Verifiable Internet Voting |
| **FC** | Faculdade de Ciências |
| **GDPR** | General Data Protection Regulation |
| **IACR** | International Association for Cryptographic Research |
| **IdP** | Identity Provider |
| **IST** | Instituto Superior Técnico |
| **LDAP** | Lightweight Directory Access Protocol |
| **O** | organizationName |
| **OU** | organizationalUnitName |
| **P2P** | Peer-to-Peer |
| **PK** | Public Key |
| **POK** | Proof of Knowledge |
| **RDN** | Relative Distinguished Name |
| **SAML** | Security Assertion Markup Language |
| **SK** | Secret Key |
| **SP** | Service Provider |
| **SSL** | Secure Sockets Layer |

| | |
|---|---|
| **SSO** | Single Sign-On |
| **TLS** | Transport Layer Security |
| **SMTP** | Simple Mail Transfer Protocol |
| **UL** | Universidade de Lisboa |
| **URL** | Uniform Resource Locator |
| **UUID** | Universally Unique Identifier |
| **VST** | Voter Security Token |
| **XML** | Extensible Markup Language |

# 1

# Introduction

## Contents

Elections have a great impact on developing strong democracies, giving people a say in the way that they want to be governed. To do so, however, it is required that the election system guarantees the correctness of its results while still preserving the voter's privacy. The traditional paper based voting methods have no difficulty in guaranteeing these properties. Privacy is guaranteed because the voter marks his ballot alone in a private voting booth. Correctness is also guaranteed if the election's officials are trustful, since every vote is recorded by them. To guarantee this property, there should be an official for every running candidate watching the ballot box sealing, its opening and the vote count. The officials will then assure that every vote is recorded as intended by the correspondent voter and that is correctly included in the tally. Since the voters should trust at least the official affiliated to his chosen candidate, a trust chain is created that guarantees the correctness of the election.

Performing an election on an internet based remote voting system, also known as remote e-voting, is a much greater challenge than traditional paper based systems [4–6], since it's more complex to guarantee both the voter's privacy and the correctness of the results. Every voter casts the vote in his personal computer which cannot be trusted, since it can be targeted by attacks that can steal the voter's vote surreptitiously or leak his choice. However, many e-voting systems have been proposed on the last decades [7], tackling both of these problems and coming up with creative solutions to them, cf. Chapter 2.

## 1.1 Motivation and Objectives

The Universidade de Lisboa's (UL's) rectory was searching for a way to simplify the logistic operation of their periodic elections, since these require the setup of voting booths in each one of their faculties. The solution agreed was to implement and adapt an already existing e-voting system to solve this problem, allowing the voters to cast their votes remotely. Furthermore, the adapted e-voting system should also be capable of allowing any approved member of the UL community to create smaller elections, either for faculties, courses, nucleus or any other group of members.

This thesis focus on finding an already existing e-voting system, implementing and then adapting it, followed by a field test and further improvements of the solution, in order to mitigate any issues detected and to fulfill the UL's community needs.

## 1.2 Requirements

The solution chosen, besides following the standard e-voting properties stated in Section 2.1.1 must also follow some specific requirements imposed by the UL:

- The final solution must authenticate users via the UL's centralized authentication system and make

it possible to create elections without requiring to upload a list of voters. In order to do so, the system must be able to read and save the voter attributes provided by the authentication system.

- The solution may allow hybrid elections, i.e., elections that are both available to vote electronically and in paper, but it must also be able to perform completely electronic and remote elections.
- The chosen solution must have its technical documentation available and legitimate to ease its implementation and also to help identify any possible security issues and limitations.

Although it is not a requirement, it is valued that the solution is open-source, does not require the use of security tokens or code systems and has also been successfully used several times to run real-world elections.

## 1.3 Main Contributions

This thesis presents the analysis, implementation and customization of the Helios Voting System [7–9], an open-source project created in 2008 that has since then received updates and gathered many contributors. The original source code of this system can be found in GitHub [10]. This thesis implementation is currently deployed online [11] and it is capable to serve the entire UL community, allowing to create and participate in elections remotely while following the requirements stated in Section 1.2.

## 1.4 Organization

The next chapter describes some cryptographic and e-voting concepts, as well as already existing e-voting systems. Chapter 3 introduces the implemented solution, along with the modifications performed in it, ending by detailing the field test performed in the solution. Chapter 4 presents the issues detected in the field test, and the solutions implemented in order to mitigate such detected issues are explained in Chapter 5. This latter chapter also contains details and a brief analysis of a real-world election that took place using the implemented solution. Finally, Chapter 6 gives the final conclusions and future work.

# 2

# E-voting Concepts and Related Work

## Contents

This chapter presents the background research that was done for this thesis. Section 2.1 provides the general concepts that are used in the construction of cryptographic voting systems, followed by the types and a list of researched e-voting systems in Section 2.2 and Section 2.3, respectively. The chapter ends with an overview of Security Assertion Markup Language (SAML) authentication systems in Section 2.4.

## 2.1 E-voting Concepts

A voting system must ensure the correctness of the election's results while still preserving secret its sensitive information and individual votes. Designing an e-voting system while keeping a good balance between these two requirements can prove to be a difficult challenge, since improving one of them tends to weaken the other. In order to ensure both requirements, several techniques and methods are used to design these systems, some of them depicted in this section.

### 2.1.1 Voting System Properties

Every voting system is expected to satisfy some specific criteria [4, 12], such as:

- *Eligibility and Authentication* - Only voters that are correctly authenticated and authorized should be able to vote;
- *Uniqueness* - Each voter should be able to vote only one time;
- *Accuracy* - The cast votes should be recorded correctly by the system;
- *Integrity* - The cast votes should not be able to be modified or deleted and if by any reason they are, the system should be able to detect such event;
- *Verifiability and Auditability* - The system should allow to verify that every vote was correctly accounted for in the final election tally;
- *Reliability* - The election system should be robust without losing votes towards any possible system failure, resultant of either an incident or attack;
- *Secrecy and Non-Coercibility* - It should not be possible to reveal how any individual voted, and voters should not be able to prove how they voted;
- *Flexibility* - The system should allow a variety of ballot question formats and be accessible to people with disabilites;
- *Convenience* - It should be possible for a voter to cast a vote quickly and with minimal technical knowledge or skills;

- *Certifiability* - The election system should be testable;
- *Transparency* - The voters should be able to have a general knowledge and understanding of the voting process;
- *Cost-effectiveness* - The election system should be affordable and efficient.

For a new election system to be adopted and used in real world elections, it is likely that it satisfies most, if not all of the requirements stated above.

### 2.1.2 Public Key Cryptosystem

The goal of a public key cryptosystem is to allow anyone to send an encrypted message that can only be decrypted by one entity. For that, it uses asymmetric pairs of keys:

- *Public Key* - This key is known widely and used to encrypt the message;
- *Secret Key* - This key is known only by the entity able to decrypt the ciphered messages.

#### 2.1.2.1 ElGamal Encryption

The Diffie-Hellman key exchange [13] allows to securely exchange cryptographic keys over a public, insecure channel. The ElGamal cryptosystem [14] is an asymmetric key encryption algorithm based on the Diffie-Hellman assumption. It relies on the assumption that discrete logarithm problems with carefully chosen groups have no efficient solution. It is composed by three stages.

1. Key generation

   Generate a large prime number *p* and a generator *g* of a cyclic group *G* of order *p*

   Select a random integer *x* from $\{1, ..., p-1\}$

   Compute $y = g^x mod\ p$

   The public key (PK) consists of (p,g,y) and the secret key (SK) is *x*

2. Encryption

   Select a random integer *r* from $\{1, ..., p-1\}$

   Compute $c_1 = g^r mod\ p$

   Compute $c_2 = y^r M\ mod\ p$, where *M* is the message

   Send the ciphertext ($c_1$, $c_2$)

3. Decryption

   Compute $M = \frac{c_2}{{c_1}^x}\ mod\ p$

#### 2.1.2.2 Exponential ElGamal Encryption

The exponential ElGamal cryptosystem [15] is a variant of the ElGamal cryptosystem. The difference lies on the encryption, where the message *M* is encrypted as $g^M$. The difficulty of this cryptosystem lays in its decryption, since recovering *M* requires to solve a discrete logarithm, considered to be a hard problem. However, as long as *M* is small, the problem can be solved algorithmically.

### 2.1.3 Digital Signatures

The goal of digital signatures is to prove that a message was issued by a certain entity. Like in Section 2.1.2, it makes use of asymmetric pairs of keys, but in this case, the secret key is used to sign the message and the public key is used to decrypt the message. This guarantees the authenticity of the message, since the entity is the only one that can sign the message using its secret key.

### 2.1.4 Bulletin Board

Many systems use a public bulletin board [7] to have transparency in the electoral process. This is achieved by making the election public data available to the public. The bulletin board consists in a public data repository in which all published data is authenticated, usually by digital signatures. This bulletin board must assure availability and guarantee that the published data cannot be deleted. In some modern systems such as the EPFL E-voting solution [16, 17], blockchain is used to build the bulletin boards.

### 2.1.5 Blockchain

Blockchain [18] is a system developed for electronic transactions that doesn't rely on centralized trust, being constituted by a list of linked blocks, each containing a hash of the previous block in the blockchain, a timestamp and transaction data. These blocks are managed by a peer-to-peer (P2P) network, which is trustful if a majority of the computational power is held by nodes that are not cooperating to attack the network.

### 2.1.6 End-to-End Verifiability

End-to-end (E2E) verifiability [7, 19] aims to allow individual voters to verify the election results without requiring them to trust the voting system, the election officials or other voters. It can be divided in two different components:

- Cast as Intended: voters can verify that their selections are correctly recorded;
- Counted as Cast: any member can verify that every recorded vote is correctly included in the tally.

The two components described above can be achieved in three phases, each phase presenting challenges that are overcome in different ways by different systems:

1. Cast as Intended: voters get convincing evidence that their encrypted votes accurately reflect their choices as they are vote casting. This can be achieved, for example, using a Benaloh challenge [20, 21];
2. Recorded as Cast: the encrypted votes are included in a list on the bulletin board in order to allow the voter to check its inclusion in it. This verification can be made possible by giving ”receipts” to voters when they cast the ballot that are also posted in the bulletin board;
3. Counted as Recorded: any member can check that all the published encrypted votes are included in the tally, without ever knowing how any individual voted. This step can be accomplished by using mix-nets or a homomorphic encryption, both of them explained in more detail in Section 2.2.

### 2.1.7 Zero-Knowledge Proofs

In a zero-knowledge proof [7, 22], a prover *P* proves the validity of an allegation to a verifier *V* without revealing it. In a voting system, this can be used to prove the validity of an encrypted vote without revealing the encrypted candidate choice. Any zero-knowledge proof must achieve both completeness and soundness [23].

#### 2.1.7.1 Soundness and Completeness

Soundness states that if a statement is false, *P* cannot convince a honest *V* that it is true, except with some small probability. On the other hand, completeness states that if a statement is true, a honest *V* can be convinced of this fact by an untrusted *P*.

### 2.1.8 MarkPledge

The MarkPledge technique [7, 24] ensures the voter that his recorded choices are consistent with his vote intention, i.e., guarantees a cast-as-intended verification. To do so, every candidate present in the ballot has associated a single bit *b*. This bit is 1 if the candidate was selected by the voter or 0 otherwise. Each bit is encrypted by a sequence of *n* pairs of ciphertexts, named Ballot Mark Pairs (BMP). After the voter selects his candidate, the prover generates a commitment of n bits called ChosenString. For $b = 1$, if the bit *i* of the ChosenString is 0, the $\text{BMP}_i$ is the encryption of [0,0], otherwise the $\text{BMP}_i$ is the

encryption of [1,1]. For b=0, however, each BMP can be either the encryption of [0,1] or [1,0]. Both the ballot and the ChosenString are sent to the voter as a commitment. The voter will then send a challenge with length *n* that indicates which bit of each BMP is to be opened. The prover opens the chosen bits for every candidate, generating a string of length *n* for each candidate. The voter can then verify if the ChosenString matches the string that corresponds to his selected candidate, proving the validity of the encryption with a soundness of $1 - \frac{1}{2^n}$.

### 2.1.9 Code voting

Code voting, first implemented by David Chaum [25], is an approach used to try and ensure a voter cast-as-intended verification. It consists in secretly sending code cards to the voters before the election that map each candidate to a vote code. The voter uses the card to vote for his desired candidate, making the encryption himself and guaranteeing that the computer does not change his vote intentions, since it has no access to the remaining voting codes. The voter sends the vote code to the server, which is verified and then a confirmation is sent back, ensuring that the vote was cast-as-intended. This approach has some weaknesses:

1. Guaranteeing that the code cards are secretly generated and anonymously delivered to each voter;
2. The server does not guarantee that the vote code was mapped to the desired candidate, only confirming that the vote has reached an entity that knows the correct verification code.

### 2.1.10 Trustee

Trustees handle cryptographic keys in a voting system, and together they hold the power to decrypt the tally, or even individual votes. Therefore, an election should make use of multiple trustees with different allegiances to grant the voters' privacy, the election's integrity and to avoid any possible collusion between them.

### 2.1.11 Threshold Public Key Cryptosystem

A (*t*,*n*)-threshold public key cryptosystem [26] shares the secret key among a set of *n* trustees. To decrypt a message, it is required the collaboration of at least *t* trustees. Its decryption protocol must guarantee that the secret key cannot be reconstructed by any of the trustees.
Shamir's Secret Sharing Scheme [27] is a good example of this cryptosystem. A polynomial function of degree $t - 1$ is generated at random, whose free term is the secret *x*, that is, $f(0) = x$. Each trustee is assigned an index *n* and a share that corresponds to $f(n)$ is secretly sent to them. After this, if *t* trustees

submit their indexes along with their shares, the polynomial can be reconstructed, and therefore the secret $x$ can be obtained.

### 2.1.12 TLS/SSL

The Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) are both cryptographic protocols that are designed to keep internet connections secure over a computer network. Their goal is mainly to provide privacy and data integrity between two or more communicating peers. A basic TLS handshake between a client requesting data and the server responding to the requests is depicted in Figure 2.1. This procedure is required in order to exchange data by TLS, since the specifications required to do so are defined in it.

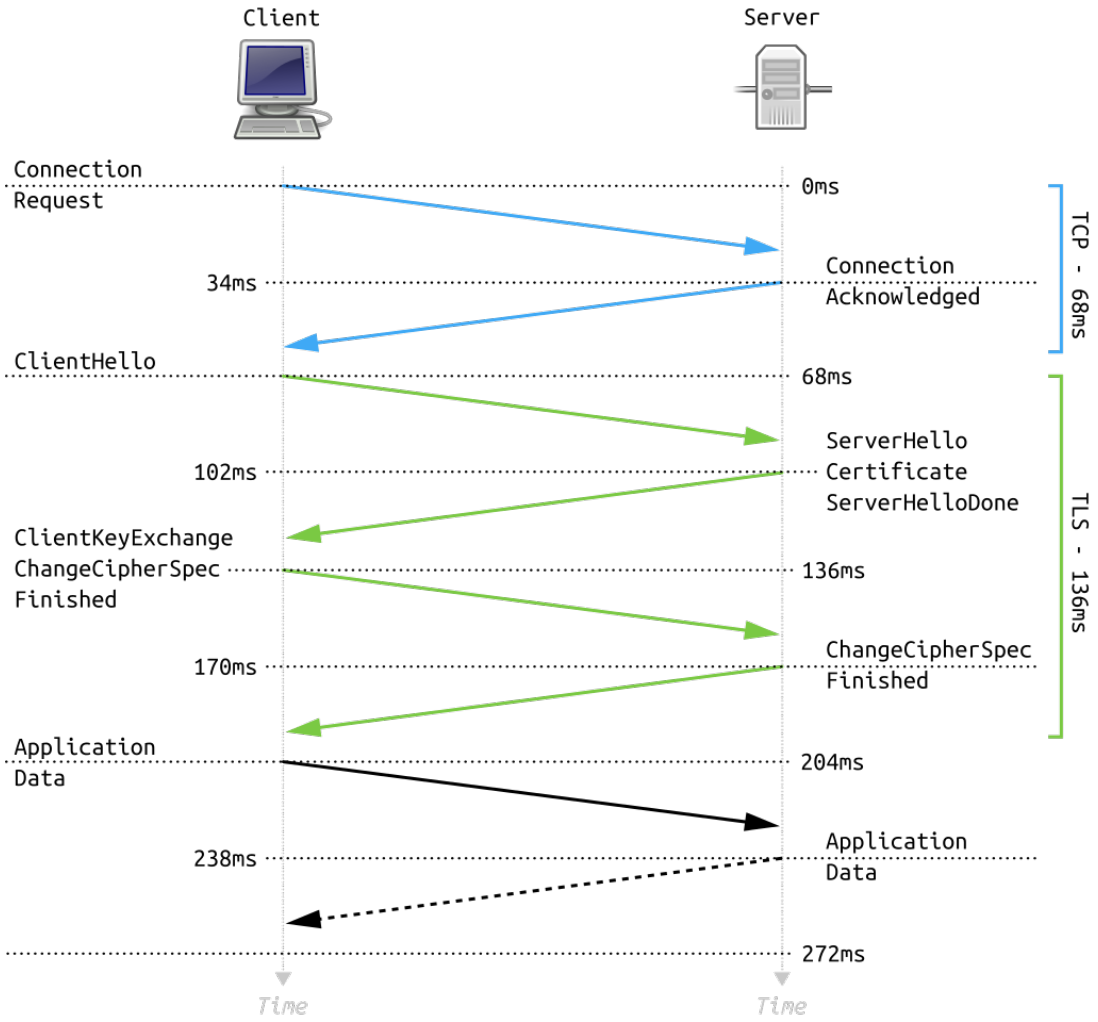



**Figure 2.1:** Example of a TLS handshake [1]

After establishing connection with the server, the client sends a **Client Hello** message to the server,

specifying the highest TLS/SSL protocol version it supports, a list of suggested cipher suites, suggested compression methods and a random number.
The server then responds with a **Server Hello** message, by sending the chosen protocol version, cipher suite, compression method and also a random number. It also sends the **certificate**, **Server Key Exchange** and **Server Hello Done** messages, the latter indicating that it is done with the handshake negotiation.
The client then verifies the server certificate and the cryptographic parameters sent by the server. If everything is in order, the client answers with a **Client Key Exchange** message that contains a **secret**, encrypted using the public key of the server certificate. Finally, both the client and the server compute a common secret called **master secret** using the random numbers and the previously encrypted **secret**.
The client sends a **Change Cipher Spec**, informing the server that every message sent to it from that moment on will be authenticated and encrypted. Then, an authenticated and encrypted **Finished** message (with the **master secret**) is sent, which will be decrypted and verified by the server.
The server also sends a **Change Cipher Spec**, informing the client that every message sent to it from that moment on will also be authenticated and encrypted. Then, it sends an authenticated and encrypted **Finished** message (with the **master secret**), that will be decrypted and verified by the client.
With the handshake complete, the client and the server can exchange authenticated and encrypted messages between them.

## 2.2 Types of E-voting Systems

### 2.2.1 Homomorphic Voting System

A homomorphic voting system [28–30] computes an election result without the need of decrypting any individual vote. To do so, it makes use of a homomorphic cryptosystem. This type of system guarantees that an algebraic operation performed on a plaintext corresponds to another algebraic operation performed on its correspondent ciphertext. For example, ElGamal is a cryptosystem that is homomorphic over multiplication while its exponential variant is additively homomorphic. The latter when used to generate the election tally can guarantee the privacy of the voters, since it allows to add every encrypted vote element-wise. However, privacy is only assured if no individual votes are decrypted during the tally generation process.

### 2.2.2 Mix-nets Voting Systems

A mix-nets voting system [31,32] anonymously mixes the encrypted votes in order to generate a shuffled election's result, so that the privacy of the voters is assured. For that it uses a mix-net, which is composed

by a chain of mix servers. Each mix server encrypts the votes and shuffles them, proceeding to pass them to the next server on the chain. The mix-net must be able to zero-knowledge prove that the election results are product of the successive re-encryption of the voter's votes, while maintaining the shuffle anonymous. Mix-nets also increase the flexibility of the voting platform, since it allows more complex types of elections, e.g., write-ins.

### 2.2.3 Blind Signatures Voting Systems

A blind signatures voting system [33, 34] authenticates the votes with a digital signature of an election authority. However, to maintain the voters' privacy, the election authority signs a blinded vote. The vote is then unblinded by the voter which is then encrypted and sent anonymously to the ballot box. To compute the results, every vote is decrypted and the validity of the votes can be verified, since they must all have the authority's signature. The main concern with this approach is that there is no counted-as-cast verifiability, making it impossible to verify the validity of the voters.

## 2.3 E-voting Systems

In this section the e-voting systems researched are presented, including a brief description about each one of them. Every e-voting system presented here complies with most of the main security properties of e-voting stated in Section 2.1.1.

### 2.3.1 VeryVote

The VeryVote system [35] is a mix-net code voting system that adapts MarkPledge's cryptographic technique in order to achieve a cast-as-intended verification.
The election server creates and sends to every voter a code sheet before the polls open. The election key is created by the trustees which is signed by the electoral commission and published in the bulletin board. On the election day, the voter types the code that corresponds to their favourite candidate.
It is then used the MarkPledge's encryption technique referred in Section 2.1.8 to create a receipt for the cast vote. The voter then checks the receipt to confirm that the vote confirmation code is associated with the selected candidate. This receipt is also signed by the electoral commission and published in the bulletin board to allow a claiming stage, during which the voters can check and revoke their votes.
Every validated vote goes through a mix-net protocol in order to be anonymized. After this process, the trustees decrypt and publish the votes in a shared and verifiable way which allows for other entities to verify the correctness of the vote decryption.

### 2.3.2 EVIV

The End-to-end Verifiable Internet Voting (EVIV) system [36], much like VeryVote, integrates the Mark-Pledge technique with a homomorphic code voting protocol. The main difference between both voting systems is that in EVIV each voter has a voter security token (VST) which can be for example a smart card, that contains a unique cryptographic key pair used to encrypt the vote.
The voter gets registered in the electoral roll by presenting himself to a local authority office, getting a VST. When every voter is registered, the election parameters, an electoral roll containing the list of voters and their public keys are published on the bulletin board. Then the trustees get an ElGamal shared threshold key distributed between them which is verified by the Electoral Commission.
The voter can now register for the election by connecting his VST to a computer and to the Election Registrar via a secure connection such as TLS/SSL. Then the VST receives the candidate list, the election public key, and creates a ballot encryption, signing it and sending it to the Election Registrar. If the ballot is correct, it's published in the Bulletin Board, that will contain all the valid ballots and be signed by the Election Registrar. Finally, the VST creates a code card that must be kept secret by the voter in order to maintain privacy.
To vote, the voter connects his VST to the Ballot Box and introduces the code associated to his intended candidate. The receipt is then presented to the voter which allows the voter to immediately prevent the vote to reach the Ballot Box if he notices something wrong. At the end of the voting phase, all the data received in the Ballot Box is signed by the Electoral Commission and then published in the Bulletin Board.
In the last phase, anyone is able to verify all of the election public data without compromising the voters privacy.

### 2.3.3 Helios

Helios [7–9] is a homomorphic voting system that allows any willing observer to audit the entire process of an election. It also uses SSL to transmit data between the voter's browser and its servers.
An election in Helios has only one administrator that creates and is responsible for it. Its ballot is composed by multiple questions, each one having multiple choices which must be setup by the administrator. The administrator is also responsible to setup the trustees for the election, and he has the option to add Helios as one of the trustees. The trustees generate a shared election key pair using ElGamal Encryption and publish the shared election public key to the server. The administrator can add, update and remove voters at will. Each voter is identified by a name and an e-mail address, to where the voter's credentials (username and a randomly generated password), the hash of the election and a link to the voting booth will be sent. After everything is set, the administrator can freeze the ballot and open the

election.

When the election begins, voters can enter the voting booth by accessing the link received by e-mail and begin the voting process. After the voter selects their desired choices for the ballot questions, their ballot is encrypted (using Exponential ElGamal Encryption) and a hash of the ciphertext (known as the ballot tracker) is displayed. The voter is now presented with a cast-as-intended verification that is based on the Benaloh's "cast-or-verify" method [20, 21]. If the voter chooses to audit the ballot, the ciphertext and the randomness used to encrypt the selected choices are revealed which allows for the voter to verify the correct encryption of the ballot. After auditing, the voter will have to confirm once again their desired answers and have them encrypted in a new ballot. The voter may audit the ballots successively until being satisfied with the encryption process. Alternatively to auditing, the voter can choose to seal the ballot discarding all randomness and plaintext information, leaving only the ciphertext. The voter is then prompted to authenticate using the credentials received by e-mail and if successful, the encrypted vote will be recorded in the server, which in turn acknowledges the vote reception by sending an e-mail to the voter's e-mail address with the ballot tracker.

A voter may vote multiple times, only the last cast ballot will be counted. The ballot trackers are then displayed in a bulletin board, next to the correspondent voter's name or an alias. Anyone can access this bulletin board and find the encrypted votes posted there.

When the administrator closes the election, the encrypted tally is computed by aggregating every encrypted vote, making use of the additive homomorphism property of the exponential ElGamal cryptosystem used to previously encrypt the individual ballots. To decrypt the tally, the trustees must submit their decryption factors which are computed using the election secret key previously created. Once every trustee has submitted the decryption factors, the tally can be decrypted and the administrator can release the results to the public.

After the results have been released, it is possible for any observer to verify the tally. The verification program downloads every needed parameter, verifies every proof and re-performs the tally based on the decryptions.

The main issue with Helios is coercion. A voter that shows their selected choices together with the ballot tracker to a third party cannot vote again without the third party knowledge. This happens because the voter's ballot tracker will be updated with a cast of a new ballot, and this update can be easily detected by the third party. Helios is also vulnerable to ballot stuffing attacks, since a dishonest bulletin board could add ballots to the tally without anyone noticing.

### 2.3.4 Belenios

Belenios [37, 38] is a homomorphic e-voting system that partly implements the Helios-C protocol [39] which makes use of voter signatures to avoid ballot stuffing attacks.

When setting up an election, the registrar (credential authority on the voting platform) generates and sends privately a signing key to each voter and their corresponding verification keys to the voting server. The server publishes the list of verification keys on the bulletin board and generates a password for each voter, sending it to them in private.
The election key is generated using a threshold public key cryptosystem using ElGamal encryption. Each trustee sends his public key to the server together with a proof of knowledge (POK) of the secret key. The public election key is then published on the public bulletin board by the voting server.
During the voting phase, the voters select their vote which in turn is encrypted and signed by their voting device. The resulting ballot is sent via an authenticated channel to the voting server using a login and password mechanism. The server performs several checks in order to verify the validity of the ballot and adds it to the bulletin board if everything is in order. If there is already a valid ballot submitted by the same voter it gets replaced by the last submitted ballot. Voters can then check if their last submitted ballot appears in the bulletin board.
When the election is closed, the trustees contribute to the decryption of the list of the accepted ballots in the bulletin board by providing their decryption factors together with a POK of correct decryption.
Like Helios, Belenios also does not provide any coercion resistance since voters may provide the randomness used to produce their ballot or they may simply sell their voting material.

### 2.3.5 EPFL E-voting System

The EPFL E-voting System [16, 17] is a mix-net laid upon a decentralized and distributed architecture that makes use of blockchain technology and verifiable cryptographic shuffles of ElGamal ciphertext pairs.
The system is handled by the cothority [40] (short for collective authority) that provides a platform to handle arbitrary blocks of data through the use of an alternative implementation of blockchain, named skipchain.
Skipchain [41] is used to store all the election related data. The master skipchain handles configuration data that is common to a set of elections, and its genesis block stores a list of servers that are meant to handle the protocols and skipchains, a list of election administrators that have the privilege to create new elections, a public key of a potential front-end application and its own skipchain identifier.
When a new election is created an election skipchain is also generated. This skipchain contains all the data related to its election including the actual election data and the ballots cast by the voters. Each election skipchain has an identifier that is appended to the master skipchain in a separated block called ”link” and its genesis block contains the server list, its own skipchain identifier and more importantly, the election’s public key. The correspondent secret key is shared among the different servers.
When a voter casts a ballot to the cothority, it is previously encrypted using the election public key.

The ballot only contains its correspondent user's identifier and the encrypted vote in the form of an ElGamal ciphertext. When the administrator closes the election the submitted ballots are re-encrypted and permuted several times by the various servers. Finally, each server is prompted to decrypt the shuffled ballots with their own secrets and the final result is appended in the concluding block of the election skipchain. At no point during the encryption protocol is it necessary for one server to accumulate the shared secrets, respecting the distributed key generation protocol.
A direct consequence of skipchain is that the final shuffled and decrypted ballots are accumulated in a single block, whereas every original encrypted cast ones are stored in different blocks. This is due to the fact that when a new block is added to the chain, it can no longer be changed. Therefore, an election skipchain has always the same structure, starting by the genesis block, followed by the encrypted ballots and ending with the accumulated shuffled and decrypted ballots in the concluding block.

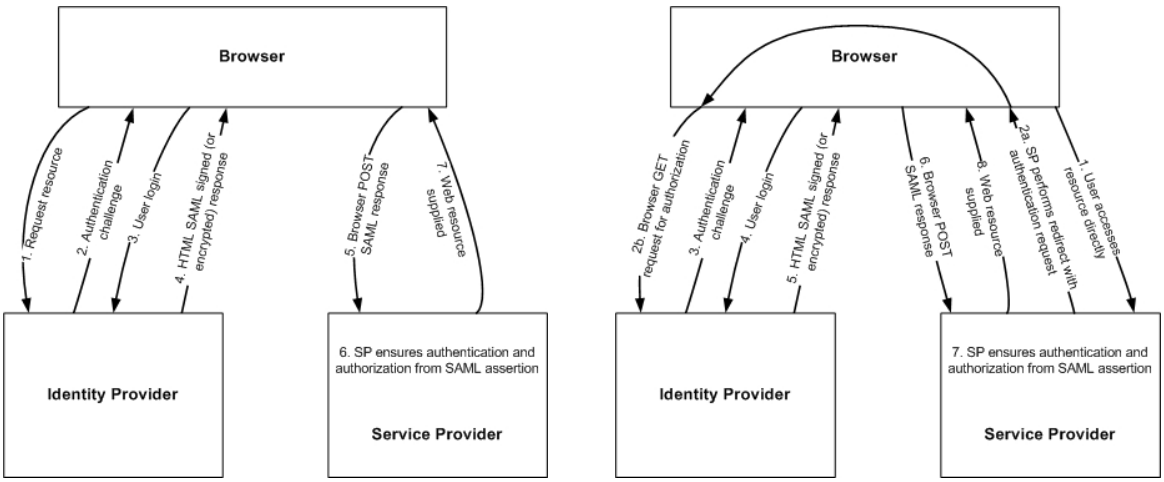
## 2.4 Authentication and Authorization

### 2.4.1 SAML

SAML [2] is used to exchange authentication and authorization between parties. The main advantage of using SAML is that a user can log into many different applications and web-based portals within an organization using a central authentication source with one set of credentials (Single Sign-On - SSO). There are three parties involved in a SAML transaction:

1. Asserting party (Identity Provider - IdP): system that provides the user information;
2. Relying party (Service Provider - SP): system that trusts the asserting party's information and uses the data to provide an application to the user;
3. Subject: user that is involved in the transaction.

The transaction from the IdP to the SP is called a SAML assertion which is defined by an Extensible Markup Language (XML) schema containing header information, the subject and statements about the subject in the form of attributes and conditions. These assertions are sent between the IdP and the SP using request and response protocols. SAML bindings then map these SAML protocols onto standard lower level network communication protocols used to transport the assertion. The highest SAML component level is profiles which dictates how the assertion, protocol and bindings will work together to provide SSO.
The most popular business use for SAML federation is the web browser SSO profile, used in conjunction with the authentication request protocol and the HTTP POST binding. This profile may either be initiated by the IdP or by the SP, both cases depicted in Figures 2.2(a) and 2.2(b) [2], respectively.




**(a)** IdP Initiated SAML Assertion Flowchart **(b)** SP Initiated SAML Assertion Flowchart

**Figure 2.2:** Web browser SSO profile initiation [2]

In both cases, the SP receives a SAML assertion from the IdP, containing the authentication, authorization and attribution of the user. The SP must then ensure the authentication and authorization of the user and will use the attributes sent in the assertion to make access-control decisions.
To set up a federation, it is needed to configure a local entity, a partner entity and an association between the two that forms the federation. This can be done by using a federation manager product, for example, Shibboleth [42] - an open source project that uses the OpenSAML toolkit which supports developers working with SAML.

### 2.4.2 Distinguished Names

Lightweight Directory Access Protocol (LDAP) [43] is mainly used as a central hub for authentication and authorization. Users attributes can be stored in LDAP and these determine what that user is allowed to access and do. Each user is referenced by its Distinguished Name (DN). A DN is a sequence of relative distinguished names (RDN) separated by commas which are attributes with associated values.

# 3

# Methodology

## Contents

The chapter begins by presenting the chosen solution and the reasoning behind its choice (Section 3.1). It's a widely used electronic voting solution with a few years which has been adapted to the UL's system and improved in terms of usability and security. These modifications and improvements are stated in Section 3.2 including the connection with the UL's Identity Management System. Finally, the chapter ends by detailing a field test (Section 3.3) that serves the purpose of identifying any problems and possible improvements to the implemented voting solution.

## 3.1 Helios

The solution that was chosen to be implemented from the systems detailed in Section 2.3 is the Helios Voting System [7–9]. From all the systems analyzed, Helios is the one that has been mostly used to run real-world elections, including the election of the president of the University of Louvain-La-Neuve and the elections of 2010, 2011, and 2012 for new board directors of the International Association for Cryptographic Research (IACR) [44]. Helios has a robust front-end and doesn't require VST's or makes use of code voting systems which simplifies the voting process. Therefore, even being subject to coercion and ballot stuffing attacks, Helios is a great choice for low-risk, small scale environments such as university student governments.

Helios is built in Django [45], a python web framework, uses PostgreSQL [46] as its relational database system and requires the configuration of a web server. Apache [47] was chosen to be configured as such web server and in order to allow an authentication via IdP, shibboleth [42] was also configured. This solution, with some modifications, can fulfill every requirement stated in Section 1.2. It also satisfies most of the criteria listed in Section 2.1.1, with the exception of:

- *Non-Coercibility* - Voters are able to prove how they voted if they record or show their voting process to a third party;
- *Flexibility* - The system does not accept ballot questions with different formats (such as write-in candidates, different languages) and does not have implemented any accessibility standards for individuals with disabilities.

The Helios version used in this project was installed in a machine running a version of CentOS 7.7.1908 [48]. It was obtained from the Instituto Federal de Santa Catarina's GitHub repository [49] in January of 2020 and its map is depicted in Figure 3.1. This map is organized by blocks each representing a major page of the platform. Each block also has a colour associated which represents the permission required to access that page or to perform that action in the platform. White blocks are accessible by any user including non-authenticated ones. Orange blocks are accessible only by registered voters of a

given election. Purple blocks can only be accessed by platform administrators. These blocks correspond to the administration tool of the platform which provides an interface where trusted users can manage content on the platform site. Only a handful of users should have access to this tool and all their actions performed in it are recorded in logs. However, it should be noticed that it is possible for a user to delete cast votes, voters, trustees, users or even elections in this tool which can seriously compromise the platform's security if not handled correctly. It's in this tool that any previously authenticated user can be given election administrator privileges which allows them to create and administer elections. Finally, the election administrators are the ones with access to the blue blocks.

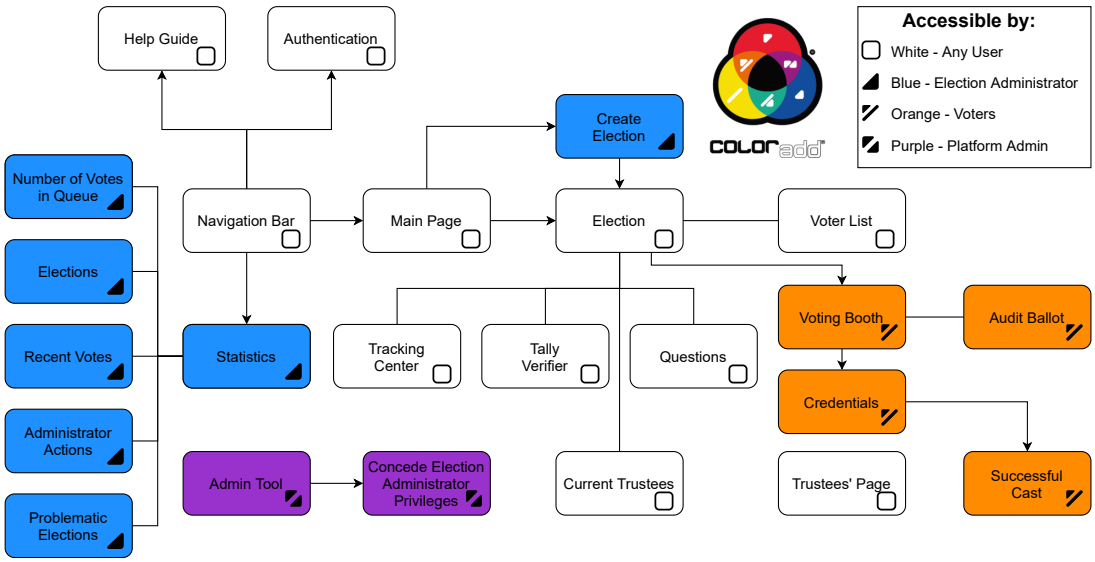



**Figure 3.1:** Helios Platform Original Map [3]

The user accesses the platform and is presented with its main page. Here, he has access to the navigation bar, a list of featured elections and, if the user is an election administrator, he will also have the option to create a new election. The navigation bar is always accessible (except in the voting booth) and allows the user to authenticate, to visit the help guide and to return to the main page of the platform. Also, an election administrator has access to a menu called statistics in it, that contains diverse information about every election. To access an election, a user chooses an election from the list of featured elections in the main page which will bring him to the selected election's page. In there, the user has access to the election's voter list, the questions present in its ballot, a list of its current trustees, the tracking center and, if the tally is computed, the tally verifier. The election's page also gives access to the voting booth where the voting process will take place. The voting booth has an auditing tool, that allows the voters to verify if their choices were correctly encrypted into the ballot if they wish to. It also gives the option for the voter to post their audited ballot in the previously mentioned tracking center. The tracking center

contains every audited ballot that was posted in it, allowing every voter to audit them. After encrypting and sealing the ballot, the user is prompted to insert credentials that were sent to him by the election administrator via e-mail and only after submitting them, the ballot is successfully cast. After casting the ballot, an e-mail confirming the cast of the ballot is sent to the voter's e-mail address. At last there is the trustees' page which is only accessible via Uniform Resource Locator (URL). Each trustee has its own trustee page and accesses it by using an URL sent by the election's administrator by e-mail.

## 3.2 Implemented Changes

This subsection states the changes implemented on the original state of the voting platform that aim to improve its security and usability and also to fulfill the requirements stated in Section 1.2. Some of the mitigated issues were already known and are mentioned in [50–52], namely general usability problems, inability to cast a ballot, inability to verify the cast ballot and security concerns regarding the authentication using the credentials sent via e-mail. To ease the understanding of the different issues and the correspondent implemented solutions, the different blocks of Figure 3.1 are explained in detail individually in Section 3.2.2.

### 3.2.1 Initial Configurations

Helios strongly relies on sending e-mails, either to contact voters and trustees or simply to inform the administrator of election related issues. Therefore, an e-mail address was created for the platform which uses an already configured internal Simple Mail Transfer Protocol (SMTP) server. This e-mail address is responsible for sending every e-mail from the platform.

To configure the intended authentication system, it is required to set up the connection with the UL's Identity Management System. To do so, it was firstly needed to install and configure a Shibboleth SP. After installing the required certificates and setting up the communication with the UL's IdP, the system was then ready to receive and process user attributes.

It was originally possible to associate a user with one and only one faculty. As a consequence of this, by default, every election administered by this user would also be associated with that faculty. This behaviour was not intended since the election may not be necessarily associated with the administrator's faculty and users can in practice pertain to multiple faculties. To circumvent this problem, users cannot be associated with faculties anymore. Instead, every election has now a faculty associated that is manually chosen by its administrator.

The platform needs election administrators that can create elections, but not everyone can be an administrator since there would be the possibility of a malicious user to create elections that are identical to already existing elections (spoofing attack), or to create a massive number of fake elections. The solu-

tion implemented is to create a new role: delegates. Delegates are users considered to be responsible and as such bear the duty of assigning and removing election administrators. This brings another benefit as well since it allows for each faculty to have their own delegates. Also, every election administrator has now an expiration date associated defined initially by the delegate that will revoke his administrator privileges automatically.
The colour scheme of the platform was changed so it's more similar to the UL's platform, the UL's favicon was added and proper titles were added to every page.

### 3.2.2 Modifications by Block

#### 3.2.2.1 Admin Tool

There is no longer the option to set users as election administrators in the admin tool. Instead, it is possible to set users as delegates which can in turn assign and remove election administrators, as stated before.

#### 3.2.2.2 Main Page

The list of featured elections was replaced by a list of faculties. Choosing a faculty from the list will display every election that is associated with it.
Delegates can now concede and revoke election administrator privileges to users and this new functionality is accessible through the main page of the platform.
The main page of the platform shows a list of elections administered by the authenticated user. This list only contained elections that had their ballots already frozen. This was changed and the list now contains every election administered by the authenticated user, with its ballot frozen or not.
A list containing the elections of which the authenticated user is a trustee was also added, as well as the user's attributes.

#### 3.2.2.3 Navigation Bar

The UL's logo was added to the platform's navigation bar which redirects to the UL's website.

#### 3.2.2.4 Authentication

To guarantee the uniqueness of each voter it is required a unique identifier. The attribute used initially to guarantee this was "eduPersonPrincipalName" (EPPN) which was composed by a RDN named "commonName" (CN) and an e-mail domain.
However, the EPPN could not be used as the unique identifier since it was not immutable. It was found

that a user has the option to change the e-mail domain present in the EPPN which means that a single user could authenticate with different unique identifiers and trick the SP into thinking that these were different users. A direct consequence of this would be the cast of multiple votes by a single user in a single election. This, obviously, cannot be allowed to happen and a different attribute had to be used as the unique identifier. The chosen replacement is the DN which in this case is formed by 3 RDN's: CN, ”organizationalUnitName” (OU) and ”organizationName” (O). However, OU is also not immutable, having multiple possible values such as ”Users”, ”Temporal”, ”External” and ”Pending”. Since only the users with OU = ”Users” are considered active users, a restriction was made on the SP, enforcing that only users with OU = ”Users” are authorized to access the platform.

#### 3.2.2.5 Help Guide

A new help guide was implemented in the platform to ease the use of the platform by users, trustees and administrators. Every functionality of the platform is depicted there along with detailed steps on how to use them.

#### 3.2.2.6 Statistics

This tool includes the number of votes in queue, a list of every election, a list containing every vote cast during the last 24 hours, a log of election administrators actions and a list of problematic elections.
Since there are no statistics in this tool, it was renamed to ”Administration Panel” to better reflect it's purpose.
The problematic elections were removed since it contained a list of elections with the ballots unfrozen for more than 24 hours. This does not represent a problem, because unfrozen elections are only accessible by the administrator, its trustees and directly by URL, not being visible on the main page of the platform. The tool can now be accessed by both election administrators and delegates.

#### 3.2.2.7 Credentials

The purpose of the previously mentioned credentials that are sent via e-mail by the election administrator is to guarantee that users are registered voters. This is successfully achieved since only users whom have received these e-mails would have the correct credentials, guaranteeing that the users were indeed registered voters. However, with the user authentication correctly configured, a user can only access the voting booth and cast a ballot if he's correctly authenticated in the platform and has been registered in the election by its administrator which makes the use of credentials redundant and therefore, they were removed. The removal of the credentials directly solves the previously mentioned security problem detected in [51].

#### 3.2.2.8 Create Election

When creating an election, the administrator will be prompted to choose a faculty to be associated with the election, guaranteeing that one election is always associated to one faculty.

#### 3.2.2.9 Election

A user is now required to authenticate before accessing an election.
It was created an administrator panel in the election's main page which is only visible to the administrator. This panel is separated from all the other information, includes some admin actions (copy, edit and archive election) and shows the ”Next Step” of the administrator. The ”Next Step” information available to the administrator is now better explained, so that he always knows what to do next.
An administrator can now delete an election in the administration panel if the ballot box isn't frozen yet. Also the options for the administrator to archive and copy the election are now only available after the release of the results.
When an administrator finished setting up an election and every trustee has uploaded their public key, the ballot can be frozen. If the election only has 1 trustee in it, a warning will now be shown to the administrator stating that only 1 trustee is selected and that a minimum of 2 trustees is highly advised to guarantee the integrity of the election. Also, when the administrator is either freezing the ballot or locking the encrypted tally, a confirmation dialog will now be prompted, stating that doing so will be irreversible.
The administrator is now only able to e-mail a trustee if he either hasn't submitted his public key, or if, with the encrypted tally locked, he hasn't uploaded his decryption factors.
When every trustee has already sent their decryption factors, it no longer states that the administrator is waiting for trustees decryptions and instead states correctly that the administrator can now launch the results.
If for some reason the administrator decides to end the election earlier than stipulated, this action will now be stated in the election's main page, specifying the time at which the administrator ended the election.
The preview/review voting booth and the election tally verifier links were moved from the bottom of the page and a reminder to verify the election tally was added.
It was also possible to review the voting booth before the administrator locked the encrypted tally if the election original ending time has finished. Since the administrator can extend the election's ending date, reviewing the voting booth is now only allowed after the encrypted tally is locked.
The platform did not always display correctly if a user was registered as a voter or not which was fixed. Also, if a question allows multiple answers, the results would also announce multiple winners which was changed to only announce one winner per question.

#### 3.2.2.10 Voter List

The administrator had 2 options to register voters in his election. The first was to allow every authenticated user to vote which wasn't very useful to the platform, and therefore was removed. The other option was to upload a list of voters, composed by the users' names, unique identifiers and e-mail addresses which is useful but still not quite enough. A new option was implemented and it works in conjunction with the voters' list, providing more versatility to the administrators. An administrator can now also register a voter based on his attributes by choosing one or more categories. These categories are composed by 8 fields, 3 of them being static and the other 5 being dynamic. The static fields are ”students”, ”professors” and ”employees” and the administrator must always choose at least one of these when adding a new category. The dynamic fields are optional and named ”faculty”, ”establishment”, ”management”, ”area” and ”nucleus”. The dynamic fields have a hierarchic relation, where ”nucleus” belongs to ”area” which by its turn belongs to ”management” and so on and so forth. The platform is initiated with no default dynamic fields since it's a big logistic operation to gather them all and it would also require manual updates. These fields are obtained during users authentication where the user's attributes are verified and if the combination of these 5 fields is unfamiliar to the server, they are saved and made available as categories. Every voter can still receive e-mails from the platform since the voters registered by categories have their e-mail addresses present in their attributes. The templates of these e-mails have been changed: they don't contain user credentials anymore, they remind the voter that they can vote as many times as they want to and they also remind the voter to verify the election tally if it has been computed.
The voter list is now ordered by voter name and not by voter Universally Unique Identifier (UUID) which makes navigation in it easier.
It was added a button that allows the administrator to remove every voter manually uploaded instead of only being able to remove them one by one.
A voter and consequently his cast votes could also be removed from an election with the ballot already frozen. This is not possible anymore, the administrator can only remove voters before freezing the ballot.

#### 3.2.2.11 Questions

The default number of answers for a question was changed from 5 to 2 and the administrator cannot choose anymore a minimum number of selected answers that is higher than the maximum.

#### 3.2.2.12 Current Trustees

The trustees are added by the administrator that needs to submit the trustee's name along with his e-mail address. Then, the administrator sends an e-mail to the trustee that contains the URL to his trustee page, so he can submit his public key and his decryption factors.

A trustee does not need to be authenticated to access his page and theoretically, anyone with knowledge of the URL can impersonate the trustee and make the submissions as him. This was fixed by associating every trustee's URL to a previously authenticated user, therefore making a trustee's URL inaccessible if the user isn't authenticated as the associated trustee. A direct consequence of this change is that the administrator is now restricted to choose the trustees from a list of previously authenticated users. Also, the trustee's e-mail address is now automatically fetched from the database unless the administrator manually submits it.
"Trustees' Page" is now accessible to authenticated trustees via "Current Trustees" instead of only being accessible by URL.
The original e-mail template that was sent to the trustees by the administrator has been updated. Since the administrator can now only e-mail the trustees if they have not yet uploaded their public key or their decryption factors, this template has been divided into 2, one for each of the situations stated before. Each of these templates contains a brief guide in order to help the trustees to perform their uploads. The URL that was used for the trustees to access their homepage was also removed from the templates.
The trustees list page has now more information directed to the administrator in it.

#### 3.2.2.13 Trustees' Page

A possible exploit was found concerning the submission of the trustee's public key. If the trustee saved the URL while submitting the public key, he could re-submit another public key with the ballot already frozen. This compromises the tally's integrity since the election's public key would not correspond to the key formed by the trustees anymore. This issue was mitigated by blocking public key re-submissions after freezing the ballot.
Helios allows the trustees to verify the validity of their secret key anytime by inserting both the public and secret keys. However, this would only check if the hash created with the submitted keys corresponded to the hash stored in the server. A new verification step was added that also checks if the submitted secret key generates the submitted 'y' parameter of the public key.
The page for the trustees to upload their keys is now more organized. Also when a trustee is generating his decryption factors using his key pair, if the key pair appears to be invalid, a correct format of the key pair is shown to the trustee. This was implemented because the trustee may have his key pair saved correctly, but in a wrong format that cannot be read correctly by the platform.

#### 3.2.2.14 Voting Booth

There was too much information exposed in the voting booth. This excess of information can confuse voters, thus a big part of it was moved to the help guide. Only indispensable information that allows a user to vote without using the guide was kept in it.

#### 3.2.2.15 Audit Ballot

Excess information was also moved to the help guide only leaving indispensable information that allows a user to audit the ballot without using the guide.
Helios uses a tool named ”Single-Ballot Verifier” that allows the voter to audit his encrypted ballot being responsible for cast as intended verification. This tool, besides verifying if the ballot is correctly encrypted and showing the encrypted answers also verifies if the ballot corresponds to the election. However, only the election fingerprint (hash) of both ballot and election was verified. A new verification step was added that also checks if the UUID present in the ballot corresponds to the election's UUID.
A button was added to the Single-Ballot Verifier that allows to restart another ballot verification and this page's buttons were reorganized to be more similar to the voting booth.

#### 3.2.2.16 Confirm Cast

After sealing the ballot, the user was prompted to insert the previously mentioned credentials that were sent to him via e-mail by the election administrator. Since these credentials are no longer used, the user is instead prompted to confirm the cast of his sealed ballot.

#### 3.2.2.17 Tally Verifier

The tally verifier does not allow multiple verifications to run in the same page at the same time anymore since the button that starts the verification is immediately replaced by an unclickable button that informs the status of the verification.
The result of the verification is now more organized and less crowded.
Some utility was added to the page, such as buttons to go to the bottom and top of the page, a ballot tracker finder and also a button to restart another tally verification.
Some URL examples were added to the tally verifier and a warning was also added that informs when the URL is invalid.
A reminder was added after the verification finishes for the voter to verify the presence of his ballot tracker in the results.

### 3.2.3 Solution Map

After the implementation of every solution, the platform map is slightly changed being depicted in Figure 3.2.

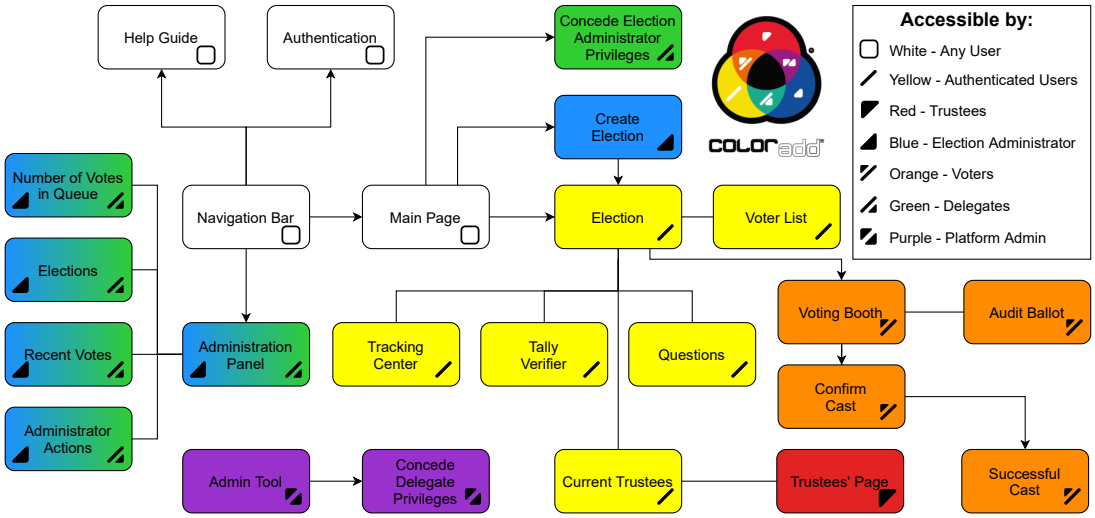



**Figure 3.2:** Helios Platform Solution Map [3]

## 3.3 Field Test

With the new solution implemented, it was time to conduct a field test with a significant number of users. The main objective of this field test was to identify any problems, oversights and ways of improving the platform. This was achieved by designing a feign election, gathering people to vote on it and to provide feedback detailing their voting experience.

The election took the form of an innocuous poll, asking the voters which was their preferred e-learning solution (from a short range of choices). Voters could be any UL's member who belonged to any of the following groups: students, professors and employees from IST; students, professors and employees from Faculdade de Ciências (FC); employees from the UL's rectory. Finally, 2 trustees were also chosen to follow and verify the election. The election was open during one day to vote from 9 am to 11 pm and a form was made available to obtain feedback from the participants that wished to share their voting experience. This form was developed with the intention of verifying whether the issues stated in [50–52] have been correctly mitigated and can be consulted in Appendix A

# 4

# Detected Issues

## Contents

Every issue detected with the field test is depicted in this chapter. The field test counted with 97 participants, 51 of which were students, 25 professors and 21 employees. Of the 97, 36 filled the form in Appendix A, providing information about any issues that were detected. These issues are detailed in Section 4.1 while any other issues detected and communicated by the participants of the field test are in Section 4.2.

## 4.1 Issues Encountered by Voters

Out of the 36 participants that filled the form there were 5 students, 17 professors and 14 employees and every issue detected by them are depicted in Table 4.1. Every issue has a code assigned to it, as well as the percentage of participants that encountered it and they are organized in 3 levels: *General*, *Voting* and *Auditing*. It should also be noted that every participant that filled the form was able to cast a vote.

**Table 4.1:** Issues Encountered by Voters

| **Percentage of Participants** | **Issue Code** | **Issue Encountered** |
|---|---|---|
| *General* | | |
| 19% | **[G1]** | Did not find the platform easy and/or intuitive to use |
| 3% | **[G2]** | Had their attributes incorrect |
| 25% | **[G3]** | Couldn't understand the meaning or usefulness of some technical terms |
| 47% | **[G4]** | Did not find the help e-mail address in the main page of the election |
| 36% | **[G5]** | Did not know that he could cast a vote multiple times |
| *Voting Booth* | | |
| 19% | **[V1]** | Felt excess or lack of information during the voting process |
| 31% | **[V2]** | Felt that the voting process is too long |
| 6% | **[V3]** | Encountered problems while casting the voting ballot |
| 25% | **[V4]** | Did not receive the e-mail confirming the cast ballot |
| 14% | **[V5]** | Did not save the ballot tracker |
| 6% | **[V6]** | Had concerns regarding their voting privacy |
| *Audit Ballot* | | |
| 58% | **[A1]** | Did not understand that an audited ballot isn't accounted for the tally |
| 50% | **[A2]** | Did not audit a ballot |
| 28% (of auditors) | **[A3]** | Felt excess or lack of information during the audit process |
| 39% (of auditors) | **[A4]** | Didn't understand the purpose of the tracking center |

### 4.1.1 General

**[G1]** *Did not find the platform easy and/or intuitive to use* - The main cause of this issue is that the platform's interface is only in English and the participants were mainly Portuguese. Even though the participants could understand English some technical and unfamiliar terms may difficult the use of the platform. There is also no visual indication that a user is authenticated and after the authentication some users couldn't tell what to do next. Finally, it is not clear that the elections are associated to one faculty - some users thought that by performing the authentication via their faculty, that the election would also be associated with it.
**[G2]** *Had their attributes incorrect* - There are users with wrong attributes which can inhibit a user to vote in elections configured with categories.
**[G3]** *Couldn't understand the meaning or usefulness of some technical terms* - Even though all the participants could cast a vote, some of them had difficult understanding some more technical terms such as *Audit*, *Tracking center* and *Ballot tracker*.
**[G4]** *Did not find the help e-mail address in the main page of the election* - The help e-mail address can be found in the bottom of the election page and most users probably don't scroll through the whole page to be able to find it.
**[G5]** *Did not know that he could cast a vote multiple times* - This feature of the platform is mentioned after casting a ballot, in the help guide and in the e-mail templates. However, this information can easily be overlooked since most users will not read all the information shown to them.

### 4.1.2 Voting Booth

**[V1]** *Felt excess or lack of information during the voting process* - Some users felt overwhelmed by the information displayed in the screen while others didn't find the information that they needed.
**[V2]** *Felt that the voting process is too long* - Too many steps, verifications and ”clicks” are needed to cast a vote. There are also too many successive reminders to save the ballot tracker. Also the buttons to go to the next step are often in different locations of the page.
**[V3]** *Encountered problems while casting the voting ballot* - The button to cast the ballot is too small in comparison to the others. Also, the last page to confirm the submission of the cast ballot creates some confusion.
**[V4]** *Did not receive the e-mail confirming the cast ballot* - The e-mail address created to send e-mails in bulk was not authorized to do so being limited to send 100 e-mails per day. Since every ballot cast would result in sending one e-mail and some voters has cast multiple ballots, the e-mail address was blocked from sending more e-mails which resulted in the latest participants not receiving the e-mails confirming the cast ballot.

**[V5]** *Did not save the ballot tracker* - As previously mentioned in **[G3]**, many voters didn't know what to do with it. Others were just lazy or believed that the ballot tracker would be available to them if they needed to consult it.
**[V6]** *Had concerns regarding their voting privacy* - Some voters felt that it wasn't clear enough that their vote was secret and that the platform didn't transmit enough robustness and confidentiality.

### 4.1.3 Audit Ballot

**[A1]** *Did not understand that an audited ballot isn't accounted for the tally* - Even though that every of the participants was able to cast a vote many didn't understand the difference between auditing and casting a ballot.
**[A2]** *Did not audit a ballot* - Half of the users did not audit a ballot. Most of them simply weren't interested in doing so but some tried and gave up halfway through the process.
**[A3]** *Felt excess or lack of information during the audit process* - The auditing tool is not intuitive enough for being used by the average voter since it involves copying the opened ballot and pasting it in the verifier.
**[A4]** *Didn't understand the purpose of the tracking center* - The tracking center generated some confusion since some voters thought that their cast ballot would be registered in it.

## 4.2 Other Issues

Some issues were not possible to detect directly with the questions of the form being identified by the trustees, participants and the election administrator. This section describes these issues and when applicable, the causes of them.

**[O1]** *Redirect* - After authenticating via the navigation bar the user is not being correctly redirected always returning to the main page instead.
**[O2]** *Review the voting booth* - After the election's administrator closes the election and no more ballots can be deposited, it is possible for the participants to review the voting booth. This review is identical to the actual voting with the exception of no ballot being cast. However, a user who has not participated in the election (typically a late voter) and reviews the voting booth is still added to the voter list. Even though that this user cannot cast any ballot, any other participants may notice the increase of the number of voters after the closure of the election which in turn may lead to an incorrect assumption that people are still voting and breaking the confidence in the system.
**[O3]** *Aliases* - In the case of elections using aliases, only the administrator can see the names of the voters. This means that in practice, a malevolent administrator manually adding participants to the elec-

tion that should not be able to cast a vote would go unnoticed by other participants and even trustees.

**[O4]** *URL manipulation* - By manipulating either the URL's of the pages that contain the voter list or the current trustees, it is possible for any participant to see the names of the election voters (even with aliases enabled) and the trustees' e-mail addresses, respectively. Both of these URL's are necessary for the tally verifier and therefore cannot be deleted.

**[O5]** *Secret key submission* - Trustees are able to both generate the key pair on the browser or to submit a previously generated key pair. In either way, the voting system uses the trustee's submitted secret key to generate a POK with JavaScript that allows the system to verify if the trustee knows his own secret key without verifying the key directly. After the POK is generated, the trustee's secret key is discarded. This event has a direct impact on the trustee's confidence in the platform since they are obligated to submit their private key without knowing what is being done behind the scenes with it.

**[O6]** *Changing the voting choice* - Some voters had difficulty or were unable to change their voting choice in the voting booth. Since the election's question required exactly 1 choice, voters assumed that to change their choice it was enough to just select any other choice. However, the input type used for the ballot choices are check boxes that are designed to block when the predefined maximum number of choices is selected, in this case, blocking when a voter selects 1 choice and consequently not allowing voters to directly select another choice. To change the voting choice, voters had to deselect their original choice, and only then could they select a new choice which confused the voters.

**[O7]** *Auditing and Verification* - Users have both the single-ballot and tally verifiers to validate their encrypted choices and the integrity of the election, respectively. Although these tools are trustworthy they are still integrated in the voting system which means that voters may be reticent in trusting them.

**[O8]** *Changing the DN* - It is still possible for a user to change his unique identifier. This occurs when users activate their "ULisboa User Account" since doing so changes their DN and therefore their unique ID is altered. Even though each user can only perform this once (when activating his "ULisboa User Account"), it still is a very concerning vulnerability.

# 5

# Issue Mitigation and Improvements

## Contents

This chapter is dedicated to find and implement solutions for the problems stated in Chapter 4. Section 5.1 will use the map depicted in Figure 3.2 to present the implemented solutions and some improvements blockwise in order to ease their comprehension. All the other issues that are currently known but were not resolved are stated in Section 5.2. The chapter ends by detailing in Section 5.3 a real world election that took place using the implemented final solution.

## 5.1 Issue Mitigation

### 5.1.1 General

**[G1]** - The platform's whole interface is now also available in Portuguese and the elections are no longer associated to a faculty. The layout and usability was also reviewed with special attention to remove complex sentences with more technical terms, explaining it further or moving it to the help guide.
**[V4]** - The e-mail address is now, theoretically properly configured, being allowed to send 1000 e-mails per hour. The templates of the e-mails have more useful information and are now automatically signed by the election's administrator.

### 5.1.2 By Blocks

#### 5.1.2.1 Main Page

**[G1]** - Since the faculties were removed from the platform, the main page now contains instead a list of elections that an authenticated user may participate. Also a list that contains every election (with its ballot frozen and not archived) is now accessible in the main page. Finally, the main page now contains a text explaining the platform's objectives and to who it's destined.

#### 5.1.2.2 Entities List

This new block contains a list of entities that should be contacted by users who want to create and administer an election.

#### 5.1.2.3 Navigation Bar

**[G1]** - The navigation bar now states if a user is authenticated by showing his name. It also contains a button to change the platform's language between Portuguese and English.

#### 5.1.2.4 Authentication

**[O1]** - After authenticating, the user is now correctly redirected to the page that he was before.

#### 5.1.2.5 Help Guide

**[G3, V6]** - The help guide is now more detailed, including explanations about more technical terms and a section that details how the voting system works and guarantees that the deposited votes are private.

#### 5.1.2.6 Create Election

When creating an election, the administrator's e-mail address is now automatically assigned as the election's help e-mail address since it is the only way that voters have to contact the administrator. The elections are now private and have aliases enabled by default to comply with General Data Protection Regulation (GDPR). The administrator can still change or remove the help e-mail address and can still create non-private elections or disable the aliases.

#### 5.1.2.7 Election

**[G1]** - The election's main page has been streamlined with the objective of simplifying the voting process, increasing the size of the button that redirects the voter to the voting booth and diminishing the intensity of less crucial information.
**[G4]** - The name of the election's administrator and the help e-mail address were moved to the top of the election's page so that voters can find it more easily.
**[G5]** - A reminder stating that voters can cast a vote as many times as they want is now displayed in the election's page as long as the voting booth is open which means that voters can now read it even before casting any ballot.
**[O2]** - The option to review the voting booth has been removed. It allowed voters to review the questions, possible choices and to audit the newly encrypted ballots. Since there is a page containing the questions and their possible choices, the utility of the revision is limited to auditing ballots and therefore, it has been removed from the platform.

#### 5.1.2.8 Tracking Center

**[A4]** - The tracking center does not bring much utility since its main purpose is to allow participants to audit ballots posted there by other voters but without giving any way to know what was the original vote intention of the voter, and therefore not allowing a valid audit. This resulted in the removal of the tracking center.

#### 5.1.2.9 Voter List

**[O3]** - The voter list now contains a field stating if the voter was manually added by the election's administrator or if he was able to vote using his attributes. Even though this does not directly mitigate the

exploit, participants are now able to detect it and inquire the administrator about it.
**[O4]** - Since the voter names aren't used in the tally verifier they are no longer sent to the URL which means that they are no longer exposed in it.

#### 5.1.2.10 Current Trustees

**[O4]** - Since the trustees' e-mail addresses aren't used in the tally verifier they are no longer sent to the URL which means that they are no longer exposed in it.

#### 5.1.2.11 Trustees' Page

**[O5]** - Trustees can still generate the key pair directly in the browser and submit it if they wish to but it's no longer mandatory. It is now possible to generate the key pair in the browser in offline mode by providing the secret key - this will generate the public key and the POK. The trustee saves these, and can then manually upload them to the server with the browser back in online mode without ever having to submit his secret key to the server. Alternatively, the trustee may generate his public key locally using discrete logarithms and the election's fixed parameters and then submitting it to generate the POK. Then the trustee saves his secret key and submits the public key and the POK to the server.
Verifications were added to identify and display possible errors committed by the trustees when submitting parameters such as verifying if the parameters are numbers and that they have acceptable lengths or verifying if the format of the key submitted to generate the decryption factors is valid and that the POK verifies it.
Finally, when a trustee is verifying if his saved secret key is valid, a reminder to turn the browser's offline mode on is displayed, so that the user never submits his secret key to the server.

#### 5.1.2.12 Voting Booth

**[V1]** - The information that was more technical and not that useful for a common voter has been moved to the help guide. By removing the excess of information, it is expected to streamline the voting experience.
**[V2]** - Every button to go "back" is now displayed on the left of the window while the buttons to "continue" are displayed on the right of the window adding some coherence to the voting booth. Also a button with the option to go "back" to confirm the answers before casting the ballot was added.
**[V3]** - The size of the button to cast the ballot was increased in comparison to the others.
**[V5]** - Before casting a ballot, the voters are informed that their ballot tracker will be sent by e-mail although they should still verify if that ballot tracker is equal to the one displayed in the voting booth.
**[V6]** - After confirming his choices for the questions of the ballot, the voter now clicks in "Encrypt Secret Ballot" instead of just clicking a generic "Next Step".

**[O6]** - When a question requires to choose exactly 1 choice, the check boxes are replaced by radio buttons to avoid any possible confusion while switching choices.

#### 5.1.2.13 Audit Ballot

**[A1]** - Information was added to the audit tool stating that the audited ballot is discarded and cannot be counted to the tally. It is required to generate and encrypt a new ballot for it to be accounted for the tally.
**[O7]** - It is now possible to download some of the election's information necessary to audit ballots locally or in a different machine. The single ballot verifier was also exported and adapted in order to make this possible to happen.
When the ballot is incorrectly formatted, an error will now be displayed to the voter. An error will also be displayed if the ballot belongs to a private election and the voter does not have permissions to participate in it.
Since the tracking center has been removed, the option for the voter to post their audited ballot in it is no longer available.

#### 5.1.2.14 Confirm Cast

**[V2, V3]** - Voters that reach this step have already clicked to cast their ballot in the voting booth which renders pointless this cast confirmation. Therefore, this step is now skipped.

#### 5.1.2.15 Successful Cast

**[V2]** - The reminder to save the ballot tracker was removed and now it is informed that it will be sent by e-mail.

#### 5.1.2.16 Tally Verifier

**[O7]** - It is now possible to download some of the election's information necessary to verify the tally locally or in a different machine. The tally verifier was also exported and adapted in order to make this possible to happen.
The verifier now opens in a different tab/window since the verification of the tally can take a while and the verifier page or even the browser may freeze. Information stating that the verification of the tally may be a lengthy process is now also displayed in the top of the page.
Finally, a warning was added if the ballot tracker finder fails to find the inserted ballot tracker.

### 5.1.3 Final Map

After the mitigation of the issues, the platform map is again slightly changed being depicted in Figure 5.1.

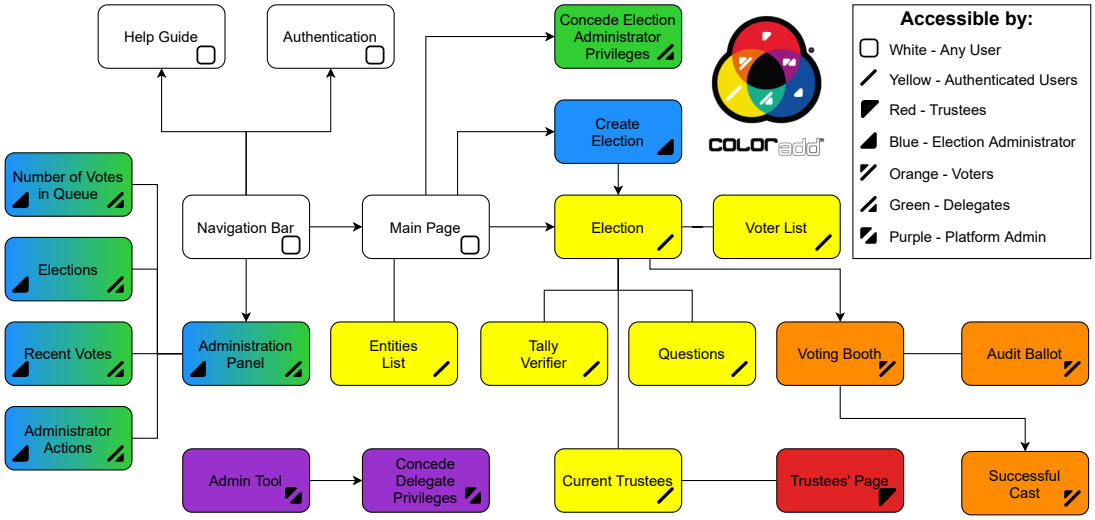



**Figure 5.1:** Helios Platform Final Map [3]

## 5.2 Unresolved Issues

The remaining issues that were detected with the field test and were not mitigated are displayed in this section.

**[A2, A3]** - It is not expected that most voters will use the auditing tool. The main goal of the verifying tools is to give assurance to the common voters that someone can and will use these tools and not necessarily being used by every participant. Consequently, it is admissible that half of the participants do not audit any ballots.

**[G2]** - Voters with incorrect attributes can contact the election's administrator so that he can manually add them to the election. This works as a temporary patch but it's not a solution to this issue. The IdP has to rectify the attributes of every user to ensure that they are correct and perform a routine maintenance to guarantee the health of them.

**[G1]** - The platform's presentation can also be massively improved, either in layout, colours and fonts. It should also be taken into account that members of the UL community with disabilities have difficulty using the tool and therefore accessibility standards for them should also be implanted in the platform.

**[O8]** - This issue can be mitigated in the IdP or in the platform.

- *IdP* - The DN is frozen and cannot be changed no matter what (with the exception of the RDN "OU" since users must have it equal to "Users" to access the platform);
- *Platform* - A different attribute is selected to be the unique identifier. This attribute must be unique, immutable and it should not be legal information, such as the civil identification number.

It was not possible to find an attribute sent by the IdP that would satisfy all 3 conditions and therefore this issue could not be mitigated.

## 5.3 Real World Election

The final implemented solution has already proven its utility - in May of the current year, the elections for the corporate bodies of the Academic Association of the University of Lisbon (AAUL) took place with a mixed electoral system for 2 days allowing online voting on the first and presential voting on the second [53, 54]. The electoral roll consisted of every UL student, totaling just over 50,000 voters. The online voting took place on the implemented e-voting system, where 2 trustees were assigned to it - Helios itself and one representative of the single candidate list. The online voting counted with the participation of 479 different voters running with two different types of incidents reported:

- A few participants reported that they could not authenticate in the platform due to errors resulting from the IdP;
- It was not possible to send e-mails informing the election results to every participant, because the quota of e-mails sent was, once again, exceeded. Even though this quota is theoretically fixed at 1000 e-mails per hour, it was not possible to send the 479 e-mails in bulk to the voters that participated in the election.

The presential voting that took place on the next day counted with the participation of 26 voters. This means that the election counted with the participation of a total of 505 real voters, just over 1% of all the potential voters. It is also worth noting that 94.8% of the real voters chose to vote electronically rather than in person, highlighting both the preference that voters have in casting their votes electronically rather than having to go in person to the stipulated polling place, and the confidence in the electronic voting system for a small scale election.
It is not possible to be sure how many voters ran into problems that prevented them from voting electronically, since many of them may not have seeked for help. Either way, a mixed electoral system with 2 days, the first electronic and the next being in person, turns out to be ideal to bypass any authentication problem while the IdP does not mitigate the reported authentication errors.

# 6

# Conclusions and Future Work

## Contents

This thesis presents the analysis and implementation of an already existing e-voting system in order to serve the UL's community necessities in this matter allowing both the creation of fully online elections and if necessary, hybrid elections. Even though the chosen system, Helios, is not state of the art such as other systems that use latter technology like blockchain, it has been proven along the years of its existence to be a reliable solution for low risk elections. This thesis also presents the modifications performed to the original solution in order to make use of the voters attributes defined in the IdP, followed by conducting a field test and a detailed analysis of every issue encountered in it along with the measures taken in order to mitigate them. It ends by stating the issues that could not be resolved and by detailing a real world election that took place using the final implemented e-voting system.

## 6.1 System Limitations and Future Work

The biggest limitation of the implemented solution is coercion [7, 55]. As stated before in this thesis, Helios does not offer reliable methods to resist coercion and therefore it should be used only in low-coercive and small scale environments, e.g., university student governments.

Helios is also vulnerable to ballot stuffing since a dishonest bulletin board could add ballots without anyone noticing.

Another current limitation is the necessity to trust the system when using both the audit and tally verifiers to ensure E2E verifiability. This limitation can be easily mitigated by distributing the verifiers across different machines which ensures a E2E verifiability if at least one of the machines is trustful. Both the single ballot and the tally verifiers have already been exported and adapted so that they can easily be implemented in the future.

Some users cannot authenticate in the platform via IdP which makes it completely impossible for them to participate in elections. Also, it has been reported that some users had their attributes configured incorrectly resulting in inaccurate permissions to access and vote on different elections. Finally, some users may still be able to change their unique identifier. These issues have to be solved directly by the IdP which must mitigate the errors that users encounter while trying to authenticate, assure the correct assignment and continual update of the users' attributes and send an attribute that may be properly used as the unique identifier.

As stated in Section 5.3, e-mails sent in bulk are still exceeding the previously set quota. It is therefore required further investigation to determine the reason for such occurrence.

If it is considered useful, a log out option may be added to the navigation bar for authenticated users. However, it is needed to be taken into account that the log out must be performed on the IdP since performing it directly on the SP is ineffective. This happens because the user will still be authenticated

on the IdP which means that he can re-authenticate automatically on the SP without needing to re-enter any credentials.

Another useful and interesting implementation would be, for example, Shamir's Secret Sharing Scheme. This cryptosystem would give more flexibility to the solution allowing the tally to be decrypted without requiring that every trustee submits their decryption factors. This could prove to be useful in the unfortunate event that a trustee loses his secret key which would result in a tally that could not be decrypted. With this cryptosystem implemented, the election administrator may configure the election in such a way that it is possible for the tally to be decrypted without it needing every trustee, therefore creating a safeguard for the situation mentioned before.

The platform's front-end also needs to be updated in order to boost the platform's presentation providing a more user-friendly interface and an overall better voting experience.

Finally, it should also be taken into account that some members of the UL's community have disabilities that may difficult their use of the platform. With this in mind, accessibility standards should be implemented in the platform, using for example the Web Content Accessibility Guidelines which are a set of recommendations that makes the web content more accessible, specially for individuals with disabilities.

# Bibliography


[1] Fleshgrinder and T. P. from The Tango! Desktop Project, "Simplified illustration of the full TLS 1.2 handshake with timing information," https://en.wikipedia.org/wiki/Transport_Layer_Security#/media/File:Full_TLS_1.2_Handshake.svg, 2015, accessed: 2021-06-28.

[2] K. Lewis and J. Lewis, "Web single sign-on authentication using SAML," *IJCSI*, vol. 2, pp. 41–48, 2009.

[3] M. Neiva, "ColorADD, color identification system," 2010.

[4] I. P. Institute, U. of Maryland, and C. Park, *Report of the National Workshop on Internet Voting: Issues and Research Agenda*. Internet Policy Institute, 2001.

[5] D. Jefferson, A. D. Rubin, B. Simons, and D. Wagner, "Analyzing internet voting security," *Communications of the ACM*, vol. 47, no. 10, pp. 59–64, 2004.

[6] A. D. Rubin, "Security considerations for remote electronic voting," *Communications of the ACM*, vol. 45, no. 12, pp. 39–44, 2002.

[7] F. Hao and P. Y. Ryan, *Real-World Electronic Voting: Design, Analysis and Deployment*. CRC Press, 2016.

[8] "Helios Voting," https://vote.heliosvoting.org, accessed: 2021-06-28.

[9] B. Adida, "Helios: Web-based Open-Audit Voting," in *USENIX security symposium*, vol. 17, 2008, pp. 335–348.

[10] "Helios Code," https://github.com/benadida/helios-server, accessed: 2021-06-28.

[11] "ULisboa — E-Voting System," https://e-voting.ulisboa.pt/, accessed: 2021-06-28.

[12] P. G. Neumann, "Security criteria for electronic voting," in *16th National Computer Security Conference*, vol. 29, 1993, pp. 478–481.

[13] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[14] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[15] R. Cramer, R. Gennaro, and B. Schoenmakers, “A secure and optimally efficient multi-authority election scheme,” *European transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997.

[16] A. Caforio, L. Gasser, and P. Jovanovic, “A Decentralized and Distributed E-voting Scheme Based on Verifiable Cryptographic Shuffles,” 2017.

[17] E. Bonvin, “E-Voting EPFL: Authentication and Frontend,” 2017.

[18] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Manubot, Tech. Rep., 2019.

[19] J. Benaloh, R. Rivest, P. Y. Ryan, P. Stark, V. Teague, and P. Vora, “End-to-end verifiability,” *arXiv preprint arXiv:1504.03778*, 2015.

[20] J. Benaloh, “Simple Verifiable Elections.” *EVT*, vol. 6, pp. 5–5, 2006.

[21] J. D. C. Benaloh, “Ballot Casting Assurance via Voter-Initiated Poll Station Auditing.” *EVT*, vol. 7, pp. 14–14, 2007.

[22] S. Goldwasser, S. Micali, and C. Rackoff, “The knowledge complexity of interactive proof systems,” *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.

[23] M. Furer, O. Goldreich, and Y. Mansour, “On completeness and soundness in interactive proof systems,” 1989.

[24] B. Adida and C. A. Neff, “Ballot Casting Assurance,” *EVT*, 2006.

[25] D. Chaum, “Surevote: technical overview,” in *Proceedings of the workshop on trustworthy elections (WOTE’01)*, 2001.

[26] B. Qin, Q. Wu, L. Zhang, and J. Domingo-Ferrer, “Threshold public-key encryption with adaptive security and short ciphertexts,” in *International Conference on Information and Communications Security*. Springer, 2010, pp. 62–76.

[27] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[28] J. D. Cohen and M. J. Fischer, “A robust and verifiable cryptographically secure election scheme.” Yale University. Department of Computer Science, 1985.

[29] J. C. Benaloh and M. Yung, “Distributing the power of a government to enhance the privacy of voters,” in *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, 1986, pp. 52–62.

[30] J. D. C. Benaloh, “Verifiable secret-ballot elections,” 1987.

[31] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[32] K. Sako and J. Kilian, “Receipt-free mix-type voting scheme,” in *International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 921 of LNCS. Springer, 1995, pp. 393–403.

[33] D. Chaum, “Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA,” in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1988, pp. 177–182.

[34] A. Fujioka, T. Okamoto, and K. Ohta, “A practical secret voting scheme for large scale elections,” in *International Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1992, pp. 244–251.

[35] R. Joaquim, C. Ribeiro, and P. Ferreira, “Veryvote: A voter verifiable code voting system,” in *International Conference on E-Voting and Identity*. Springer, 2009, pp. 106–121.

[36] R. Joaquim, P. Ferreira, and C. Ribeiro, “EVIV: An end-to-end verifiable Internet voting system,” *Computers & Security*, vol. 32, pp. 170–191, 2013.

[37] V. Cortier, P. Gaudry, and S. Glondu, “Belenios: a simple private and verifiable electronic voting system,” in *Foundations of Security, Protocols, and Equational Reasoning*. Springer, 2019, pp. 214–238.

[38] “Belenios specification,” http://www.belenios.org/specification.pdf, accessed: 2021-06-28.

[39] V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, “Election verifiability for Helios under weaker trust assumptions,” in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 327–344.

[40] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, and B. Ford, “Decentralizing authorities into scalable strongest-link cothorities,” *CoRR, abs/1503.08768*, 2015.

[41] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford, “{CHAINIAC}: Proactive software-update transparency via collectively signed skipchains and verified builds,” in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1271–1287.

[42] S. Cantor and T. Scavo, “Shibboleth architecture,” *Protocols and Profiles*, vol. 10, p. 16, 2005.

[43] “Lightweight Directory Access Protocol (LDAP): The Protocol,” https://www.hjp.at/doc/rfc/rfc4511.html, accessed: 2021-06-28.

[44] “IACR Elections Page,” https://www.iacr.org/elections/, accessed: 2021-06-28.

[45] “Django: A High-Level Python Web Framework,” https://www.djangoproject.com/, accessed: 2021-06-28.

[46] “PostgreSQL,” https://www.postgresql.org/, accessed: 2021-06-28.

[47] “The Apache HTTP Server Project,” https://httpd.apache.org/, accessed: 2021-06-28.

[48] “The CentOS Project,” https://www.centos.org/, accessed: 2021-06-28.

[49] “Helios IFSC Branch,” https://github.com/ifsc/helios-server, accessed: 2021-06-28.

[50] L. P. Alonso, M. Gasco, D. Y. M. del Blanco, J. A. H. Alonso, J. Barrat, and H. A. Moreton, “E-voting system evaluation based on the Council of Europe recommendations: Helios Voting,” *IEEE Transactions on Emerging Topics in Computing*, 2018.

[51] F. Karayumak, M. M. Olembo, M. Kauer, and M. Volkamer, “Usability Analysis of Helios-An Open Source Verifiable Remote Electronic Voting System.” *EVT/WOTE*, vol. 11, no. 5, 2011.

[52] C. Z. Acemyan, P. Kortum, M. D. Byrne, and D. S. Wallach, “From Error to Error: Why Voters Could not Cast a Ballot and Verify Their Vote With Helios, Prêt à Voter, and Scantegrity {II},” {*USENIX*} *Journal of Election Technology and Systems ({JETS})*, vol. 3, pp. 1–25, 2015.

[53] “AAUL - Election of Corporate Bodies, 2021,” https://www.aaul.pt/post/elei%C3%A7%C3%B5es-para-os-%C3%B3rg%C3%A3os-sociais-da-aaul-para-o-mandato-de-2021-2022-1, accessed: 2021-06-28.

[54] “AAUL - Result of the Election of Corporate Bodies, 2021,” https://www.aaul.pt/post/elei%C3%A7%C3%B5es-para-a-associa%C3%A7%C3%A3o-acad%C3%A9mica-da-universidade-de-lisboa, accessed: 2021-06-28.

[55] G. S. Grewal, M. D. Ryan, S. Bursuc, and P. Y. Ryan, “Caveat coercitor: Coercion-evidence in electronic voting,” in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 367–381.

# A

# Field Test Form

# Sistema de voto electrónico da UL

Este questionário contém 25 perguntas de escolha múltipla, e está dividido em 8 secções.
Deve demorar entre 5 a 10 minutos a responder.
A maioria das perguntas contém um parágrafo de escrita aberta opcional, que serve para nos ajudar a entender a causa dos problemas sentidos.

*Required

1. Email address *

2. Que dispositivo(s) usou para realizar a votação? *

*Tick all that apply.*

- [ ] Computador
- [ ] Smartphone
- [ ] Tablet
- [ ] Other: _____

3. 1) Achou a interface da plataforma de voto fácil e intuitiva de usar? *

*Mark only one oval.*

- ◯ Sim
- ◯ Não

4. Se não, especifique o que o fez sentir isso.

5. 2) Usou o guia de ajuda? (https://eduardoasdcosta.github.io/Helios/) *

   *Mark only one oval.*

   - ◯ Sim
   - ◯ Não

6. Se utilizou, achou útil e simples de usar? Se não utilizou, diga o porquê.

7. 3) Os seus atributos (presentes na página principal do sistema de voto após efetuar o login) estavam corretos? *

   *Mark only one oval.*

   - ◯ Sim
   - ◯ Não
   - ◯ Não encontrei os atributos
   - ◯ Não verifiquei se estavam corretos

8. 4) Sentiu algum receio de cometer erros ou de não usar a plataforma corretamente? *

   *Mark only one oval.*

   - Sim
   - Não

9. Se sim, diga o que gerou tal pressentimento.

10. 5) Houve algum termo mais técnico com o qual se deparou que não conseguiu perceber o seu significado ou a sua utilidade? *

    *Mark only one oval.*

    - Sim
    - Não

11. Se sim, qual/quais?

12. 6) Encontrou um email de ajuda na página de votação? *

*Mark only one oval.*

- ◯ Sim
- ◯ Não

13. 7) Percebeu o que é um rastreador de boletim de voto (ballot tracker)? *

*Mark only one oval.*

- ◯ Sim
- ◯ Não

14. 8) Percebeu que podia votar várias vezes, e que apenas a última vez será contabilizada em termos de contagem de votos? *

*Mark only one oval.*

- ◯ Sim
- ◯ Não

15. 9) Conseguiu aceder à cabine de votação (voting booth)? *

*Mark only one oval.*

- ◯ Sim *Skip to question 17*
- ◯ Não *Skip to question 35*

16. Se não, diga o que o impediu de aceder à cabine de votação.

Votação

17. 10) Sentiu alguma falta e/ou excesso de informação durante o processo de voto (voting booth)? *

*Mark only one oval.*

- Sim
- Não

18. Se sim, especifique exatamente onde / com o quê.

19. 11) Sentiu que o processo de voto é longo demais? (Demasiados passos, demasiado tempo?) *

*Mark only one oval.*

- Sim
- Não

20. Se sim, diga que aspectos causaram isso.

21. 12) Encontrou/teve algum problema em submeter o boletim de voto? *

*Mark only one oval.*

- Sim
- Não

22. Se sim, especifique o(s) problema(s).

23. 13) Conseguiu votar? *

*Mark only one oval.*

- Sim *Skip to question 25*
- Não *Skip to question 30*

24. Se não, diga o que o impediu de votar.

Votar

25. 14) Recebeu um email do sistema de voto da UL a confirmar o depósito do seu voto? *

*Mark only one oval.*

- Sim
- Não

26. 15) Tomou nota do seu rastreador de voto (ballot tracker)? *

*Mark only one oval.*

- Sim
- Não

27. Se não, diga o porquê.

28. 16) Teve algum receio com a privacidade do seu voto, que este não fosse enviado ou que fosse incorretamente enviado? *

*Mark only one oval.*

- ◯ Sim
- ◯ Não

29. Se sim, diga que receio e o que causou tal.

Auditoria de boletins de voto 1

30. 17) Tentou auditar algum boletim de voto? *

*Mark only one oval.*

- ◯ Sim *Skip to question 32*
- ◯ Não *Skip to question 35*

31. Se não, diga o porquê.

Auditoria de boletins de voto 2

32. 18) Sentiu alguma falta e/ou excesso de informação durante o processo de auditoria de boletins de voto (audit ballot, single-ballot verifier)? *

*Mark only one oval.*

- Sim
- Não

33. Se sim, especifique exatamente onde / com o quê.

34. 19) Percebeu o que significa publicar o voto auditado no centro de rastreamento (tracking center)? *

*Mark only one oval.*

- Sim
- Não

Após Votação

35. 20) Percebeu que um boletim auditado (audited ballot) não é contabilizado para a contagem? *

*Mark only one oval.*

- Sim
- Não

## Sugestões

36. Algo mais que queira comentar/adicionar ao seu feedback, ou simplesmente uma sugestão.