Tiago Alexandre Pereira Matos

UAlg Secure Vote Module

Universidade do Algarve

Faculdade de Ciências e Tecnologia

2024

Tiago Alexandre Pereira Matos

UAlg Secure Vote Module

MsC Thesis in Computer Science

Work done under the supervision of:
Professor Doutor Joel Guerreiro

Universidade do Algarve
Faculdade de Ciências e Tecnologia

2024

Statement of Originality

UALG SECURE VOTE MODULE

DECLARAÇÃO DE AUTORIA DE TRABALHO:

DECLARO SER O AUTOR DESTE TRABALHO, QUE É ORIGINAL E INÉDITO. AUTORES E TRABALHOS CONSULTADOS ESTÃO DEVIDAMENTE CITADOS NO TEXTO E CONSTAM DA LISTAGEM DE REFERÊNCIAS INCLUIDA.

CANDIDATO:

(TIAGO ALEXANDRE PEREIRA MATOS)

Acknowledgements

I would like to thank my supervisor Prof Dr Joel Guerreiro for all the help and support given during the development of thesis and arousing my interest in cybersecurity, all the work made on this thesis could only be made due to it.

I would also like to thank my parents for everything they've ever done and for all the motivation during this thesis.

Special thanks to my godmother Cristina and my cousin Gentil for all the help provided, this thesis wouldn't be possible without your help.

Special mention to my grandparents, although not among us anymore i know you would be proud of this thesis.

Finally i would like to thank all my friends and family who have dedicated a small amount of their time showing their interest on the work being made on this thesis and distracting me when i needed to "disconnect" from all this work.

Abstract

This dissertation proposes the development of an election system to be held, following the common electoral procedures. Since elections collect sensitive data, the system must be as secure as possible while respecting its users privacy. To register who voted on which candidate would completely hinder an election and the public trust on a system would be completely ruined. The objective of this web-based system is for users to vote from whatever location without having, physically to go into a polling station. A web-based system requires components for data processing or backend application, data storage i.e. databases and data presentation or frontend application. For a backend application, a REST API will be developed, this API is responsible for processing all data regarding elections and other features of the system itself. This API must be as secure as possible without compromising its users privacy, therefore a cipher suite must be used. The cipher suite describes the encryption being used on the system, where all votes must be encrypted before stored, so nobody can view it before the tallying process starts. Since asymmetric encryption is sometimes used and due to these keys being too long in length, a key management system must also be developed to store these keys. This key management system is an independent component of the main system to decentralize the keys from the election system. In case of the election system being compromised, the key management system is not necessarily also compromised. All data will have to be stored in databases since its the most efficient way to store it. The databases must take into account the system requirements, where in some situations, availability may be precedent over ACID capabilities. An user interface must also exist to interact easily and securely with the application and the system itself. This user interface must also encrypt the users votes so the backend application never "sees" the votes in plaintext when a vote is cast and stored.

Keywords: Elections, Electronic Elections, Secure Vote, Internet Voting

Resumo

Esta tese propõe um sistema para a realização de eleições, seguindo o procedimento eleitoral comum. Uma vez que as eleições recolhem informações sensíveis, este sistema deve ser o mais seguro possível, ao mesmo tempo que respeita a privacidade dos seus utilizadores, uma vez que nunca é registado quem votou em quê. Fazer isso prejudicaria completamente uma eleição e a confiança pública nas eleições seria arruinada. Este será um sistema web para que os utilizadores possam votar onde desejarem, sem terem que se deslocar fisicamente até uma mesa de voto, embora os utilizadores devam votar em locais privados uma vez que o sistema não consegue proteger contra outras pessoas verem os votos sendo submetidos. Um sistema web requer componentes para o processamento de dados, ou seja, uma aplicação backend, armazenamento de dados, ou seja, bases de dados, e apresentação de dados, ou seja, uma aplicação frontend. Para uma aplicação backend, será desenvolvida uma REST API, que será responsável pelo processamento de todos os dados relacionados com eleições e outras funcionalidades do próprio sistema. Esta API deve ser o mais segura possível, sem comprometer a privacidade dos seus utilizadores, sendo necessário usar uma cipher suite. Essa cipher suite descreve a criptografia usada no sistema, todos os votos devem ser encriptados antes de serem gravados, para que ninguém possa visualizá-los antes do início da contagem de votos. Como a criptografia assimétrica às vezes é usada e devido ao comprimento dessas chaves, um sistema de gestão de chaves também deve ser desenvolvido para grava-las. Este sistema é uma componente independente do sistema principal, a fim de descentralizar as chaves do sistema de eleições, caso o sistema de eleições seja comprometido, as chaves não serão necessariamente comprometidas. Todos os dados terão que ser gravados em bases de dados, pois é a maneira mais eficiente de gravar dados. As bases de dados devem ter em consideração os requisitos do sistema, uma vez que em algumas situações a disponibilidade pode ser mais importante do que as capacidades ACID. Uma interface também deve existir para que os utilizadores tenham uma aplicação fácil e segura para interagir com o próprio sistema, essa interface também deve encriptar os votos dos utilizadores, para que o aplicação backend nunca "veja" os votos em texto quando um voto é submetido e gravado.

Termos chave: Eleições, Eleições Eletrónicas, Voto Seguro, Votação através da Internet

Nomenclature

Abbreviations

ACID : Atomicity, Consistency, Isolation, Durability

AES : Advanced Encryption Standard

API : Application Programming Interface

CORS : Cross-Origin Resource Sharing

CSRF : Cross-Site Request Forgery

DBMS : Database Management System

DDoS : Distributed Denial of Service

DoS : Denial of Service

DSA : Digital Signature Algorithm

ECC : Elliptic Curve Cryptography

ECDH : Elliptic Curve Diffie-Hellman

ECDHE : Elliptic Curve Diffie-Hellman Ephemeral

ECDSA : Elliptic Curve Digital Signature Algorithm

EKM : Exported Keying Material

GCM : Galois/Counter Mode

HMAC : Hash-based Message Authentication Code

HTTP : Hypertext Transfer Protocol

HTTPS : Hypertext Transfer Protocol Secure

IT : Information Technology

IETF : Internet Engineering Task Force

JSON : JavaScript Object Notation

JWT : JSON Web Token

KDF : Key-derivation Function

KMS : Key Management System

MAC : Message Authentication Code

MD5 : Message Digest algorithm 5

NIST : National Institute of Standards and Technology

NoSQL : Not only SQL

PRF : Pseudo Random Function

RDBMS : Relational Database Management System

REST : Representational State Transfer

RSA : Rivest-Shamir-Adleman

SHA : Secure Hash

SOP : Same-Origin Policy

SQL : Structured Query Language

SSL : Secure Sockets Layer

STO : Security through Obscurity

TCP : Transmission Control Protocol

TLS : Transport Layer Security

URI : Uniform Resource Identifier

XOR : Exclusive OR

XSS : Cross-Site Scripting

Contents

List of Figures

List of Algorithms

CHAPTER 1

Introduction

1.1 Motivation and Scope

Elections are the key point of any democracy, they allow the population to choose an individual or groups of individuals to hold certain offices and its widely used to choose their representatives. Elections go far as the ancient Greece and Rome, where they used elections has a mechanism to select rulers. In more recent times elections are seen in almost every developed country and are used to select the ruling government for a mandate. Paper elections are the most common type of elections, where every voter selects their preference, by choosing a candidate from a ballot and places it into an urn. These votes are never associated with the voter and afterwards all votes are counted and the results are publicly displayed to the population. In recent times different forms of elections have been appearing, via letter, in which the post offices are used as a way to deliver the votes to the counting tables, meaning the voters do not have to physically go to a certain location to vote. Another way is electronic election, which uses electronic equipment to submit the voter's choices, these can either be by using a machine set in predefined locations, similar to the paper elections but without a paper vote, or using software allowing such elections to be held. Such software must be able to reliably held the elections without compromising users privacy in any circumstance, because privacy is the key component in any election regardless of its form. In software terms, the most common form of maintaining data privacy is data encryption, transforming data into not readable gibberish to both human and machine. This gibberish can be converted back into the data by using the used keys that created them. But when dealing with sensitive data the software must also take into account security, so the data being processed is

not compromised. Due to this issue the system has to be carefully designed and conceived to mitigate possible data compromises while keeping the data anonymous to maintain users privacy. This dissertation presents a system that was developed for remote elections to be held, using several technological and security components, addressing all issues and types of elections.

1.2 Objectives

This dissertation has the following objectives:

- Develop a system that follows the electoral procedure. This system has to permit the creation of elections, by inserting election dates, candidates and voters. The system must also permit its voters to cast votes and in the election end, the counting of such votes and disclosure of election results. The system must be secure to mitigate eventual data compromises.
- Develop an interface to allow users to easily interact with the API and minimize possible human errors when using the API functionalities. This interface must be easy to use and straightforward to minimize time loss when performing any action.
- Identify which cryptography algorithms can be used in this application. A large part of the data being processed in this application is sensitive and requires encryption before any storage procedure takes place. The encryption has to be strong enough to withstand possible crack attempts. It is also required to validate data origin and integrity to guarantee it was not compromised during the API requests and reply's validating if the data arrived from a authorized user using the application.

1.3 Thesis Outline

Chapter 1 introduces the Secure Vote module, which is a system for allowing elections to be held online. All the objectives are detailed with emphases on its features and operations while also detailing the system's security and cryptography.

Chapter 2 details what exactly is an election and how Secure Vote will be implemented, specifying what the system requires in order to properly hold elections, the system's conception, how the Key Management System, responsible for securely managing the encryption keys used by secure vote, will be implemented, which database systems will be used and finally how the system should be deployed.

Chapter 3 shows the different menus on the secure vote's interface and what operations can be performed on each menu.

Chapter 4 explains the cipher suite used in secure vote, this cipher suite defines all cryptographic algorithms, other algorithms and protocols being used by the system. All these operations are explained and how they are used by secure vote.

Chapter 5 details the security measures and policies used by secure vote in order to protect its endpoints and actions performed by users.

Chapter 6 concludes this thesis, briefly explaining operational and security features developed for secure vote. Future works is also presented on this chapter.

CHAPTER 2

Related Work

2.1 Election System Development

Election systems already exist and are already being used on some countries to perform the electoral procedures. Countries like Switzerland [1], Estonia[2] and Norway[3] are perfect examples where such systems are used.

Dyachkova et al. [4] developed a real time and completely anonymous system for casting votes in public networks. The system applies two user permissions: regular users and administrators. Regular users are allowed to vote and view the results while administrators are not allowed. Administrators can in turn create and delete elections. They can also create groups of users defining who is allowed to vote on specific elections. Users authenticate using a password that can be changed. To prevent influence on the voters choice, election results are only presented after the end of the election.

Varshney et al. [5] propose a system that keeps in mind the interest of the common user. The system firstly presents a homepage to allow the user's registration and authentication. The register requires the filling of some information and automatically an id and password are created for the user authentication. When the election day arrives, voters can then submit their votes but with only a 10 second window. After submission a numeric code is generated to help finding the voter voting location. If the vote is successfully casted then the system displays a message, otherwise the voter will have to recast the vote again.

Alamleh et al. [6] suggested that any system, capable of executing an election, must be audible for official entities ensure data integrity and if the system was not compromised. To

perform this, all data must be stored in safe environments that can maintain its confidentiality, integrity and availability. Election logs must be recorded and stored as well. A system must also be sustainable, maintained over a period of time and meet the approved requirements.

Sliusar et al. [7] propose a system based on Blockchain technology. The system requires users to confirm their identity and make sure the user is indeed eligible to vote on the election by presenting an identity document to a operator. The operator is an independent third party authorized to verify the document. The user reads a generated QR code allowing the operator to register the voter into the Blockchain, and automatically the system generates a token that functions as a ballot associated to the user. When the voter casts his vote, a transaction between the user wallet, created during registration and the candidate's wallet, is generated. This process maintains anonymity since the voter cannot be traced to its identity. The generated token has a time limit, set by the Election Administrator, that will become invalid after the limit finishes, and the option to cast a vote becomes inactive. The voter can at any time view its vote and if it was counted in the tallying procedure. To properly use the system several interfaces are supplied, one for the Administrator, one for the Operator and a mobile application for the Voter.

Schmidt et al. [8] state that election systems should be outsourced to verified third parties for their technical implementation and operation. These third parties, called Voting Service Providers or VSPs, can provide secure buildings, the hardware, software and required skilled personnel to perform a secure electronic election, releasing the host from these tasks, being more feasible these types of elections. The VSPs must be trustworthy thus they must be regulated and supervised by proper authorities. When using a VSP, several actors with different roles are used: the VSP who operates the electronic election on behalf of the host, the election host which is the institution that holds the election, the voters who can vote on the election and the public.

Ruth et al. [9] evaluated if internet voting could or not be the next step in governance, without physically having to go to a polling place. It was used in the United States on many elections but never on national elections. The authors stated that usage of remote elections are not a matter of technological challenge but a political one. Obviously, these elections have problems handling data security, password protection, voter anonymity and server overload, but even if they were solved, representation of constituencies, discrimination and other political issues would still be concerns to public officials. European countries have also tried to use remote election systems and is noted that voters who usually describe themselves as abstainers choose internet voting. USA tried identifying voters using their social security number, password, birth date and place.

Spain, on a pilot experiment, tried identifying voters by using an authorized smart card and a PIN code, where every voter who had access to the internet could use the system, but the experiment turnout to be disappointing due to only a small number of the voters used the system with allegations of fraud and software glitches. So far, the leader in federal internet elections is the country of Estonia even with a low percentage of voters using the system. To vote, using this system, voters are required to have a valid Estonian digital identification and two PINs, one used for entering the system and casting the ballot and another for encrypting the ballot and sending it to the server. Voter anonymity is kept using "double envelope" security in which the digital ballot is not deciphered until the voter's identity was separated from it. Several Estonian parties expressed concerns over the usage of the system regarding the secrecy of the vote, they also expressed concerns, due to the unsupervised nature of the system, making it impossible to observe thus creating potential for illegal pressure, coercion or inducement of voters, such occurrences may take place in the voter's home or workplace and any person with a laptop and ID card reader could travel to voters residences and "collect" votes. This system also did not meet expectations with only a small percentage occurring on the system. These remote election systems are convenient and can be secured and the voters anonymity be maintained, but to ensure that votes are genuinely free of choice and prevent someone from standing over the voter's shoulder trying to change the voter's choice is not avoidable. On regular paper elections this type of persuasion does not occur.

Kate et al. [10] developed a system for elections in which users register by filling their information, and from that data, a message digest is created and then embedded into a random selected image by the server. This image is splitted into two shares. One is sent to the user via email and the other is saved on the server database. To authenticate, users send their share, the server uses it and the other stored share to regenerate the image and extract the digest. If it matches, the user is authenticated.

Yang et al. [11] suggest a system based on the Three-Ballot model, where after the user is identified from the administration center, the voter fills his ballots and sends them to polling. Polling counts the ballots and sends them to be stored and audited. When the election ends, the votes are counted, encrypted and sent to the bulletin board to publish the results. Before any election, Administration, Polling, Storage and Auditor generate each a key pair to allow secure communication. Voters authenticate using their ID and password, after which a message is sent to Administration to verify if the voter is eligible. If the voter is eligible, then it may proceed

with filling and casting the ballots created by the Ticket center. The ballots are filled according with the features of Three-Ballot to protect the voter's privacy. After filling the ballots they are digitized, blinded and a blind signature from Administration is applied on every ballot which then deblinds the signature. The ballot and the signature are then encrypted with the Polling public key and sent. The voters keep one of the ballots as confirmation. After reception, Polling deciphers the ballot using the private key and verifies the signature. If the signature is verified then the ballot is counted to the result, encrypted and sent to Storage and Audit. If the signature cannot be verified, the vote has been tampered and does not count. When the election ends, Bulletin Board publishes the final results.

Agarwal et al. [12] system requires the existence of a polling booth for voters to cast their votes. Voters register in the system using their fingerprint biometric and to cast the vote, the system requires the fingerprint validation . If the fingerprint is validated and the voter has not yet voted, then the voter can cast their vote in a 10 second window and the vote is stored.

Foster et al. [13] propose a system that combines several developed voting systems. This system requires its eligible voters to register with their LEO beforehand, physically or by using a three-way mailing method. The first, requires voters to visit a polling site before every election and the second requires voters to have time for three-mailings exchange. The system itself is contained on a bootable CD that contains a complete, minimal OS. This requires voters to be able to configure their computers to boot from a CD, insert the CD and turn on the computer. This CD is mailed to each voter weeks before each election, regulations and penalties for tampering with absentee ballots can be modified to fit the usage of these CDs. Voters will have several weeks to use the CD, with the end date before the start of the paper elections. This allows the LEO to adjust local voter list to prevent repeating votes from those who vote using the system. Apart from voters the system also has administrators and managers.

Shirazi et al. [14] developed a voting system to be robust and resistance to coercion. The system must be always available and be fault-tolerant. It must allow voters to cast their votes even under attack. The system's roles are Supervisor, which administers the election, Register, which authorizes voters and posts the list of authorized voters, Voter, which is a user authorized to vote, Registration Teller, which is responsible for the distribution of the private credentials of each voter and for posting the corresponding public credential on the bulletin board and Tabulation Teller, which is responsible for mixing, decipher and tallying the votes. The system uses two storage services, a Ballot Box used to store the casted votes and a Bulletin Board used

for all kind of information for election verification. During the voting process, a voter has to authenticate, after receiving the required credential from the registration teller. Afterwards, a voter can cast the vote, which is encrypted and sent to the ballot box(es). During tallying the votes are retrieved from the ballot box(es), where invalid votes are eliminated and the remaining votes are shuffled, deciphered and tallied. The result is posted on the bulletin board.

Usmani et al. [15] developed a voting system focused on enhancing secure and authenticated voting. A user can both create an election or vote on elections. No registration process exists on the system, it uses the National Identification Number to authenticate users. To create an election the user must authenticate using its National Identification Number and verify its identity by scanning the National Identification Card barcode. On the next step, the user automatically receives an OTP and the system requires the OTP insertion. If the user successfully verifies its identity and fills the required fields, the data is stored in the database. The user is now the election admin and receives two unique codes, one to be sent to the election voters and another for the admin to be able to count votes. For a voter to vote it must first authenticate using its National Identification Number and enter the unique ballot code received from the election admin. After verifying the user identity with the card barcode scan and inserting the OTP, the voter can cast its vote. The vote is encrypted and stored in the database sending a notification informing that the vote has been successfully submitted.

Lambrinoudakis et al. [16] state that an electronic voting system is a system that records, stores and processes election data, so, it must provide the required services for organizing and conduct an election, support all actors that interact with the system, different types of electoral procedures, be customisable and ensure that only eligible voters can vote, a voter can only vote once, votes are secret, every vote is tallied and voters trust their votes are counted. These systems must follow the conventional election process thus it must allow actions like an authenticate actor or vote casting. Each system actor has their own role defined in how it interacts with the system. The identified roles are Election Organizers, who ensure the election is properly conducted, Election Personnel, who creates and manages the election under the Organizers supervision, Judicial Officers, who monitor and ensures the election is performed properly and legally, Party Representatives, who monitor the election but do not interact directly with the system, Independent Third Parties, who audit the system's operation and actions performed by the system's actors respecting the election's integrity and voters anonymity who actually participate in the election.

I. M. Rodiana et al. [17] proposed a system for electronic voting that has five stages: Registration, Voting, End of the Day, Central Collection and Final Repository. Users register using their resident ID, after receiving a QR code with the voter's unique number, the terminal location, where the voter can cast the vote, and a pair of keys. During the voting period, voters receive an encrypted empty ballot from the committee only deciphered with the previously received private key. After voting, the voter receives an hash that can be used for the voter to visualize results. The voters identity is encrypted to maintain anonymity and can be used to prevent repeated votes. At the end of the day, ballots are randomly stored and voters can access votes using the hash. In each terminal, the committee will produce a tabulation that has to be verified by both, committee and witness. After verification and counting, the data is signed by both parties to ensure authenticity. During central collection all terminal data is encrypted using the central committee public key and sent to central committee to verify every terminal's data. During final repository all data has been verified by central committee.

P. Annar Shankar et al. [18] propose a system for a migrant to cast votes. The system consists on three modules: a mobile application, a backend server and a web application. The mobile application verifies the voter's ID using an OTP. Voters can, temporarily, change the voting location where is required a voter photo, a digital copy of his ID and the new location. Failing one step results in a failure. Voters can also verify their activity by scanning an encrypted QR code present on the web application on the voting station. The backend server is responsible for storing the voter's data such as name, gender, mobile number, date of birth, address, constituency, photo, voter ID number and national ID card number in a database. This server also processes user's requests, like the changing of the photo or the voting location. Generating parties list and counting votes are also handled at the backend server. On election day the voter scans a QR code on the website and sends details to the server, triggering an event on the particular voting location. The web application on the polling station listens to the trigger and displays the constituency list allowing the voter to cast a vote, being encrypted and stored anonymously to avoid tampering. The web application is placed at the polling station by the election commission. After displaying the list of parties the application enables a 3 minute timer, camera and microphone to avoid malpractices. If the voter's face is detected the vote buttons are enabled. After the vote casting, data is encrypted and sent to a distributed server, avoiding, therefore, data tampering. Each voter must be physically present on election booth to be able to cast the vote and avoid forgery.

As described above several solutions for a voting system were addressed by distinct authors, each with advantages and disadvantages. Dyachkova et al. [4] system has the advantage of having two different roles, each with permission to access certain features of the system but the disadvantage of requiring different systems for each role forcing the user, who both votes and manages elections, to have two separate accounts, cannot be a good feature. The authentication process only requires a password and there is no way to recover the access if the user forgets the password. Comparing with the Varshney et al. [5] developed system, the requirement of having a system for each role is no longer valid and the user who creates the election becomes directly its admin. The system recognizes different roles without requiring distinct systems and an ID, generated by the system, and a password for user authentication provides extra security compared with Dyachkova et al. system. The biggest disadvantage of Varshney et al. [5] system is the vote casting functionality within only a 10 second window, that may not be enough time for the voter to choose, nevertheless, this functionality may prevent coercion. Sliusar et al. [7] uses Blockchain technology, an excellent approach, because Blockchain transactions are audible, anonymous and maintain integrity. Dyachkova et al. and Varshney et al. systems cannot completely guarantee integrity and anonymity. A major disadvantage of the Blockchain system is the usage cost and the inconvenience requirement for voters to physically register through a third party. Kate et al. [10] system has basically the same features previously mentioned only diverting in the authentication method using a file, because it is more secure and complex to forge a random image file, but may cause problems if the user loses the file, deletes or even if it is stolen. Yang et al. [11] systems relinquishes the standard single ballot model and focuses in a three ballot model, where a voter can verify the process, what other systems cannot reproduce, except the Blockchain system. This model major disadvantage is because it is more complex to use compared to the single ballot model making it less convenient to use thus leading to users not adhering to its usage. Agarwal et al. [12] system contains basically the same features as other systems described before but uses fingerprints for authentication making the system more secure due to forging fingerprints complexity, but relying only on biometrics to authenticate can be a disadvantage where a voter can be in a situation where it cannot use the fingerprint and the requirement to physically address a voting station to vote. This fingerprint authentication can be used but complemented with another authenticating method like a password, similar to how modern smartphones operate. Foster et al. [13] system requires voters to receive the system in a bootable CD, which is more secure compared to other systems but requiring encrypted

internet connection to transmit the vote. To use this system, users must physically register or use three-way mailing, reducing the system's convenience and are required to vote before the paper elections start to avoid repeating votes. The main disadvantage compared to other described systems is the bootable CDs usage, because nowadays are not commonly used and the requirement for users to boot their computers using this bootable CD which requires technical knowledge that some users may not have. The distribution of these bootable CDs can also be quite expensive depending on how many voters are and the software is specifically for one election only, being, therefore, more expensive than the Blockchain system, that is better and more convenient to use. The whole system infrastructure must also be adapted to a more modern infra-structure than a CD.

Shirazi et al. [14] system is convenient to use where users are not required to physically address themselves to a voting station, also because of the distributed credentials usage, making elections securer, and by using multiple ballot boxes increasing availability, where at least one ballot box is functional the election can always be held. This system however has the disadvantage of not being transparent because voters cannot check if their votes were used for tallying. Systems using the Blockchain, where all transactions can be checked, or models like the three-ballot, where the voter receives a receipt after voting, are better because voters have the ability to verify their vote. Usmani et al. [15] system authenticates users through the personal ID card, although the system has no roles thus every user can create elections and vote. The disadvantage is that voter lists cannot be set thus every user can vote in every election, even not eligible users. Creating an election and voting requires entering an OTP, making the system securer and after the OTP is used it cannot be reused again. This system, if poorly implemented, can lead to security flaws because other voters could use the OTP to vote on behalf of another. Confirming the cast vote via email can provide some transparency but not as much as other systems described before. I. M. Rodiana et al. [17] system is similar to the other described systems with the difference that the voter receives an encrypted ballot and a hash after casting the vote. Receiving the encrypted ballot ensures nobody else can access the ballot, although, usually election candidates are known publicly thus it's not very advantageous to encrypt it before. Any vote casted reduces the system performance, and receiving an hash after the vote increases transparency and can be used to check if the vote was tallied, working on par with the receiving of an email confirming the vote has been casted, like Usmani et al. system. This may lead to certain issues because hashes can be brute-forced, votes can be compromised and

discovered, before being tallied and, even more severe, these hashes are unique to every voter making it possible to know the vote itself and the anonymity of a specific voter in the election is compromised. An election system must allow a proper electoral procedure to be held requiring features to properly secure users authentication as well as the elections setup, the voter lists, vote tallying and the results presentation.

P, Annar Shankar et al. [18] system authenticates users by mobile application and like as Usmani et al. system uses an OTP, requiring only the user's personal device. This system may lead to problems when devices are compromised creating issues not just on authentication but also in the voting procedures when voters are required to scan a QR code to vote. Every system must be audible [6], thus votes must be stored and logs must be registered for possible audits. Authentication must be convenient while being secure. Using an ID or a email with a strong password is the best option that fits both of these requirements, other technologies like fingerprint biometric reading can also be used as a complement for authentication. Ballot boxes must be available at any times, emphasizing availability with encrypted votes to prevent possible data compromises and results being only revealed after tallying without any way to trace them back to the voter. For convenience purposes, no voting window should be used, allowing voters the required time to cast their vote in a secure and private location of the voter's choice preventing possible coercion attempts. Following Lambrinoudakis et al. [16], every system must be able to handle any type of election and supply all requirements, this can be facilitated by using a VSP, suggested by Schmidt et al. [8], where all these requirements are addressed by an authorized and capable entity to deploy and manage these systems. The major question regarding these systems is not if it is technologically feasible, but if they are politically accepted. Ruth et al. [9] states that data security and voter anonymity problems will always be present and greatly contribute to the political issues involving remote voting. There have been many attempts in many different countries to use these systems and the results have always been disappointing although it resulted in a reduction of the so called "abstainers" because these systems provide a convenient method to vote although coercion is always a possibility because someone may be looking over the voter's shoulder to make sure the voter votes "correctly".

2.2 Cryptography and Security

Election systems handle sensitive data, that cannot be compromised in any circumstance. Obviously, these systems must have strong cryptography and security features to prevent or, in worst situations, mitigate possible data compromises.

The system developed by Dyachkova et al. [4] uses an algorithm called "blind" Signature, based on the RSA. Blind Signature algorithm allows the creation and of a ballot signature without disclosing sensitive information, like an option chosen as vote, and the signing protocol can be easily changed without violating the algorithm. Data transmission uses ring signature protocol to protect voter's anonymity and prevent repeated votes from the same voter. The system is deployed as a node network, all connected to each other and to a server, that receives data from all connected nodes. The protocol does not require any server response, maintaining voter's anonymity. The system provides administrators and regular users interfaces, to prevent any user to access the source code and gain administrator functions. Due to how the algorithm works, it may be possible to forge votes from the ballot of another user, so the system, to prevent this, requires new keys generation for each vote.

Varshney et al. [5] system encrypts the generated number using AES, and then is stored into the database. Due to AES complexity, even on its smaller key size (128 bits), the code maintains its confidentiality. The access credentials are randomly created after filling the required information making it harder to attack. The system does not allow users to access the ballot after the vote has been casted.

As stated by Alamleh et al. [6], an election system must be able to protect data from an unauthorized disclosure or access. Data confidentiality, integrity and availability is imperative in these systems. Voting systems must control the user identification and authentication and if he is an eligible voter in the election. The officials must be authorized to handle the election and can be accountable for any performed action bacause the system can trace back all actions. To maintain the election confidentiality, the system must perform anonymization to protect the identity of its users.

The system proposed by Sliusar et al. [7], using Blockhain technology, allows increased transparency to the voting process and electoral authorities are not needed. To guarantee voter anonymity, only the user knows exactly which vote is his. The system also improves reliability and security by the Blockchain stored data and due to the fact that its impossible to hack a block

without affecting the other blocks avoids possible vote tampering. The Blockchain technology guarantees fault tolerance due to it's decentralized features and if one or more devices fail, the system will carry on working. This technology increases civic engagement because the voter can participate in an election regardless of the location, increases efficiency because it is faster and cheaper than a paper election and increases processing speed due to the decentralization of the Blockchain allowing each region to operate its own node distributing the load. But, the system has also a few issues, there is no automated service to guarantee the voter identification and if someone who identifies into the system is indeed the voter.

Schmidt et al. [8] requires VSPs to host and operate all hardware components. If connected to external networks they must support secure communication via SSL/TLS protocol. Several servers are used, one configures and customizes the voting system according to the specific election, another stores all voters ballots, a third mixes and deciphers votes after casting, a fourth tallies votes and a fifth publishes results. Some servers are connected to the external network while others are not. Trusted components like communication, storage and erasure, cryptography, logging, monitoring, installation and configuration and personnel are required for a proper electronic election to be held by a VSP. The VSP must also maintain the system availability and protection, physically and logically.

Kate et al. [10] developed system uses a message digest embedded into an image to register and authenticate users. This message digest uses SHA2 hashing algorithm generating a 256 bit string. SHA2 is one of the strongest message digest algorithms. After the vote cast, is encrypted using AES and stored into the database. When tallying starts, votes are deciphered, counted and then announced.

Yang et al. [11] suggested system uses user's private information to authenticate, all other processes do not require any private information. The system verifies all processes end-to-end, holding one voter's ballot that will be used to verify the Bulletin Board results, therefore the public can calculate the final result and compare them with the authority published results. Using the Three-Ballot model ensures coercion resistance because it is impossible to prove the voter's vote by the receipt. This system does not require any cryptography to encrypt the votes while casting, they are encrypted and signed when sent. To tally, the signature is verified and the vote is deciphered, which is an advantage because every voter has his own public key and by using a public key infrastructure, reduces encryption and decryption computation while data communication efficiency is improved.

Agarwal, Samarth and Haider, Afreen and Jamwal, Abhishek and Dev, Param and Chandel, Rajeevan [12] system authenticates using voter's fingerprint biometric previously stored and compared using hashing. Votes are not encrypted and only the vote casting is registered to prevent the same voter from repeating the vote.

Foster et al. [13] proposed system requires users to boot an OS from a CD to access the voting system. The system is able to reach voting servers through a modem pool connecting via a toll-free number encoded on the CD. Using telephone numbers allows telecom companies to block and trace DoS attacks and are harder to spoof than IP addresses. After the voters boot the system an ID and private key is required, and then the voter accesses the ballot. After the ballot is completed and confirmed the system blinds it and creates a partially unblinded scratch ticket, to be confirmed as a voter receipt. To prevent the ballot being associated with the voter, information is encrypted with the voting servers' public keys before it is sent. After the voter identity verification is processed, every ballot is signed by the administrators, a ticket is generated and encrypted to be verified by the tallying server. Managers sign after the administrators without knowing which administrator's signed, preventing acceptance or rejection based on the list of administrators that signed the vote. The manager's digital signature is required to prevent a voter from submitting multiple ballots. After that procedure, the message is anonymized and encrypted with the generated public key and the voter's Id is sent to the server for authentication. If the authentication is valid, they are transferred physically to a web server so voters may verify if their ballots have been correctly received by the system. The tallying server receives every blind ballot, the tickets and keys. If the ballot is authenticated it is then checked for correctness, then counted and the results are displayed.

Shirazi et al. [14] specify that their system uses Decisional Diffie-Hellman, RSA and SHA-256. Multiple ballot boxes are used simultaneously, therefore not all can be corrupted, at least one of the boxes will be able to correctly handle the voter's vote. Votes are casted on an anonymous channel. For the system to work properly, voters must trust the equipment and at least one registration teller. Adversaries cannot simulate a voter, doing so would make both, adversary and voter, valid voters and both could trigger a valid vote.

Usmani et al. [15] developed system requires users to use their National Identification Card to authenticate. An OTP is used to verify the users identity therefore another user cannot cast a vote using a stolen ID card. The submitted votes are encrypted by AES before they are sent to the database. After recovering the data from the database it is then decrypted by the election

admin after which it can be used to evaluate and count the votes.

Lambrinoudakis et al. [16] state that trust is a critical success factor in an election. In electronic elections trust can be achieved by assuring that nothing can manipulate the system. Enforcing measures such as the participation of several actors and implementing duties separation thus making some features only accessible to some roles. Every action must be registered so as to held the actor who performed it responsible.

I. M. Rodiana et al. [17] uses RSA algorithm with 1024 key length asymmetric encryption to encrypt data and MD5 message digest for hashing. The system requires a key management scheme to securely use RSA keys. Key are generated for the committee and witness before registration and the public key is sent to the CA. Voter's keys are generated during registration and sent to CA. Key distribution is how keys are distributed. Public keys are sent to CA and can be accessed anywhere while private keys are never distributed. Key lifetime is the time schedule duration the keys remain valid. This system keeps keys valid for two months. Key can be regenerated or deleted with central committee authority.

Cetinkaya, Orhan [19] states every voting system must have cryptographic voting protocols and must satisfy several requirements: Voter privacy, Eligibility, Uniqueness, Fairness, Uncoercibility, Receipt-freeness, Accuracy and Individual Verifiability. Due to how contradictory some of these requirements are, the possibility of fraud is unavoidable in an electronic voting environment because ensuring trust is very hard.

Due to how sensitive the voting system data is, becomes necessary to implement measures to protect both system and users. To keep the data secure and anonymous, a cryptographic solution is required. Dyachkova et al. [4] system uses "blind" and ring signatures, providing a solution to anonymously cast a vote, verify origin and preventing repeated votes. By using "Blind" signature it is possibly to forge votes if the same key is used, therefore a new key is generated for every vote, so the voter is not able to vote on behalf of another. Using a network of interconnected nodes with a server that sends no responses, increases the voters anonymity but makes it impossible to perform data validation because it is performed on client side and not in the server side, bypasses can occur leading to system errors if bad data is received. On the other hand, Varshney et al. [5] uses AES for encryption, maintaining data confidential. Sliusar et al. [7] used the Blockhchain cryptographic suite, which is more secure and private than other systems but much more expensive to implement and maintain, being the main disadvantage because elections always have budgets that cannot be exceeded.

Kate et al. [10], like Varshney et al., encrypts votes using AES but to authenticate data uses SHA2. Using message digests or hashes to authenticate users can lead to unauthorized accesses because hashes can be brute-forced and the system does not confirm the voter identity, which may lead to vote casting by another unauthorized user. Yang et al. [11] three-ballot model system does not require any vote encryption before data is sent to the server being impossible for a vote to be traced back to the voter. This system uses signatures to guarantee votes are from eligible voters, combined with the three-ballot model makes the system coercion resistant. Although not specified, this system can use AES to encrypt votes, just as other described systems. Agarwal et al. [12], on contrast with other systems, does not encrypt the casted votes, just stores them after authentication. The system does not use a cryptographic suite and requires voters to physically vote on voting stations. The votes are not transmitted, so it does not require any vote encryption, nevertheless, votes should be encrypted before storage using, for example, AES to prevent possible data leaks which may lead to vote coercion. The system registers voters who have already casted their votes to prevent vote repetition but does not provide any means of transparency. Foster et al.[13] system is secure and anonymous but has the disadvantage of using obsolete technology that can be updated to a more modern approach, replacing CDs with SD cards and modem pools with VPNs. This system encryption uses a public key that can be securer than using AES, because the public key cannot be used to decipher and the private key cannot be calculated using the public key. The system requires a secure storage for the private keys due to the asymmetric encryption of RSA requiring a public/private key pair to correctly operate. To secure the private keys the system could use AES symmetric encryption because of a single key usage for both encryption and deciphering. If the encryption system is not correctly implemented it may lead to keys being compromised thus compromising all cast votes. Dyachkova et al. system is an example of what could be adapted for asymmetric encryption usage, because the server could supply public keys to the nodes and private keys could be stored. Shirazi et al. [14] uses a random oracle, this oracle responds with unique responses to repeated queries obtaining the same response. The oracle can be used to confirm if a vote is in the ballot box increasing transparency. Similar to other systems described before, using RSA is an great approach for voting systems because of the freely shared public keys usage but requires private keys to be securely stored. Usmani et al. [15] system, just as other systems like Varshney et al. and Kate et al., uses AES to encrypt the votes before storing, but encrypting only before storage may not be wise since it means the system temporally knows the vote and the

voter who casted it, removing any anonymity critically required by an election. As stated by Alamleh et al. [6], every system must be able to protect data from unauthorized disclosure and data confidentiality, integrity and availability are required thus its possible to define the usage of a cipher suite. TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 is a cipher suite that satisfies Varshney et al., Kate et al., Yang et al., Foster et al., Shirazi et al. and Usmani et al. systems due to all these using AES but specially because how sensitive the election data is TLS_ECDHE_ECDSA_WITH_AES_256_GCM_should be used instead, it provides increased security with a larger key size for AES and uses GCM mode to verify data integrity, guaranteeing that the data deciphered is indeed correct preventing vote tampering. Systems like as Kate et al., not directly using AES, uses SHA256 also known as SHA2, Yang et al. and Foster et al. systems use signatures which can be created by ECDSA, much more efficient than RSA when signing but due to being an ECC algorithm may lead to weak encryption, if a weak elliptic curve is used. But RSA can and should be used if the vote is to be encrypted on the client side, systems such as I. M. Rodiana et al. [17] encrypts their votes using RSA but a secure key length must be defined to not cause a weak encryption. As stated by Cetinkaya, Orhan [19], a voting system must have some sort of cryptographic protocols to satisfy several requirements which can be achieved by defining a cipher suite and TLS_ECDHE_ECDSA_WITH_AES_256_GCM_is an excellent choice, as shown. TLS protocol secure communications, ECDHE can be used to share secret keys, ECDSA can be used to sign votes, AES_256_GCM can be used to encrypt the votes but vote encryption should be made on the client side so RSA should be used to encrypt the votes due to the capability of the public key be safely distributed without causing any security issues. RSA private keys must be securely stored, and encrypted by AES, and if required, sent via secure communication using both AES and ECDHE. SHA256 can also be used for integrity checks but its susceptible to brute-force attacks thus HMAC should be used with SHA256, making the hash more complex to brute-force. Trust is of utmost importance for an election, as stated by Lambrinoudakis et al. [16], and as such measures must be implemented and enforced to prevent manipulation of the system. Ruth et al. [9] suggests the use of "double envelope" security where votes are only deciphered during tallying, maintaining voter anonymity while reducing vote coercion. To facilitate the implementation of a cryptographic suite, a VSP can be used, as stated by Schmidt et al. [8], because VSPs will host and operate the system and should use their own cryptographic suite. If an entity cannot guarantee proper security and anonymity, it should outsource hosting and operation to a VSP. Using the Blockchain is also a good option due to the

fact that is secure, private, has its own cryptographic suite and removes the need to use a VSP. Blockchain has one drawback, because it is quite expensive to implement and maintain.

CHAPTER 3

System Conception

3.1 Election

An election can be interpreted as a concept of people making choices. This choice making usually involves a set of considerations and a rule for combining them. Both of these influence an individual's choice, an already made choice or a choice to be made. The collective of these individual choices is what are called elections[36].

The mainstream election begins by creating a ballot containing the election choice options, candidates or simply answers like as yes or no, and defining the electoral period. During this period, voters which are allowed to cast votes (eligible voters) can submit their choices on the voting tables and place the vote inside an urn. Every single vote is anonymous and cannot be traced back to who submitted it, but the voter is registered so every voter can only submit one single vote. These urns are sealed from the beginning up until the process of tallying is initiated, where the votes are counted and revealed. Results are made public and every voter can check the outcome.

Electoral fraud is a phenomenon that can happen in an election and if the results are suspected of fraud, the usual procedure is to perform an audit. Audits are preferably performed by an independent unbiased group from the election. If fraud is indeed proved, results have to be denied and the election must be redone. If an audit cannot prove if the suspicion of fraud is real, the election should be redone as well, just to be safe, voters have the right of choice and that has been put in jeopardy. But if no fraud is proved, results are accepted and decisions are made.

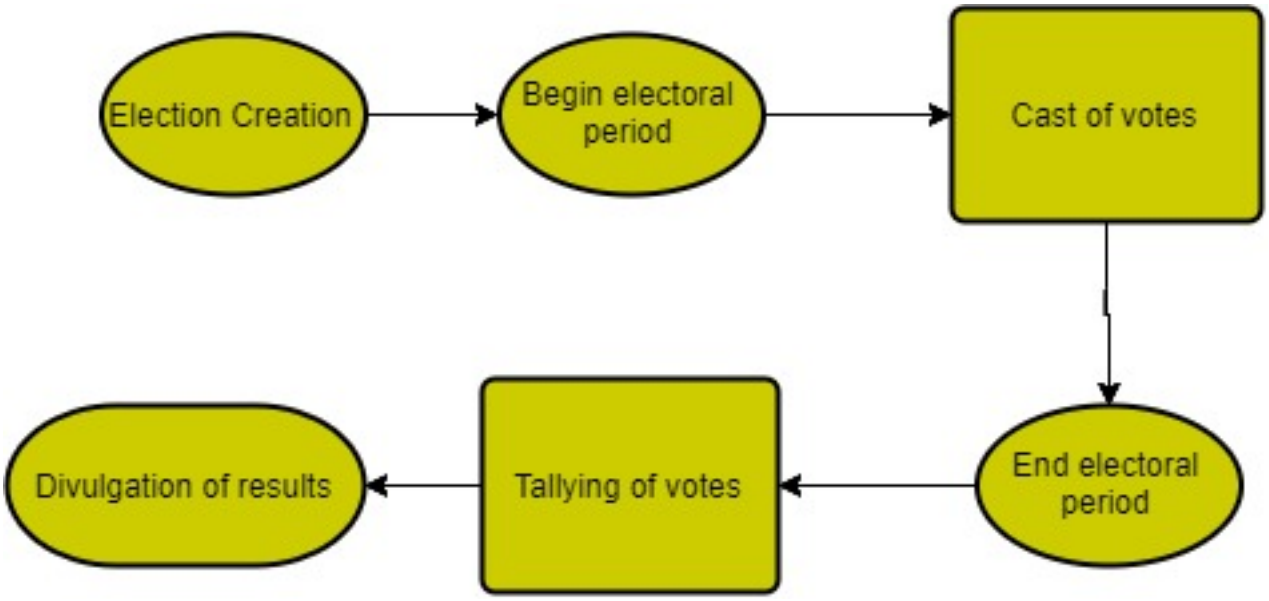

Figure 1: Example of a common electoral procedure

3.2 System requirements

In order to properly conceive the system it is required to analyze and choose how the system will function and what must be taken into account prior to its development. The system must enable users to create and participate in elections without the need to be physically present, most like paper elections where voting tables are used. As such, several requirements are necessary for the proper procedures to take place[25][35]:

- **Ballot flexibility:** It should be possible to have different types of ballots, from yes or no ballots to ballots with multiple parties;
- **Voter flexibility:** It should be possible to have different lists of eligible voters for distinct elections;
- **Spontaneity:** Running an election should require as little preparation as possible;
- **Remote participation:** It should be possible to cast a vote from wherever the voter chooses;
- **Usability:** The system should be usable by anybody;
- **Efficiency:** The system should perform its tasks as fast as possible;
- **Strict voting:** It should only be possible to cast votes on the ballot's candidates or in no candidate, resulting in a blank vote;
- **Logging:** The system should log as much information as possible without compromising individual privacy to facilitate eventual audits.

By following all these requirements the system will allow for the proper electoral procedures to take place and successfully conduct an election.

An election can be considered a "sensitive" procedure and every election differ from one another, be it in terms of candidates and/or voters. As such, the following issues have been identified and the proper procedure for each should be followed:

- **Eligibility:** Only votes from eligible voters should be accepted;
- **Uniqueness:** Only a single vote should be accepted from each voter;
- **Equality:** Every eligible voter should be allowed to see the election results, regardless if the voter cast or not a vote;
- **Vote secrecy:** It should impossible to link a voter to his or her vote;
- **Integrity:** It should be impossible to replace a casted vote with another vote;
- **Robustness:** The system should be able to perform its functions despite minor errors.

These issues, if not taken into account, will not permit a proper election to be held, so the system to be developed should take these specifications into account and not compromise the election's procedure.

3.3 Application Programming Interface

An election system requires a logical component to process data [24][27], so an application programming interface has to be developed. An application programming interface or API is an interface through which a software module is accessible. An API supplies access points, sometimes called end-points, invoking a specific API method or API class instances [37].

API's have different architecture models, each with their own advantages and disadvantages. Nowadays, one of the most used model is Representational State Transfer or REST. REST dates back to 1999 when the Internet Engineering Task Force or IETF submitted the well known Hypertext Transfer Protocol-HTTP/1.1. Roy Fielding later defined a set of principles built around the HTTP and URI standards giving birth to REST as it is known today.

The key principles of REST are:

- **Everything is a resource:** Every piece of available data on the internet has a format that can be described by a content type;
- **Each resource is identifiable by a unique identifier (URI):** Resources can be accessed via URI and should be uniquely identifiable. Also, these URIs can be human readable;
- **Use the standard HTTP methods:** HTTP protocol defines a set of actions:
  - GET
  - POST
  - PUT
  - PATCH
  - DELETE
  - HEAD
  - OPTIONS
  - TRACE
  - CONNECT
- **Resources can have multiple representations:** Resources may be represented in different forms, with the supported format and the endpoint should be able to use it;

- **Communicate statelessly:** The API should not keep its state through consecutive requests which isolates the caller from the server side changes, although part of the state can be kept in the URI.

Following these principles, REST can achieve:

- **Separation of the representation and the resource:** The caller must specify the resource representation it needs during request and the server has to handle the representation accordingly, returning the appropriate content type with a relevant HTTP status code;
- **Visibility:** Every aspect of a REST service should be self-descriptive and follow the natural HTTP language;
- **Reliability:** It should be taken into account which HTTP methods are safe and which are idempotent. The definition of safe and idempotent is:
  - **Safe:** A method is considered safe if it does not modify or cause any side effect on the state of the resource.
  - **Idempotent:** A method is considered idempotent if the response is always the same regardless of how many times it is requested.
- **Scalability:** By having an API stateless it would serve all clients in an acceptable amount of time.
- **Performance:** An API can process a single request in an acceptable amount of time.

REST APIs are commonly used in current days due to it being easy to implement and integrate into already existing infrastructures [38].

Since the secure vote module requires a logical component in order to successfully process election data, a REST API must be developed. A REST API is the logical choice due to the already existing infrastructure that can integrate the API in a relatively easy way. This API, in its endpoints, has to permit for an election to be held by following the procedure in figure 1. The API must also log all information as possible and have auditing capabilities. Certain endpoints can be used only by determined individuals, meaning the API must have user dedicated endpoints and forms to authenticate authorized users and permissions with the capability to alter them. For this system to work as intended the following permissions were defined:

- **REGULAR**
- **MANAGER**
- **AUDITOR**
- **ADMIN**

For the proper functionality of this module it is require to process and store data. Data entities must be identified and properly mapped and their properties defined [20][21][22][23][24][25][26][27][28][29][30]. The following entities were identified:

- **User**
- **Election**
- **Candidate**
- **Manager**
- **Voter**
- **Vote**
- **Log**
- **Blacklist**

The following class diagram shows the relationship between different entities present on the secure vote module API:

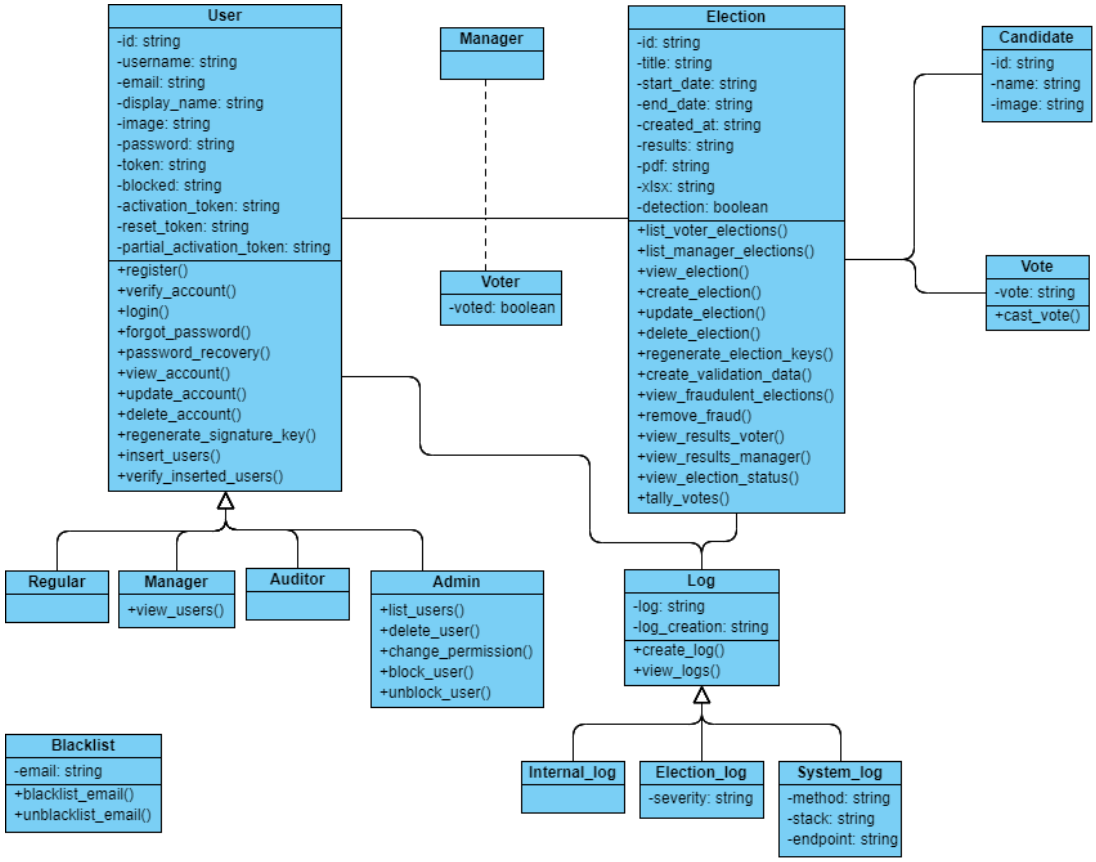

Figure 2: Class diagram for secure vote module's API.

It is possible to see in the diagram above, that all permissions are subclasses of the User class, since the different permissions are all users just with permission to access distinct methods on the module. The Election class has a relationship with the User class through two associative classes, Manager, which represents the users who are managers of said Election, and Voter, which represents the users who are eligible voters of said Election. Since an Election cannot be properly run without candidates for the voters to vote a Candidate class exists which is related to the Election class stating which candidates belong to which elections. A Vote class is also necessary to represent cast votes of a certain election. Since the module should log as much information as possible a Log class is required, with the subclasses Internal, Election and System being used to represent actions taken by the admins, actions taken during an election and possible error occurrences of the module respectively. The Blacklist class is used to represent emails that are banned from registering.

The following use case diagram depicts the operations regarding the User class:

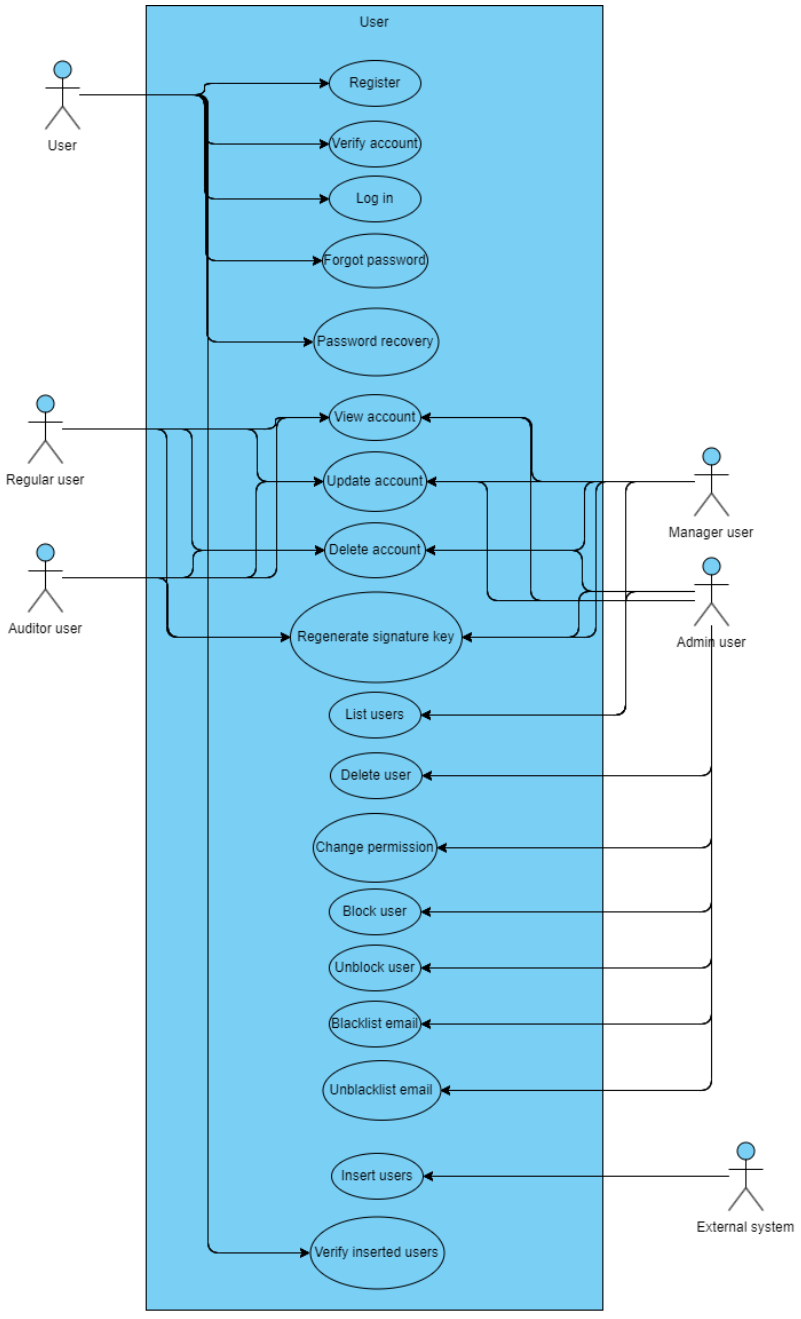

Figure 3: Use case diagram for User class operations.

The User class contains all operations regarding users and their subclasses. This admin subclass has the user management module operation. Class User also has operations to change the attribute value of their instances, including the signature key, used for vote validation after cast. The External system has the ability to create users in secure vote.

The following use case diagram shows the operations regarding the Election class:

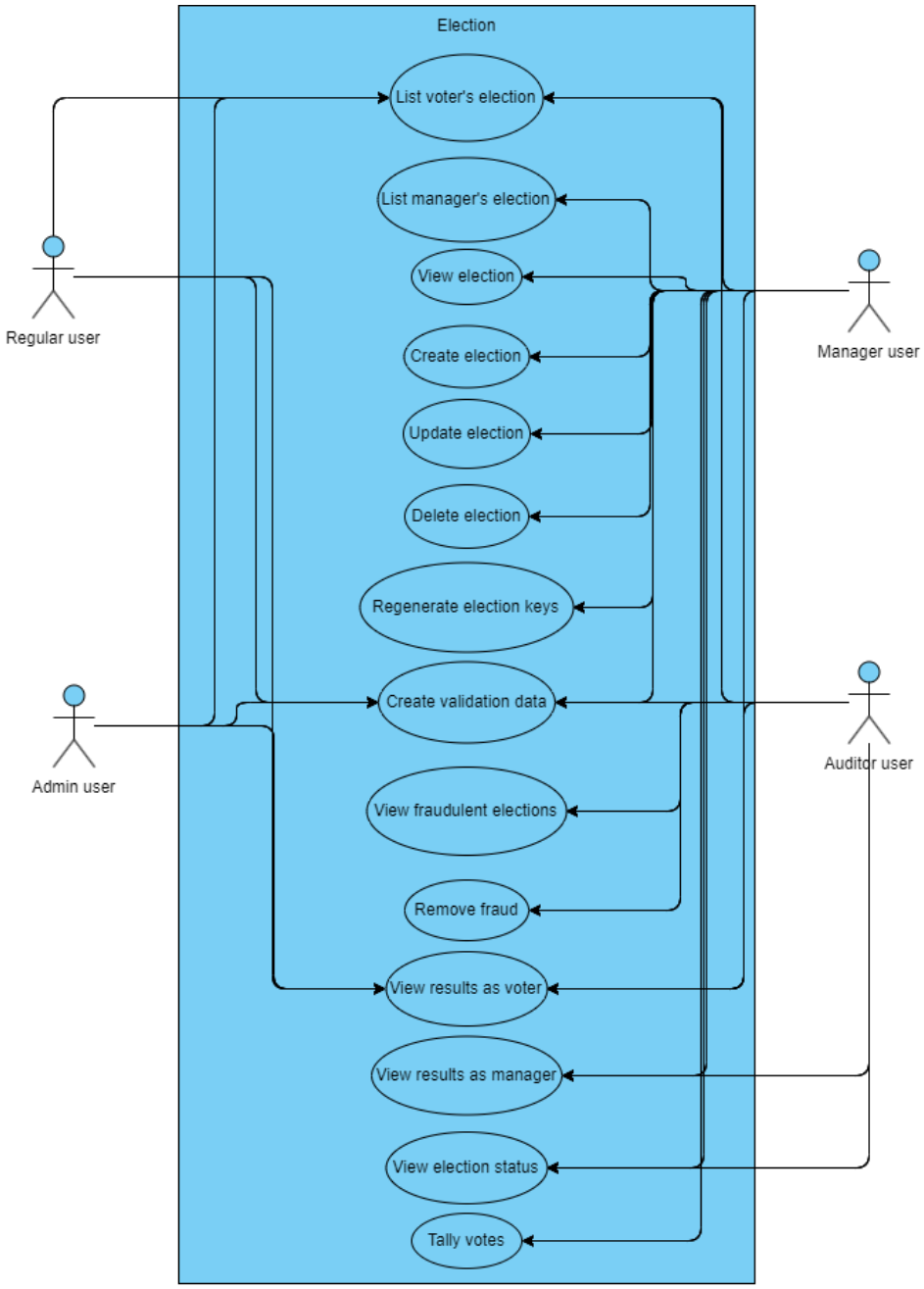

Figure 4: Use case diagram for Election class operations.

The Election class contains all operations regarding the elections. This class has the operation to creating and alter elections, including regenerating the election keys, responsible for encrypting and deciphering the cast votes. Such alterations can only be made if the election has not yet started. If an election is found to be fraudulent, during tallying, the class has an attribute detection for such a situation. Every user has permission to view the election results if they are electors but only managers can access results including the voter list, showing who did and did not vote without reveling the vote itself on what choice.

The following use case diagram shows the operations regarding the Vote class:

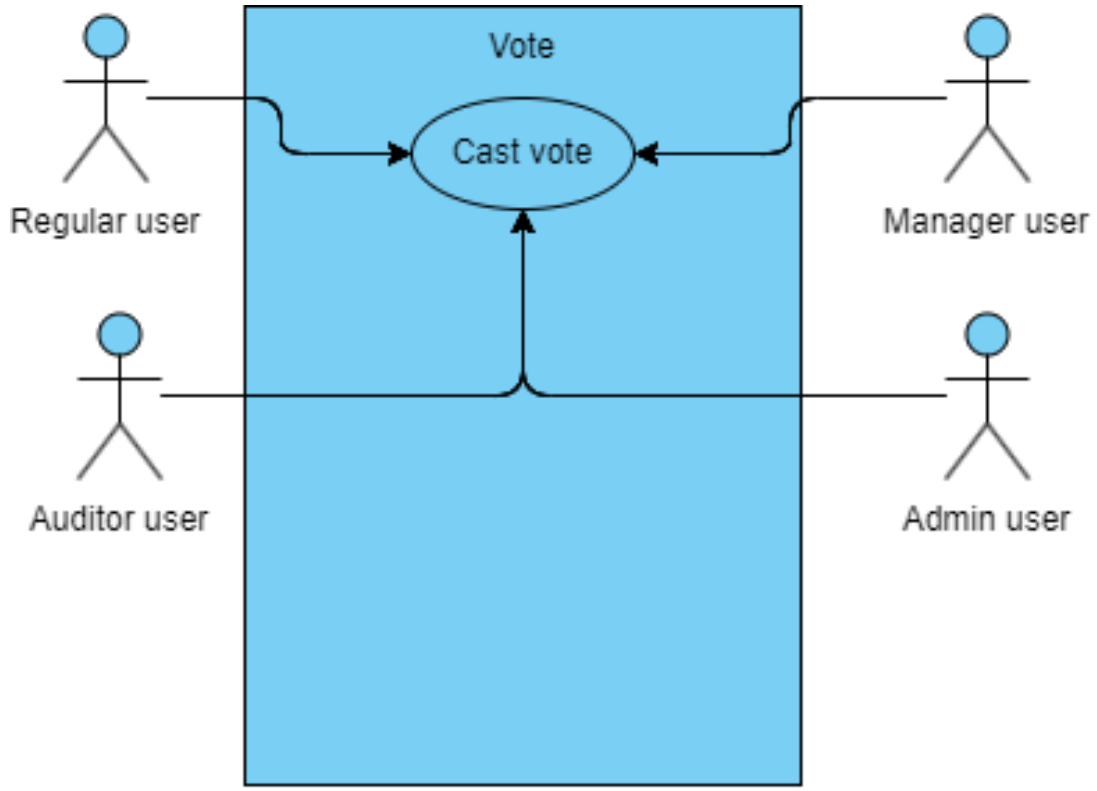

Figure 5: Use case diagram for Vote class operations.

The vote class only has an operation for casting a vote, viewed as instantiating an object of class Vote. The votes are associated to an election but not to a user respecting voter privacy. The vote tallying is performed on the Election class, counting all votes.

The following use case diagram depicts the operations regarding the Log class:

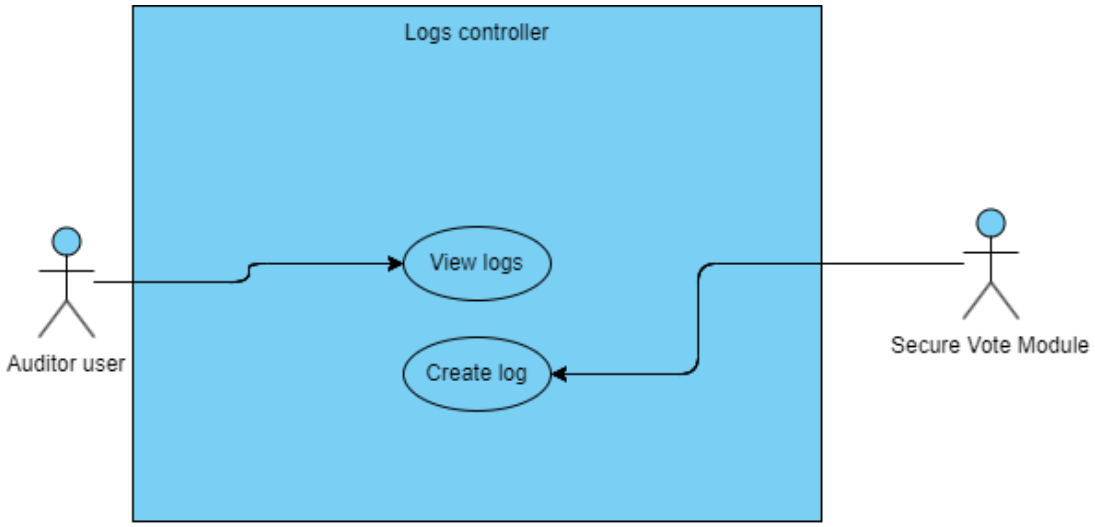

Figure 6: Use case diagram for Log class operations.

The Log class contains all operations regarding the logs and their subclasses. The module must log as much information as possible for eventual audits. Logs are created by this module and auditors can access and view the logs. Log subclasses are internal, election and system are admin actions, like creating an election or visualize possible system errors. Auditors can view the internal and the election logs but not the system logs, these just assist IT personnel and are not relevant for auditors.

The following use case diagram shows the operations of the Blacklist class:

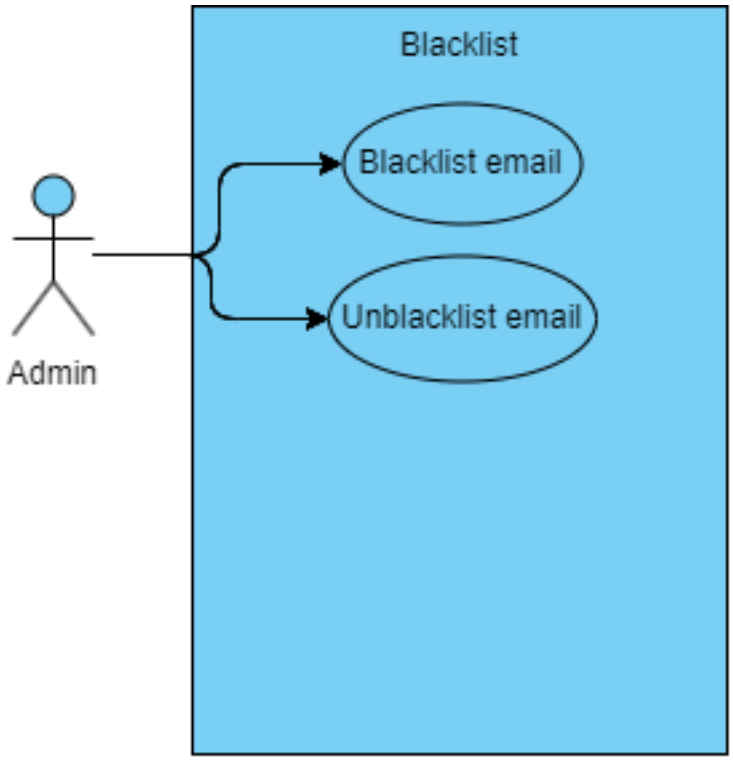

Figure 7: Use case diagram for Blacklist class operations.

This class has operations to add or remove emails from a blacklist. This blacklist blocks emails from registering an account, also used to block possible spam from successfully registering in the module.

Some classes such as Manager, Voter and Candidate have no operations since these classes are used to support other classes, because elections, to properly work, must have candidates, voters and managers.

3.4 Key Management System

The API module uses various encryption methods, therefore a Key Management System or KMS must be implemented to safeguard the system generated encryption keys. If compromised, it would be a major security issue and in no circumstance should occur [25]. A key management system permits users to manage their encryption keys. Users can fetch, create, update and delete keys [39]. The secure vote module will have an external KMS, so, if an attack occurs on the secure vote API, it will not affect the KMS and vice-versa. In this particular case, the KMS will have a API to allow the secure vote's API to fetch, create, update and delete election and user signature keys. The stored private keys will be encrypted due to their sensitive nature while the public keys do not require such encryption.

The KMS's API,like the secure vote's API will have its own entities and operations. The following entities were identified:

- **Key**
- **Log**
- **Auth**

The following class diagram shows the relationship between different entities present on the KMS's API:

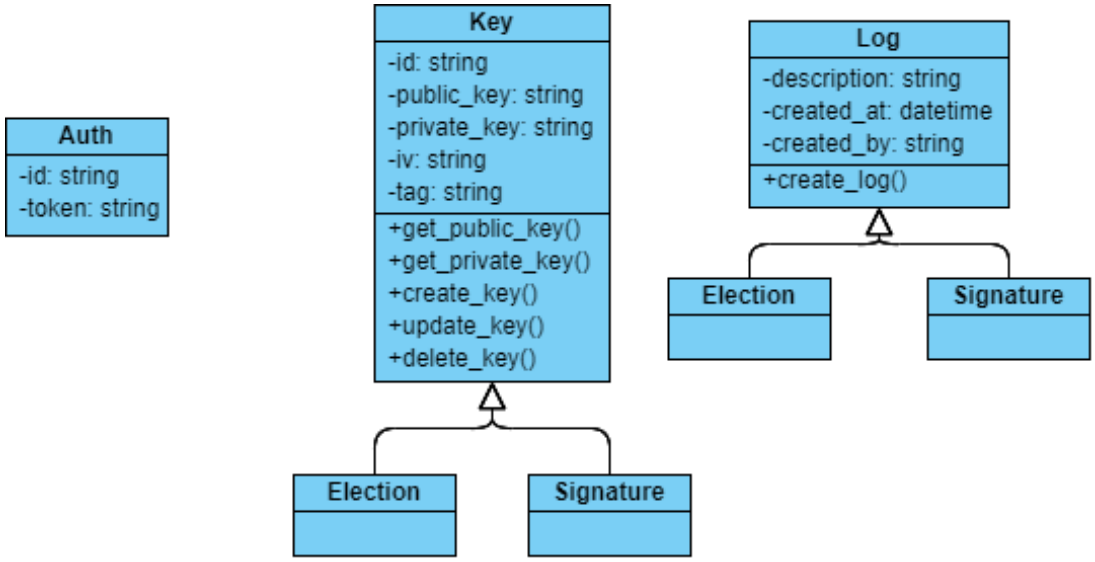

Figure 8: Class diagram for KMS's API.

The figure above shows a diagram where it is possible to see the lack of relationships between all entities because they do not require any relationship to execute their operations. This system main functionality is to safeguard signature keys and election keys and the operations for key creation, fetching, update and deletion. The system also logs all information containing an operation to create logs for key creation or deletion. Those logs and the secure vote system logs, are meant to be viewed by IT personnel and do not require a fetch operation since they be viewed directly on the database. The Auth class is used to set who has access to the KMS.

The following use case diagram shows the Key class operations:

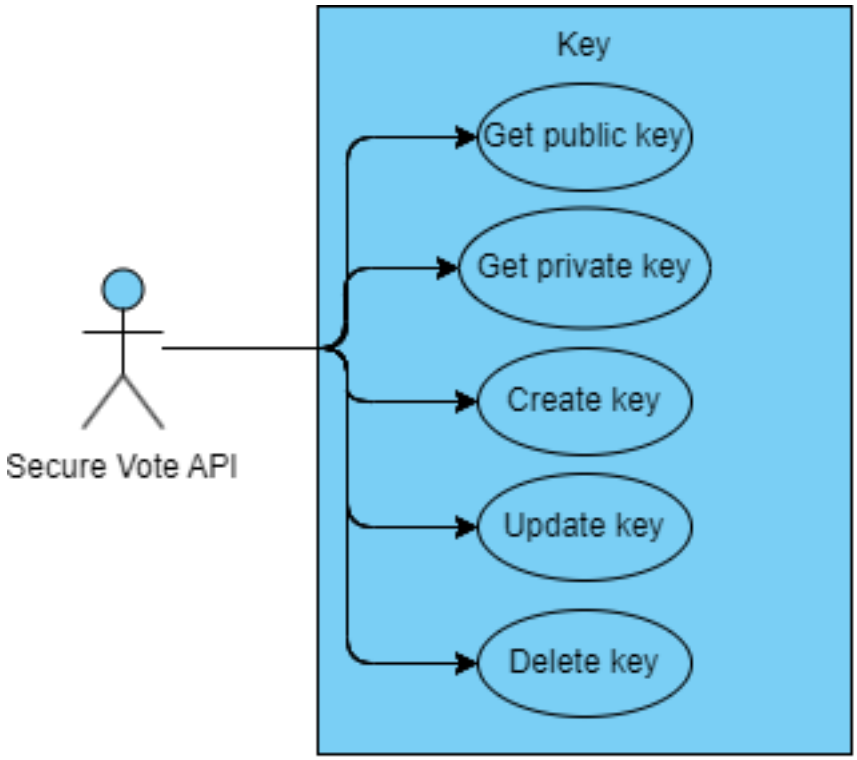

Figure 9: Use case diagram for Key class operations.

The secure vote API can use the KMS's API to fetch, create, update and delete keys, from signature, used by voters when voting to validate the cast vote, or election, used to encrypt and decipher votes.

The following use case diagram shows the operations regarding the Log class:

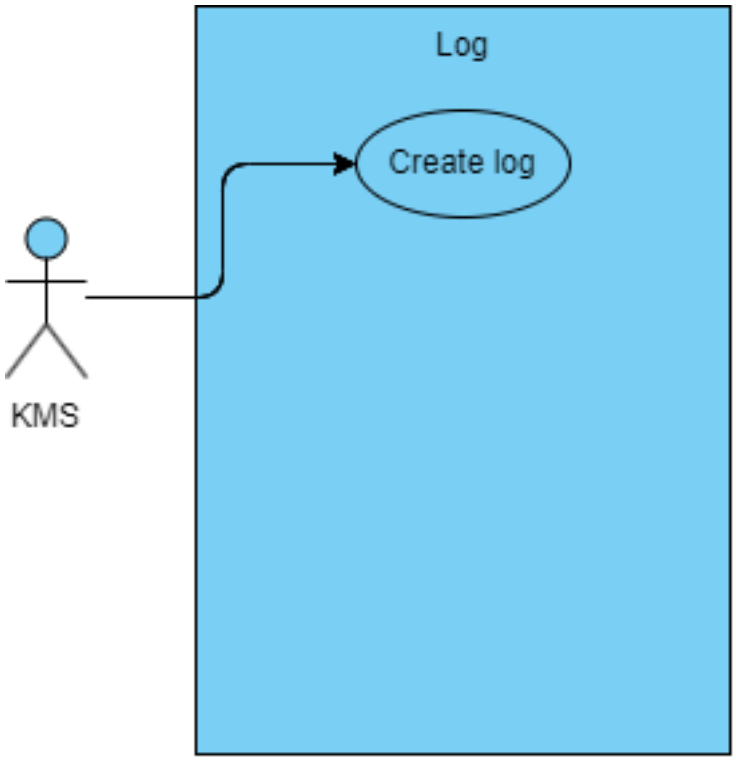

Figure 10: Use case diagram for Log class operations.

The KMS logs any action performed on it's system. These logs are used by IT personnel to check the execution of the KMS's operations, it may also be used to check eventual unauthorized accesses to the keys.

3.5 Databases

The secure vote module and the KMS will process and store data, so, the best way to store data is using a database system. A database system simplifies the managing of the data and extracting useful information tasks in a timely fashion. A database is a collection of interrelated data stored together with controlled redundancy to serve one or more applications in an optimal way. When dealing with database systems it is common to use a database management system or DBMS. A database management system is a program or group of programs that work together with the operating system to create, store, retrieve, control and manage data. It acts as an interface between the application program and the stored data in the database [40]. DBMSs vary on how data is stored and fetched, the most commonly used are Relational Database Management Systems or RDBMS.

A RDMS uses the concept of storing data in rows inside tables, where each row represents an entity of the real world with table columns being properties of these entities. In a relational database a set of tables exists, where each table has a predefined name and a set of column names or attributes. Each attribute has a predefined domain and only this domain values are accepted in such attribute column. A table is then filled row-wise with the values corresponding to the entity state and each row is a tuple of values. Each table corresponds to the mathematical notion of a relation where the set of tuples in a relation are a subset of the attribute domains cartesian product. The definition of the attribute names for the relation is called a relation schema, hence the set of the relation schemas of all relations in a database is called a database schema. Each relation schema can have intrarelational constraints and the database schema can have interrelational constraints. These constraints describe dependencies between the stored data with the intrarelational constraints inside a single table, and interrelational constraints describing dependencies between different tables. The constraints can be used to verify if the inserted data is semantically correct. Relationships are also translated into a relation schema. In order to properly map the values from the entities connected by a relationship, the relationship must also contain key attributes of the entities participating in the relationship. These attributes are called foreign keys.

To properly retrieve and insert data into a database it is required a way to communicate. The standardized language to communicate with RDBMSs is called Structured Query Language, SQL for short. Using SQL it is possible to define, manipulate and query data stored in the database.

Several users or processes should be able to work with database system at the same time. This is called concurrency management, aiming to make concurrent data accesses possible. To make this possible RDBMSs usually support transactions and provide a component to control concurrent transactions. A transaction can be defined as a sequence of read and write operations on a database, that must be treated as a whole and must not be interrupted.

Most RDBMS manages transactions according to the following properties:

- **Atomicity:** All transaction operations are executed or none at all;
- **Consistency:** After the transaction, all values in the database are correct;
- **Isolation:** Different user transactions do not interfere with each other;
- **Durability:** All transaction results persist in the database even if a system crash occurs;

Database systems who adhere to these properties are called ACID-compliant[41].

Although RDBMSs have their strengths, in some cases they are not ideal candidates for data storage depending on the application type. In some situations, relational data representation is inadequate, because due to normalization they will complicate how the data is retrieved and inserted, causing some entities to be unnecessarily stored in different tables, complicating data manipulation. Semantic overloading is another issue that occurs in some cases where both entities and relationships are contained in a relation schema, and there is no way to express if the relationship between entities have a different nature than the entities themselves. RDBMSs also have weak recursion and homogeneity support, requiring both horizontal and vertical homogeneity for the tables data. Horizontal homogeneity, where all rows have a defined fixed uniform format in the columns. Vertical homogeneity is where all values in one column belong to the same predefined attribute domain. Other issues like the restricted data types and lower throughput are also present.

Over the years, data processing experienced several changes, bringing with them new challenges for database management such as:

- **Complexity:** Data can be organized in complex structures;
- **Schema independence:** Documents can be processed without a given schema definition;
- **Sparseness:** If a schema exists, it may happen that a lot of data items are not available. If these are missing, data are represented as null values and occupy storage space unnecessarily;
- **Self-descriptiveness:** Metadata describes value use and semantics, attached to individual values, can be interpreted directly without acquiring from other source metadata information;
- **Variability:** Data changes constantly so the database system must be able to handle these changes;
- **Scalability:** Data can be distributed through interconnected servers. The database user does not need to know from where data originates and must be able to interact with the database system without knowing the data distribution. It must also support flexible horizontal scaling with servers leaving or entering the network on demand;
- **Volume:** A database system must provide high input and output (reads and writes) throughput to provide system availability and responsiveness.

As a reaction to these challenges, Non-relational databases have been developed. Non-relational databases, also referred as NoSQL databases. NoSQL database systems offers query languages and access methods other than SQL, and recently, database systems that use a different data model from relational tables, support query languages and/or access methods different from SQL, can handle schema evolution or schemaless data, support data distribution on a server network by default and not strictly following ACID properties[42].

Secure vote API and the KMS API, both require data storage and database systems to successfully operate therefore data schemas must be set.

The following diagrams describe the schemas used to store the secure vote's data:

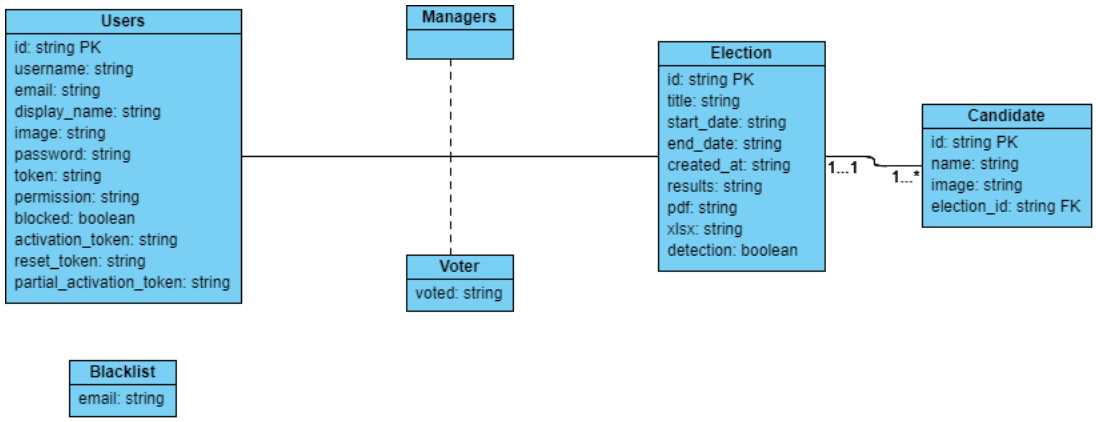

Figure 11: Diagram for the relational secure vote data schema.

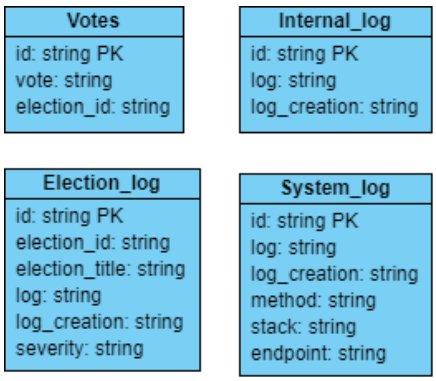

Figure 12: Diagram for the non-relational secure vote data schema.

Analyzing the above diagrams, compared with diagram 2, it is possible to see that the diagram has been divided into two, one relational and another non-relational. One of the reasons for this division is the decentralization of data, if a database is compromised not all data will be compromised, only the accessed database data. The main reason is because votes and logs have operations that must not fail execution because vote casting and logging are this module's key operations. Excluding possible API crashes, a downed database will cause execution operations to fail, so to mitigate it, was decided the Cassandra database system usage.

Cassandra is a NoSQL distributed wide-column database system developed by Apache with the main focus on high availability without any single point of failure, making it a strong database system to store secure vote data and logs. Cassandra however is not ACID-complaint due to the lack of transactions but operations that will be performed on it by the module do not require any transactions whatsoever because they are single queries, not sequenced queries. Cassandra allows prepared statements preventing NoSQL injection attacks and, therefore, should be used.

As for schema 11 any RDBMS can be used for storage. PostgreSQL is the database system that should be used to store schema data because it is ACID-complaint and allows stored procedures automatic updates preventing any SQL injection attempts. All RDBMSs support stored procedures, but PostgreSQL automatically updates them, which is not a feature of all RDBMSs. PostgreSQL is ACID-complaint, allowing transactions for data to be stored, for example, when an election is created, the candidates, voters and managers are also registered in the database. This registrations should be managed using a transaction when dealing with data manipulation on multiple tables and if one of these queries fails, the data will not be consistent and data is stored improperly. By using a transaction, all queries either execute or fail, solving the problem completely.

Comparing schema 11 with the diagram 2 it is possible to see that the first has some attributes not present on the second, due to the permissions that having to be stored together with the users to identify the particular user's permission. Several tokens are also stored, the reset token is used when a user wants to recover the password and the activation tokens are used to verify if the submitted email is actually the user's email. This is accomplished by sending an email with the activation URI, containing the token, to the user previously submitted email. This email can only be seen if the submitted email actually belongs to the user. The partial activation token is used when an external system creates users in secure vote, and then user's must verify the account before accessing the module.

The following diagram describes the schema used to store the KMS's data:

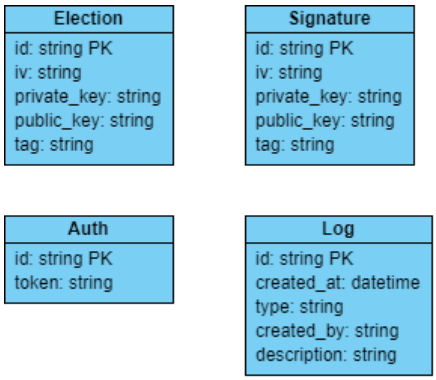

Figure 13: KMS Diagram data schema.

Analyzing the above diagram compared with diagram 8 it is possible to see that there is a table for each key type. The objective is to improve data organization, because using a table with an attribute "type" could be confusing. The log table has a attribute type to identify if the log applies to a signature or an election key, the reason is due to logs not being fetched by the KMS while keys are. Any DBMS, NoSQL or RDBMS, can be used for storage using this schema but MongoDB is the system that should be used, because it is schemaless, allowing the data to change the schema without having to alter the database itself, permitting the keys to be stored regardless of what encryption type is used. Due to the lack of relationships, a NoSQL database system should be used and not a RDBMS because, if there is no use for relationships, it can take advantage of the increased throughput provided by a NoSQL database system.

3.6 Deployment

After all components of the module are implemented it is required to be deployed [27][30]. The following deployment diagram describes an optimal way to deploy this system:

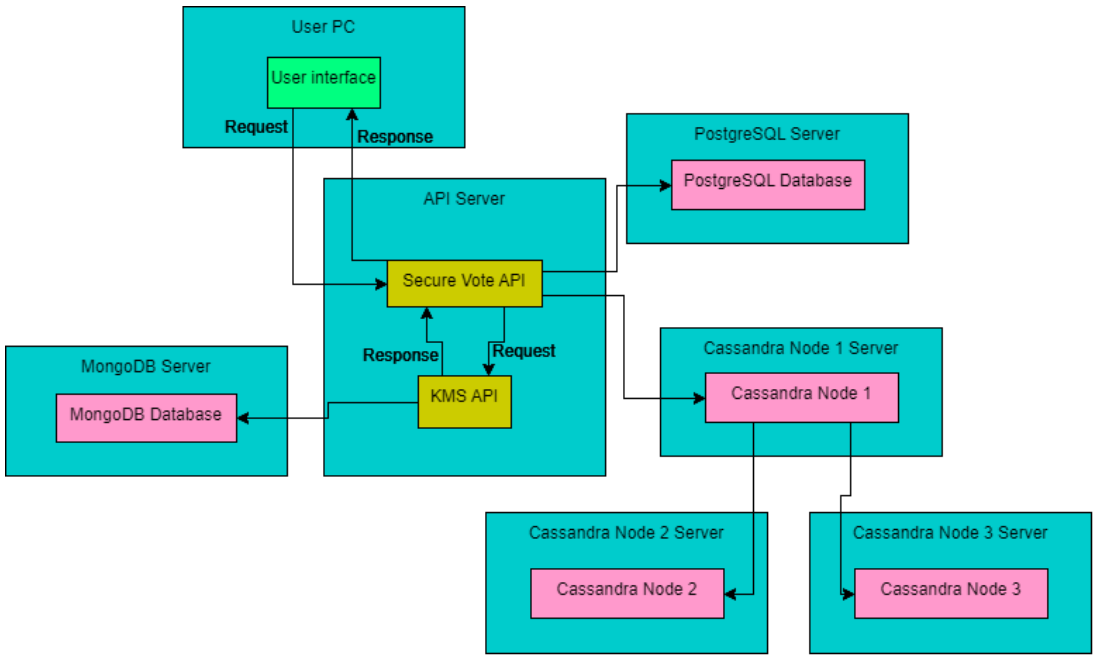

Figure 14: Diagram for an optimal system deployment.

The above diagram shows the server usage for the APIs and each used database system. Taking advantage of the decentralized nature of Cassandra, every node server usage is optimal in case a server crashes or a DoS/DDoS attack occurs. Continuing on decentralization, MongoDB and PostgreSQL should have their own servers to mitigate possible data compromises, isolating such attempts to specific servers. Although this approach is optimal it may not be feasible due to the amount of servers required and the higher budget required. Following this approach it is possible to use fewer servers or even a single server to implement the system has described in 14, without any apparent performance loss but it will hinder the system overall security and make possible data compromises to be more severe.

CHAPTER 4

System Interface

The secure vote API supplies a logical component required to held an election but a simple and easy interface for users to use this module is required. The following images show the secure vote interface main menus (for more detailed information regarding the usage of secure vote interface please consult secure vote user manual present at the end of this document):

Disclaimer: All interface images shown on this document and the user manual contain real-life people images obtained from **Unsplash** (URI: https://unsplash.com). License: https://unsplash.com/license.

4.1 Register

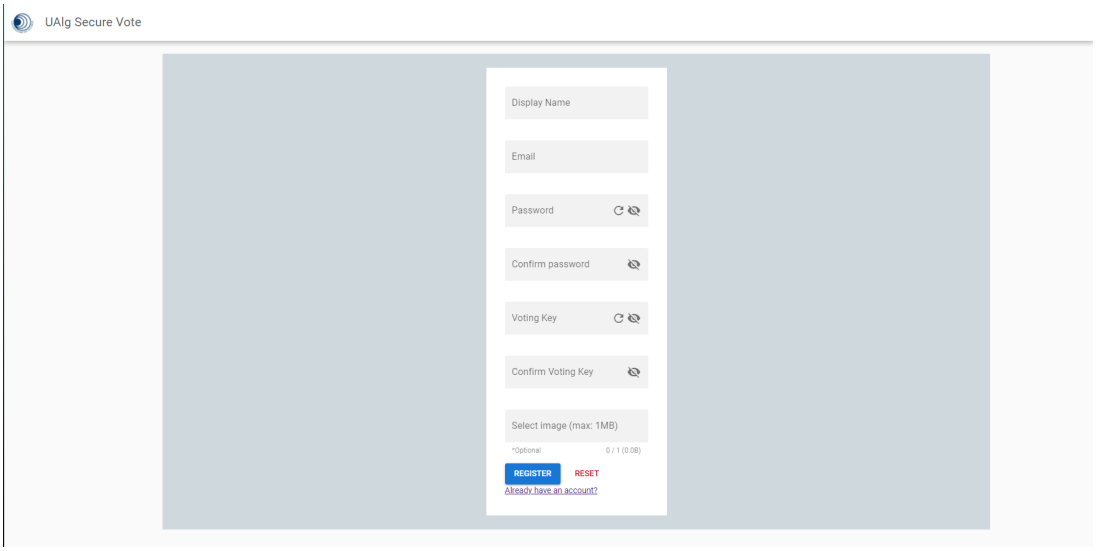

Figure 15: Menu for user registration.

This menu allows users to register an account in the secure vote module. To register the following fields must be filled:

- **Display Name**: the name the user wants to be displayed in the system, usually the user's name.
- **Email**: the email the user wants to use to register in the system.
- **Password**: the password the user wants to use to access the system. This password must be at least 8 characters long and have upper and lower case characters, numbers and special characters. The ↻ symbol generates and fills the field with a strong password.
- **Confirm password**: the password entered on the Password field must be reentered in this field to be confirmed.
- **Vote key**: the vote key the user wants to use to confirm its vote. Whenever the user casts a vote the user must enter this key in order to confirm the vote. This key must be at least 12 characters long and have upper and lower case characters, numbers and special characters. The ↻ symbol generates and fills the field with a strong vote key.
- **Confirm vote key**: the vote key entered on the Vote key field must be reentered in this field to be confirmed.

- **Select image**: upload an image to be the user's avatar in the system. This field is optional which means it can be empty in case the user doesn't want an avatar. Only .jpg, .png and .svg files of up to 1MB size are accepted.

After all required fields are filled the Register button registers the user's account and redirects the user to the Login page. If any error is present the user is not registered and a message is shown stating the error. The password must be different from the vote key. The Reset button clears all fields and clicking on "Already have an account?" redirects the user to the Login page.

4.2 Login

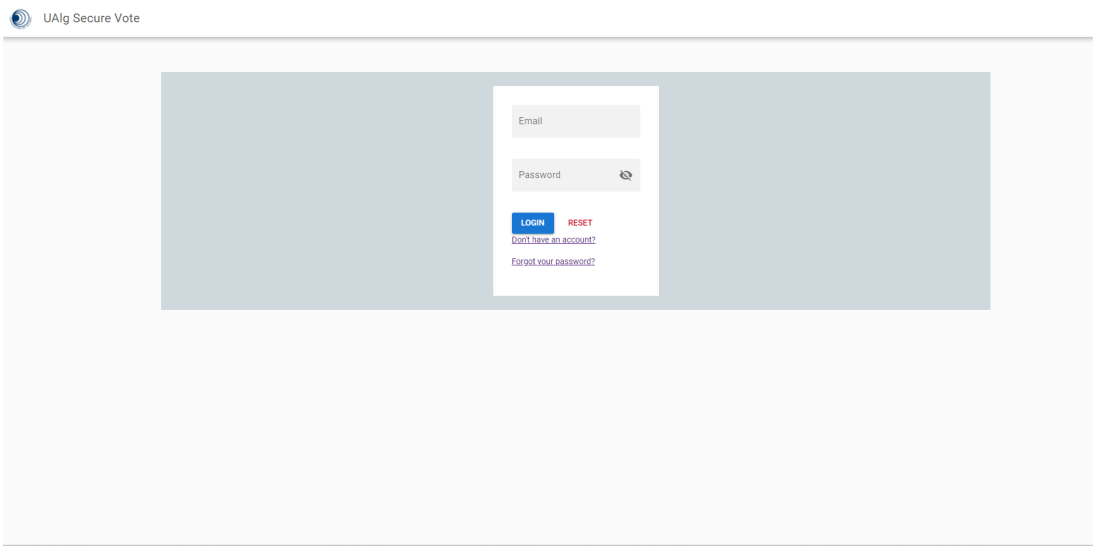

Figure 16: Menu for user authentication.

This menu allows users to authenticate in the secure vote module and access its functionalities. To authenticate the user must fill the following fields:

- **Email**: the email the user used to register.
- **Password**: the password the user submitted during register.

After filling all the fields and if all are correct the user is successfully authenticated and redirected to the Elections page. Otherwise a message is shown stating that the email does not exist and/or the password is wrong. The Reset button clears all fields and clicking on "Don't have an account?" redirects the user to the Register page. Clicking on "Forgot your password?" redirects the user to the Forgot Password which allows the user to recover its password if its forgotten.

4.3 Forgot Password

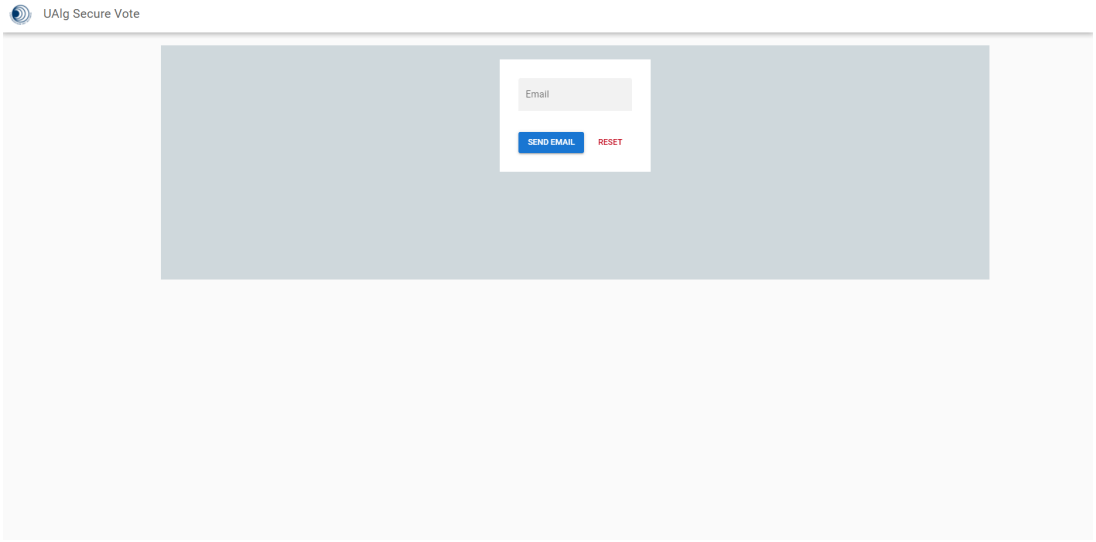

Figure 17: Menu for password recovery.

This menu allows users to reset their password. To begin the recovery process the user must fill the following field:

- **Email**: the email the user uses to authenticate.

After filling the field and clicking on the Send email button the user will receive an email with details to successfully recover its password. If the inserted email does not match any registered email a message that no such email exists will be presented. The Reset button clears the field.

4.4 Logout

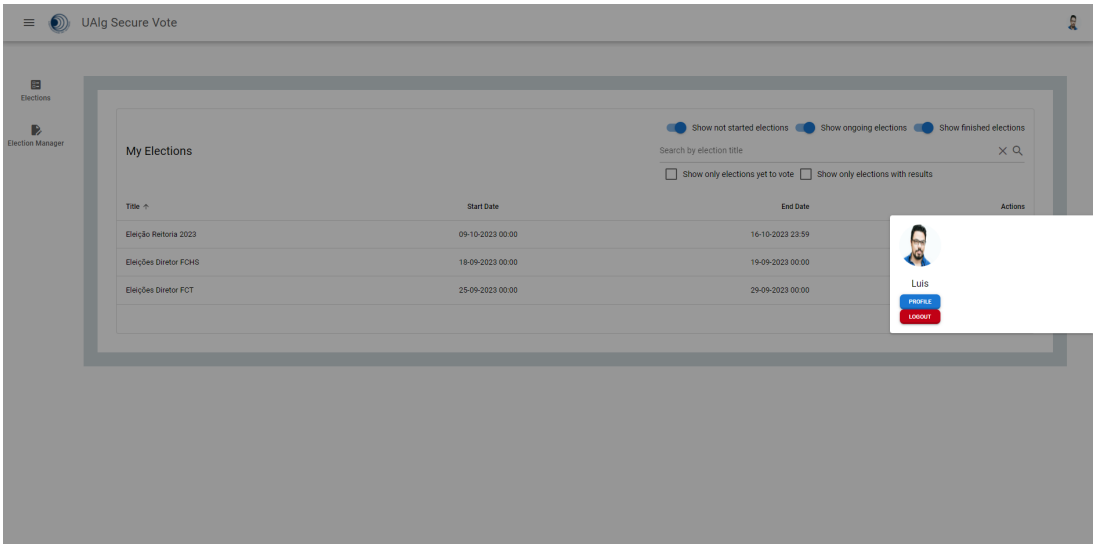

Figure 18: Menu for logout.

This menu allows authenticated users to logout from the system. Clicking on the avatar on the top right corner present in every page opens a side menu with two buttons.

Clicking on the Profile button the user is redirected to his Profile page.

Clicking on the Logout button the user is redirected to the Login page and will have to authenticate again to access the module again.

4.5 Elections

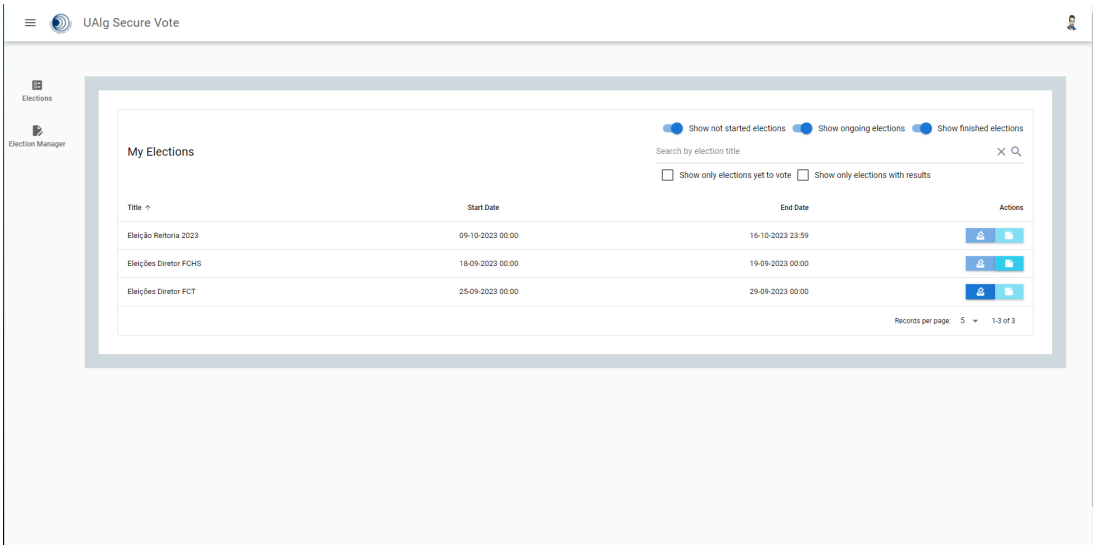

Figure 19: Menu for elections.

This menu allows authenticated users to see the elections where they are eligible voters. Users can vote, if the election is ongoing and have not previously voted, and visualize results after the election ended. This menu also supplies filters to facilitate election finding, already ended or to be held.

Clicking on the button will open the selected election ballot. This button is only available if the election is ongoing and the user hasn't yet voted.

Clicking on button will open the selected election results, if the votes have already been tallied

4.6 Ballot

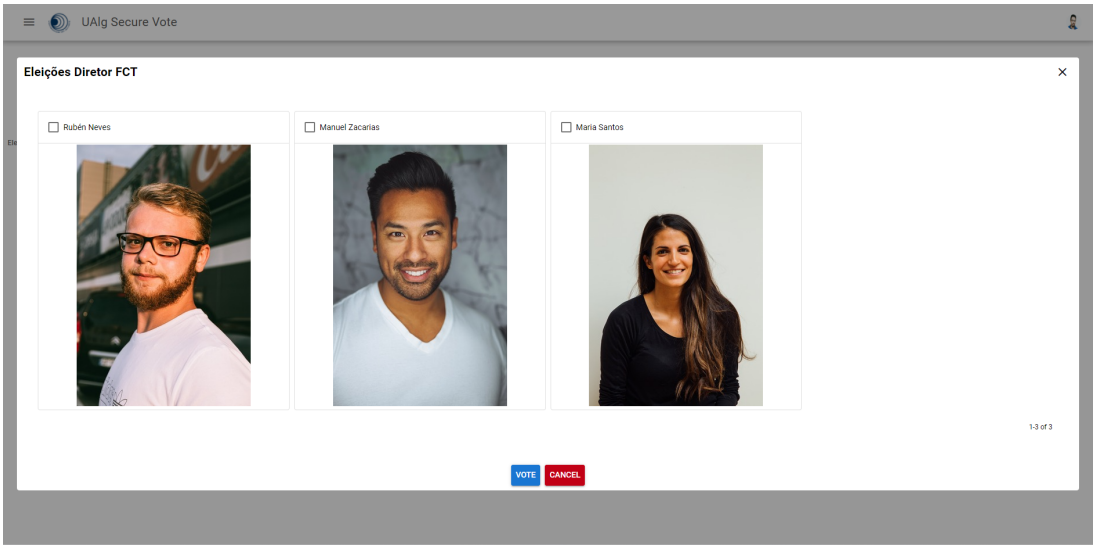

Figure 20: Menu for ballot.

This menu allows authenticated and eligible voters to vote on the election. This menu is only accessible if the voter is indeed eligible. To cast a vote the user must select the candidate it wishes to vote and click on Vote. The user is then prompted to insert its vote key to confirm the vote. If the vote key is correct the vote is casted and the user receives an email confirming the vote on the election. If the key is incorrect then a message stating the key is not correct is shown. The Cancel button closes the ballot without submitting a vote.

4.7 Election Results

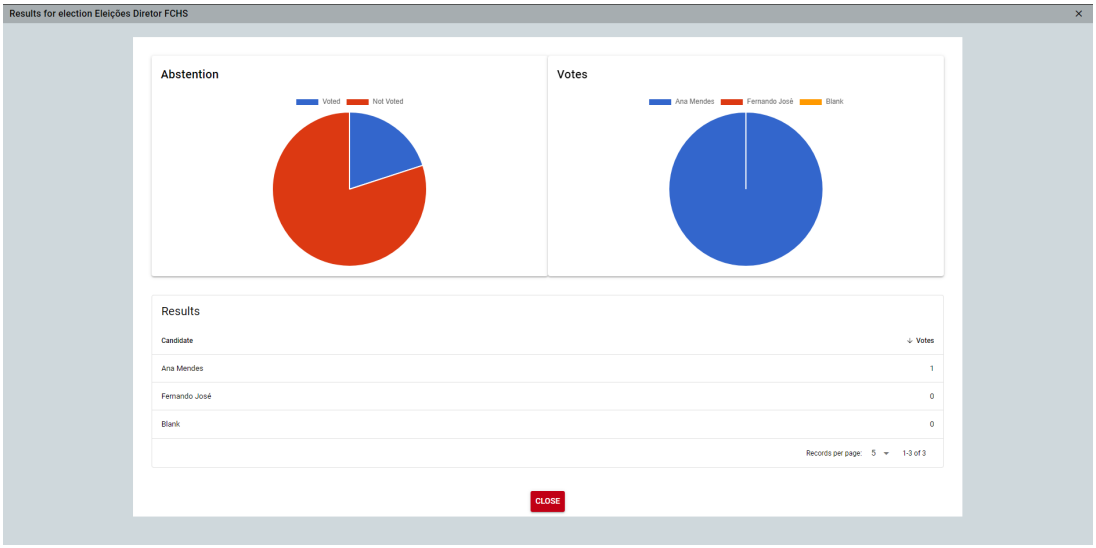

Figure 21: Menu for results.

This menu allows authenticated and eligible voters to check the election results. This menu is accessible only by eligible voters regardless of participation on the election.

4.8 Election Manager

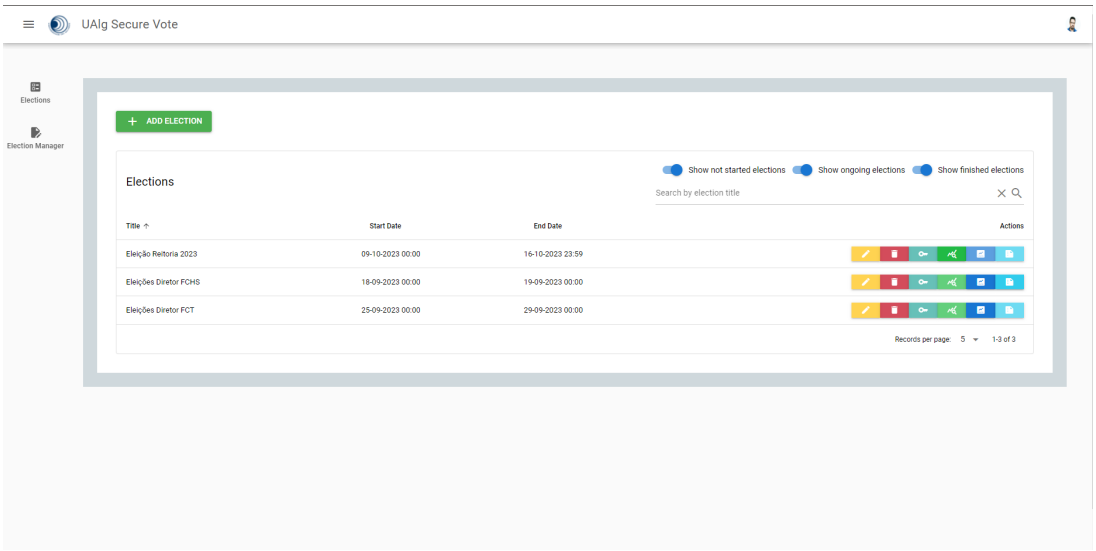

Figure 22: Menu for election manager.

This menu allows authenticated managers to create, edit and delete elections. It also allows managers to regenerate the election key. Managers can also check the managed ongoing election status. After the ballots close, managers can tally votes and view their managed election results. Filters are supplied in this menu to facilitate election search.

Clicking on the + Add Election button on the top left corner of the menu opens the Add Election window and allows managers to create an election.

Clicking on the ✎ button will open the Edit Election window and allows managers to edit managed elections.

Clicking on the 🗑 button will open the Delete Election window and allows managers to delete managed elections.

Clicking on the 🔑 button will open the Regenerate Election Key window and allows managers to regenerate the election key for the managed election.

Clicking on the button will open the Election Status windows and allows managers to view status of an ongoing managed election.

Clicking on the button will open the Tally Election Votes window and allows managers to tally the election votes.

Clicking on the button will open the Manager Election Results and allows managers to view the managed election results.

The following rules describe when these buttons are enabled:

- It is only possible to edit, delete and regenerate election keys if the election has not yet started.
- It is only possible to view the election status of ongoing elections.
- It is only possible to tally votes of elections that have ended.
- It is only possible to view the results of an election if the votes have been tallied.

Otherwise these buttons are disabled.

4.9 Add Election

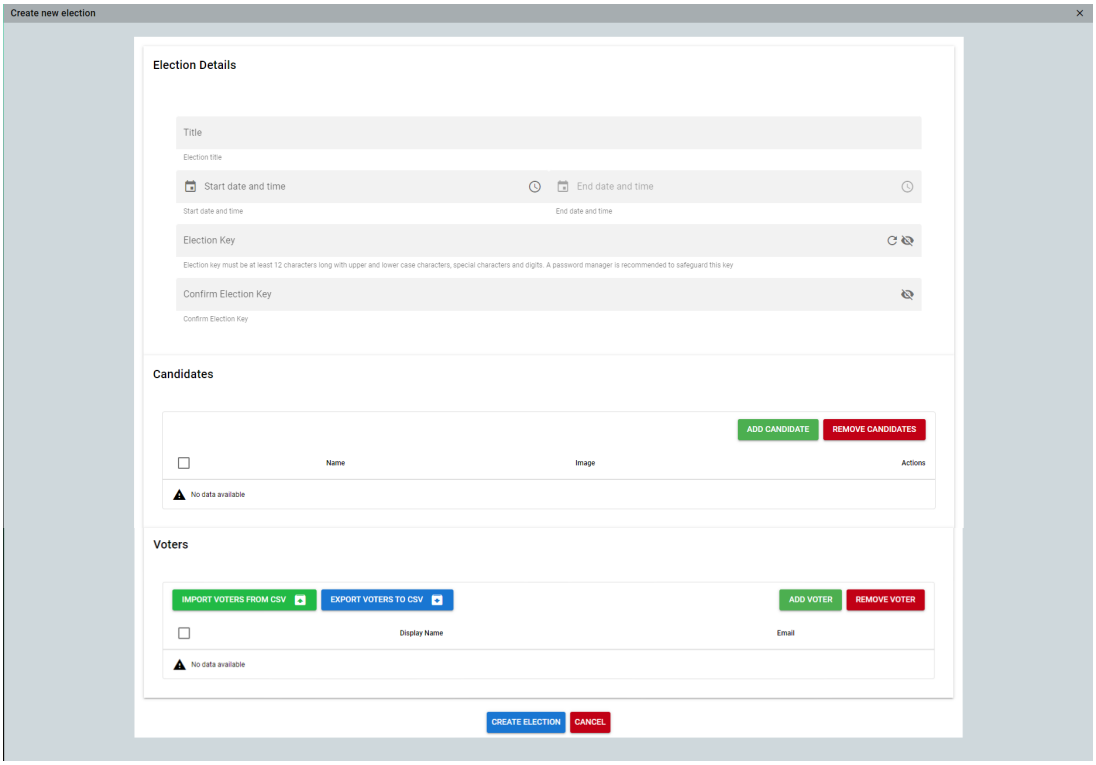

Figure 23: Menu for election creation.

This menu allows authenticated managers to create elections. To create an election, managers must fill the following fields:

- **Title**: the title of the election.
- **Start date and time**: the date and time when the election starts i.e. the election accepts votes.
- **End date and time**: the date and time when the election ends i.e. the election stops accepting votes. This date and time must always be after the start date and time.
- **Election key**: the key for this election. This key must be at least 12 characters long and have upper and lower case characters, numbers and special characters. The ↻ symbol generates and fills the field with a strong key.
- **Confirm Election key**: the election key entered on the Election key field must be reentered in this field to be confirmed.

The Election key should be safely stored. Although this key can be changed before the election starts if its lost after the election starts the votes cannot be tallied and as such the results cannot be viewed.
The Candidates table presents the election's candidates and has two actions: Add candidate and Remove candidate.
Clicking the Add candidate button opens the create candidate modal. This modal requires the following in order to create a candidate:

- **Name**: the name of the candidate.
- **Image**: the image of the candidate, this field is optional.

These candidates are then shown on the Ballot of said election.
To remove candidates the manager has to select the candidates it wishes to remove by checking their checkbox on the table and then clicking the Remove candidate button. The selected candidates will then be removed.
The Voters table presents the election's voters and has the following options:

- **Import voters from CSV.**
- **Export voters to CSV.**
- **Add voters.**
- **Remove voters.**

Clicking on the Import voters from CSV button a manager can import a .csv file and the voters on the file which are verified and not blocked users on the module will be imported to the voters table.
Clicking on the Export voters to CSV button a manager can create a .csv file with the voters on the table, this file can then be used with Import voters from CSV.
Clicking on the Add voters button will open a modal. This modal contains a table with all verified and not blocked users on the module which can then be selected and saved as election voters.
Just as with Candidates a manager can select the voters it wishes to remove from the voters table by checking the checkbox of the voters and then click on the Remove voters button to remove the selected voters from the table.

Importing the voters from a .csv file and entering the voters manually i.e. using Add voters can both be used to select the election voters.
When using Import voters from CSV a manager should use Export voters to CSV generated .csv file but a manager can use any other .csv file if this file contains an "Email" property (case insensitive) and the values under this property are emails that belong to verified and not blocked users on the module, these users will be imported as election voters while the other emails are discarded.

After the details, candidates and voters are inserted clicking on the Create election button will create the election and the manager will be redirected to the Election Manager page.
If the manager does not insert the election details and candidates a message will be shown stating that the manager did not fill all required information to create an election.
Clicking on the Cancel button the manager will be redirected to the Election Manager page and all inserted data is deleted.

4.10 Edit Election

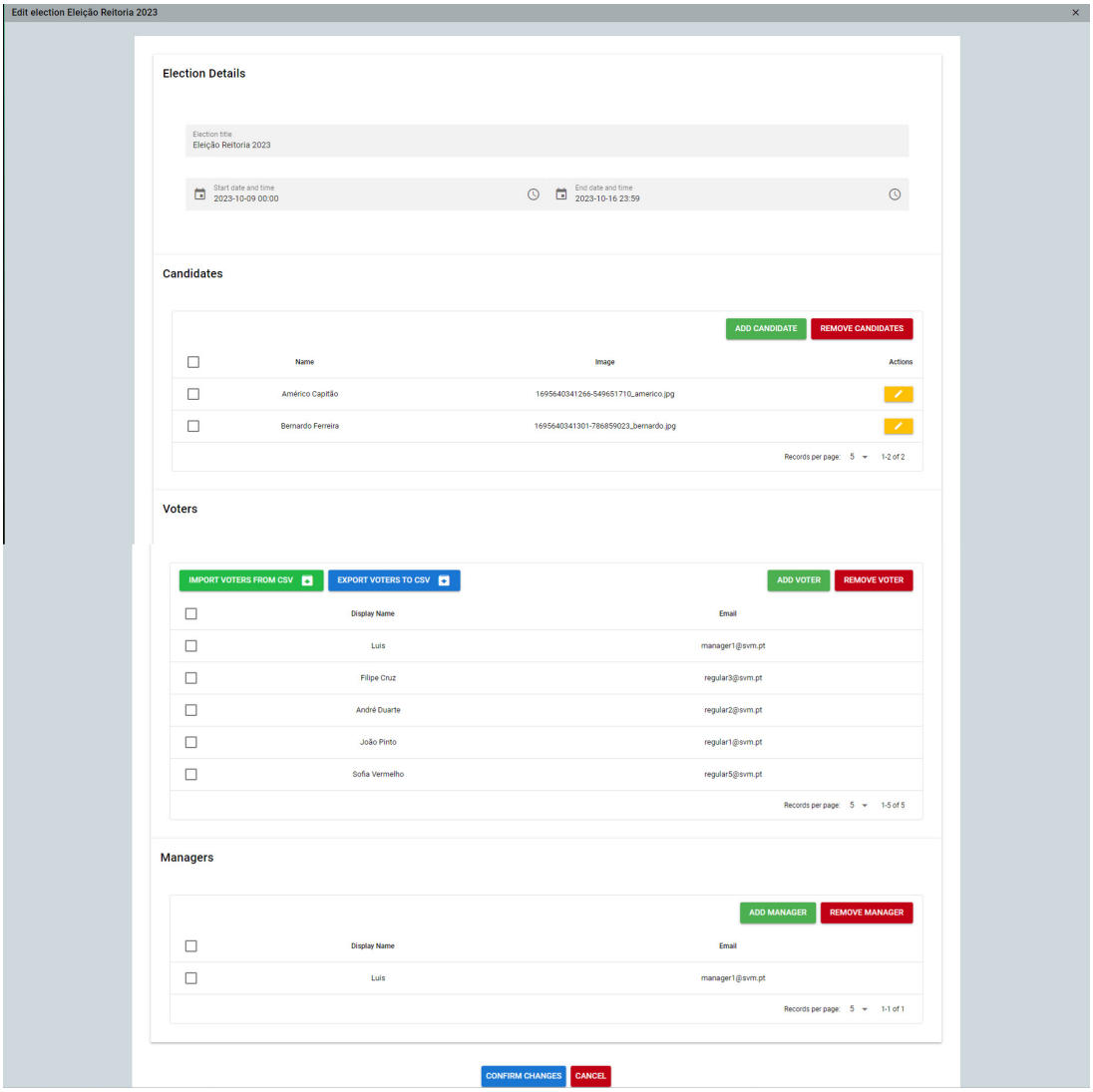

Figure 24: Menu for editing elections.

This menu allows authenticated managers to edit a managed election. All data regarding the election will be presented with the exception of the Election key.

The manager can change the election details, candidates and voters. All requirements applied on the Add Election window also apply on this window.

In this window the manager can also add and remove managers from the election. If all managers are remove a message is shown stating that the requirements are not fulfilled since an election must always have at least one manager.

After performing all the changes clicking on the Confirm changes button saves the changes. The manager is then redirected to Election Manager page.

Clicking on the Cancel button all changes are not saved and the manager is redirected to Election Manager page.

Added managers will not have the Election key since the module does not store this key, this key must be shared manually between managers with the utmost care as to not compromise this key.

4.11 Delete Election

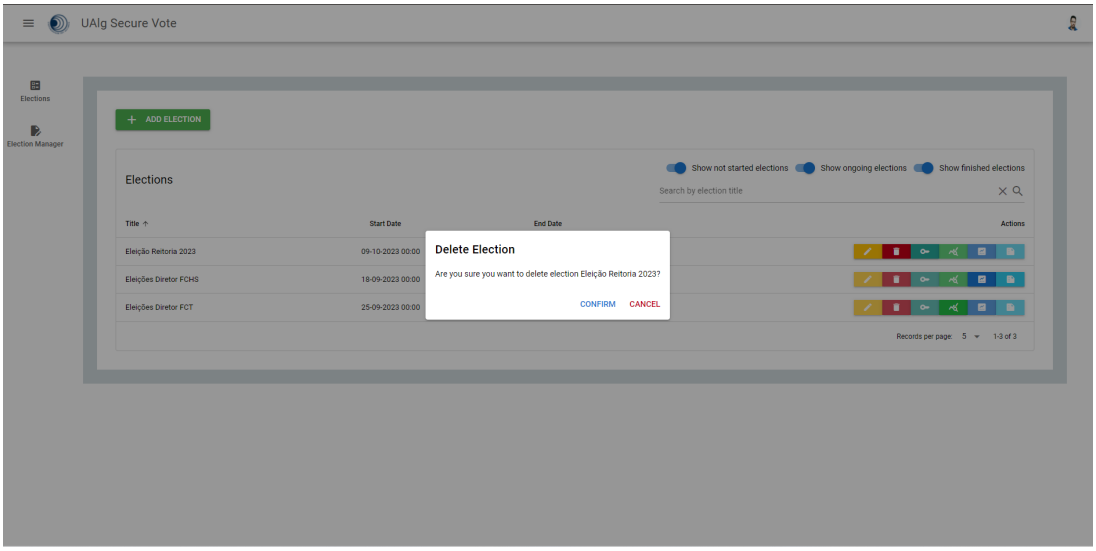

Figure 25: Menu for deleting elections.

This menu allows authenticated managers to delete a managed election. Clicking on the Confirm button the election will be deleted and the manager is redirected to Election Manager page. Clicking on the Cancel button the manager cancels the deletion and is redirected to Election Manager page.

4.12 Regenerate Election Key

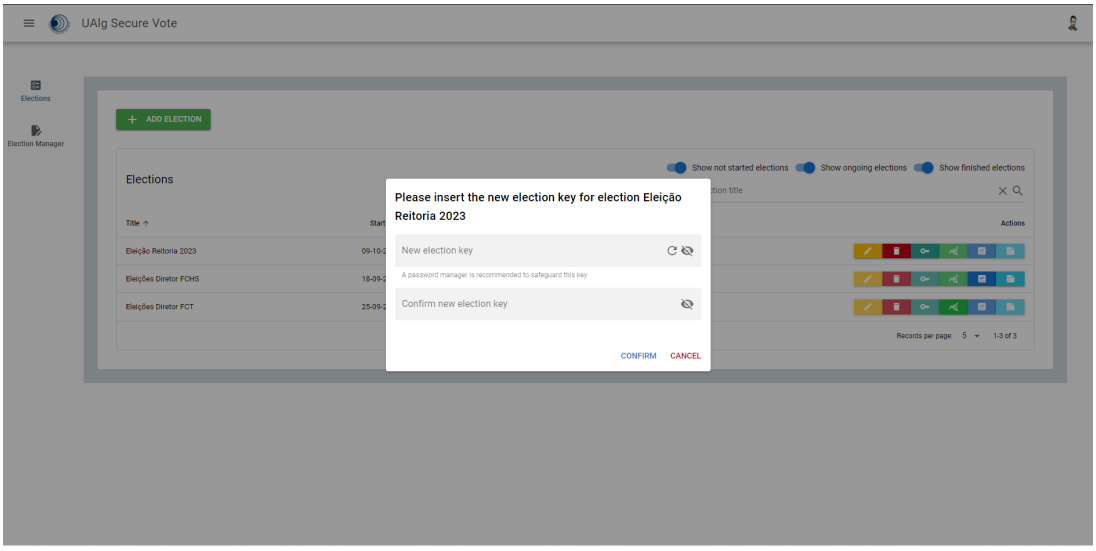

Figure 26: Menu for regenerating an election key.

This menu allows authenticated managers to regenerate a managed election key. To regenerate an election key a manager must fill the following fields:

- **New election key**: the new election key to this election. This key must be at least 12 characters long and have upper and lower case characters, numbers and special characters. The ↻ symbol generates and fills the field with a strong key.
- **Confirm new election key**: reinsert the same key inserted on the New election key field to confirm this key.

Clicking on the Confirm button the new key is generated and the manager is redirected to Election Manager page. Clicking on the Cancel button deletes the data in the fields and does not change the election key and redirects the manager to Election Manager page.

4.13 Election Status

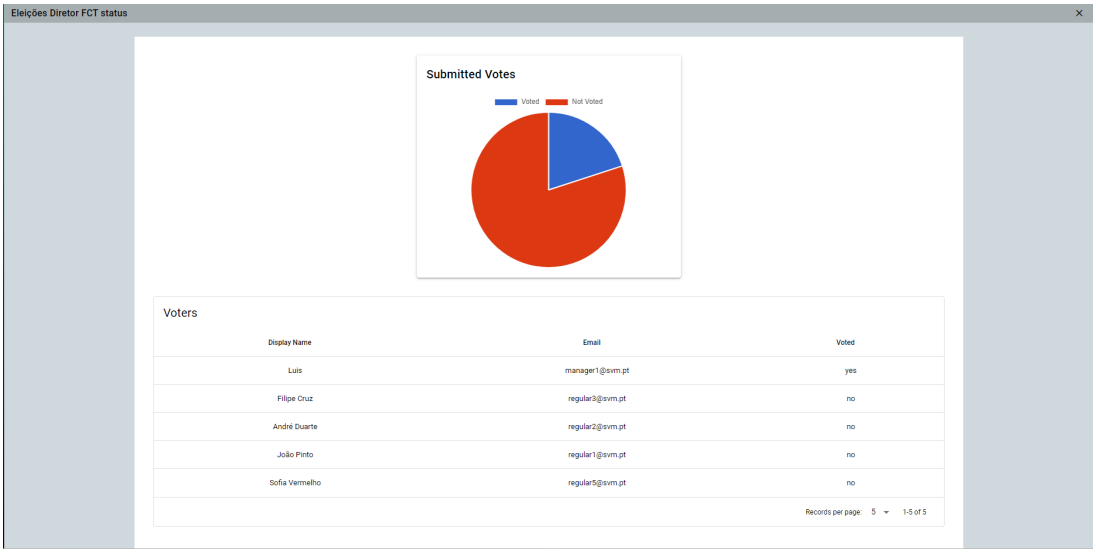

Figure 27: Menu for election status.

This menu allows authenticated managers to view the election status of an ongoing managed election. On this window managers can check the affluence of the election as well as which voters have already cast their vote.

4.14 Tally Election Votes

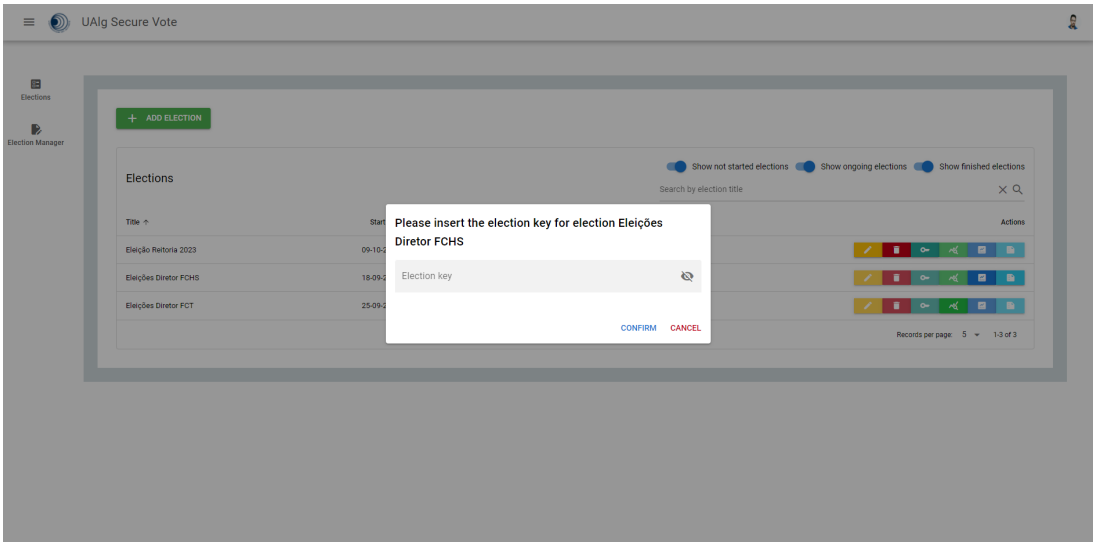

Figure 28: Menu for tallying election votes.

This menu allows authenticated managers to tally a managed election's votes. To being the tallying process a manager must fill the following field:

- **Election key**: the key chosen for this election.

After inserting the key in the field click on the Confirm button. If the election key is correct then the votes will be tallied and the manager is be redirected to Election Manager page, otherwise a message stating the election key is wrong will be shown.
The Cancel button clears the Election key field and redirects the manager to Election Manager page.
If a possible fraudulent election is detected during tallying the results will only be visible to managers and auditors and auditors will receive an email regarding the detection of a possible fraudulent election.
If no detection of fraudulent election exists or if the auditors deem it a false positive the results will then be available to all eligible voters and each one of these voters will receive an email stating the results are now available to be viewed.

4.15 Manager Election Results

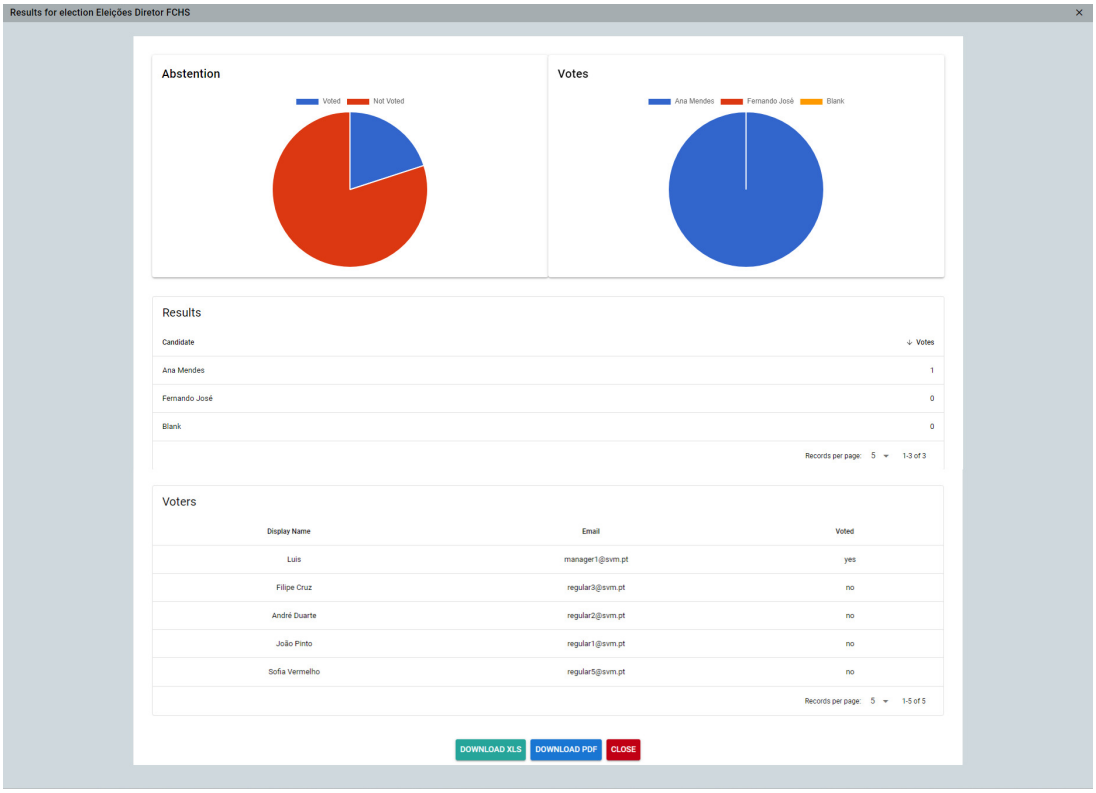

Figure 29: Menu for election results as manager.

This menu allows authenticated managers to view the managed election results. Just as in Election Results, this window shows the election results but it also shows the eligible voters list and which voters voted or not. Managers can also download these results as a PDF or a XLS file by clicking on the Download PDF and Download XLS buttons, respectively.

The Close button closes this window and redirects the manager to Election Manager page.

4.16 Auditing

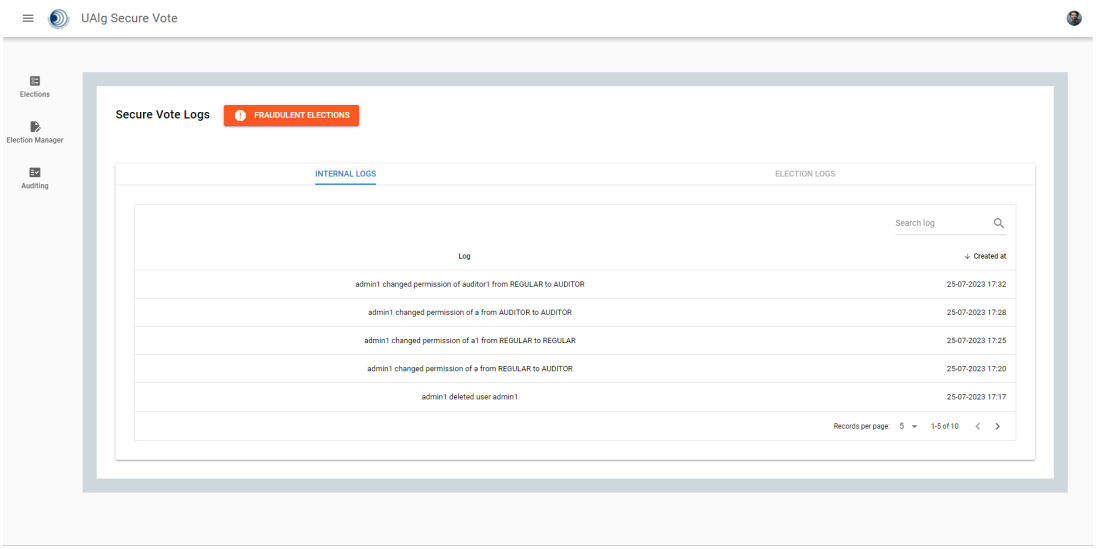

Figure 30: Menu for auditing internal logs.

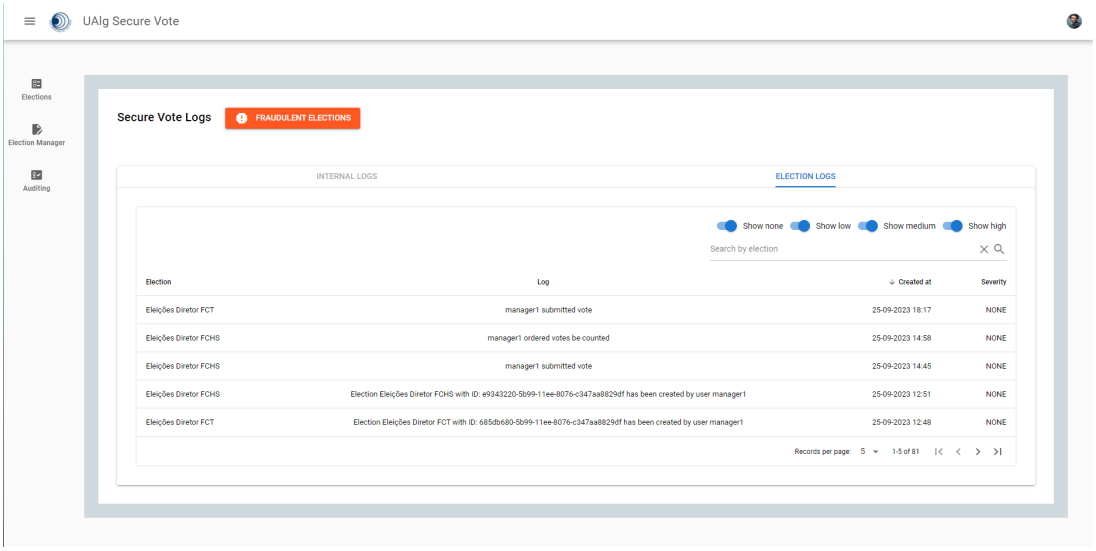

Figure 31: Menu for auditing election logs.

This menu allows authenticated auditors to visualize the internal and election logs. Logs are all the changes made by admins and actions occurred during the election period. If there is a fraud suspicion on a election result, a "detection" will be set and auditors can visualize the election logs and determine, if it is a false positive, deny such suspicious and accept the results or vice-versa. This menu also supplies filters to facilitate the task of log auditing. Clicking on the Fraudulent Elections button opens the Fraudulent Elections menu which shows the elections that are suspected of being fraudulent.

4.17 Fraudulent Election

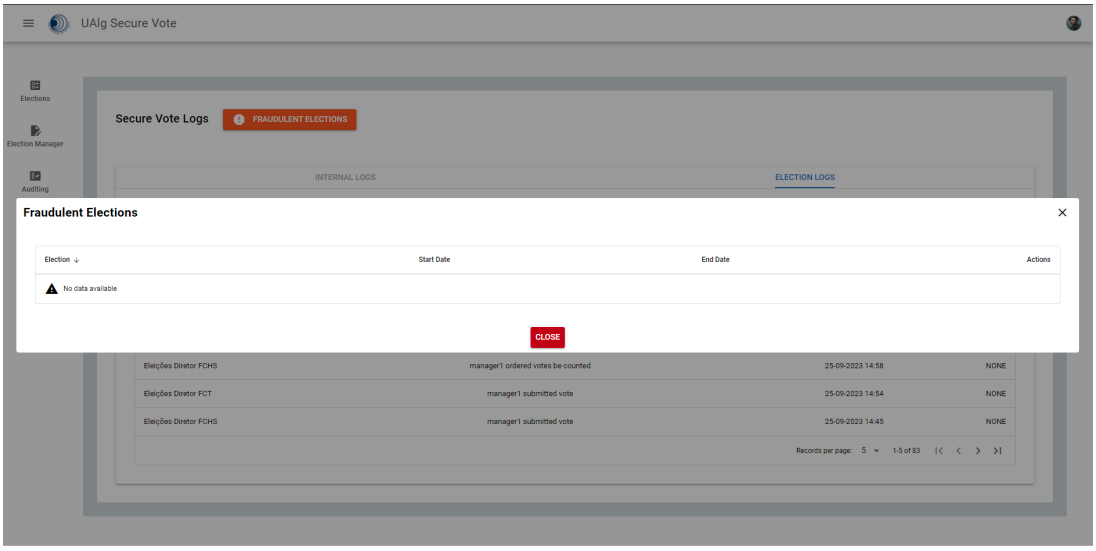

Figure 32: Menu for fraudulent elections.

This menu shows elections suspected of being fraudulent. If auditors deem the possibility of fraud a false positive auditors can deny said fraud by clicking on the ⦸ button. This button prompts the auditor to insert the motive for the denial which is registered as an election log.

4.18 Admin

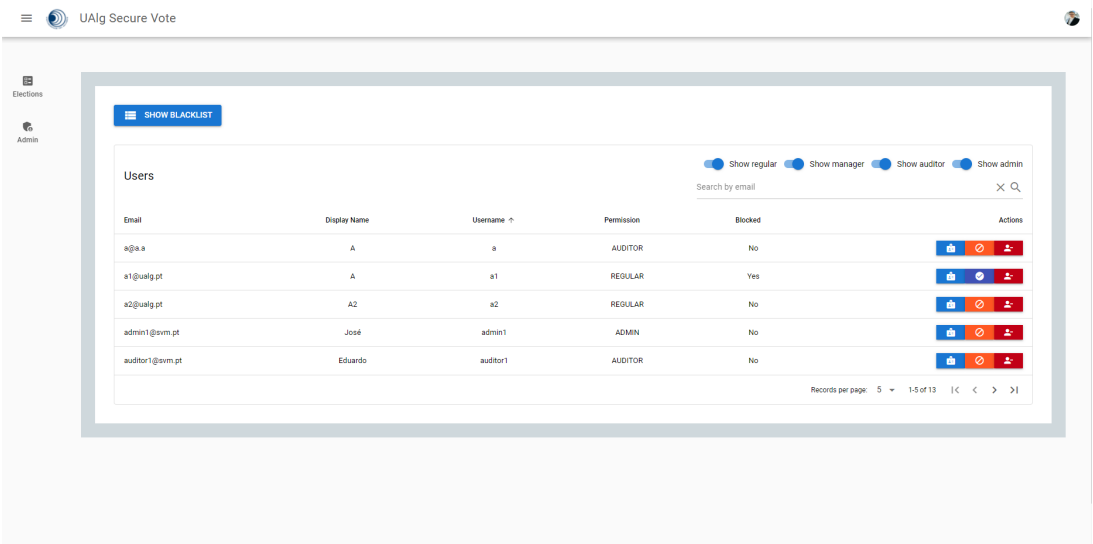

Figure 33: Menu for admin.

This menu allows authenticated admins to view all users registered in the module. Admins can change their permissions, block, unblock and delete users. It is only possible to block and delete users if the user is not a voter in an active election. As with other menus, this menu also supplies filters to facilitate the specific users search. Clicking on button opens the Change Permission menu that allows admins to change user permissions. Clicking on the button allows admins to block said user while clicking on the button allows admins to unblock said user. Only blocked users can be unblocked and vice-versa. Clicking the button allows admins to delete a user account.

4.19 Change Permission

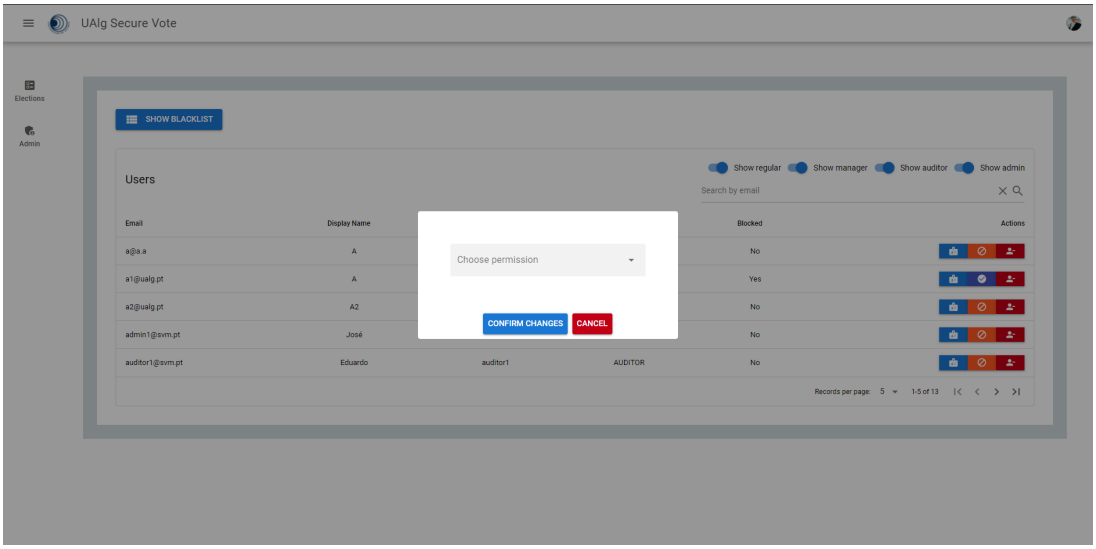

Figure 34: Menu for changing user permissions.

This menu allows authenticated admins to change user permissions. After selecting the desired permission from the dropdown clicking on the Confirm changes button will confirm the changes and redirects the admin to the Admin page.. The Cancel button clears the choice from the dropdown, if any, and redirects the admin to the Admin page.

4.20 Block/Unblock User

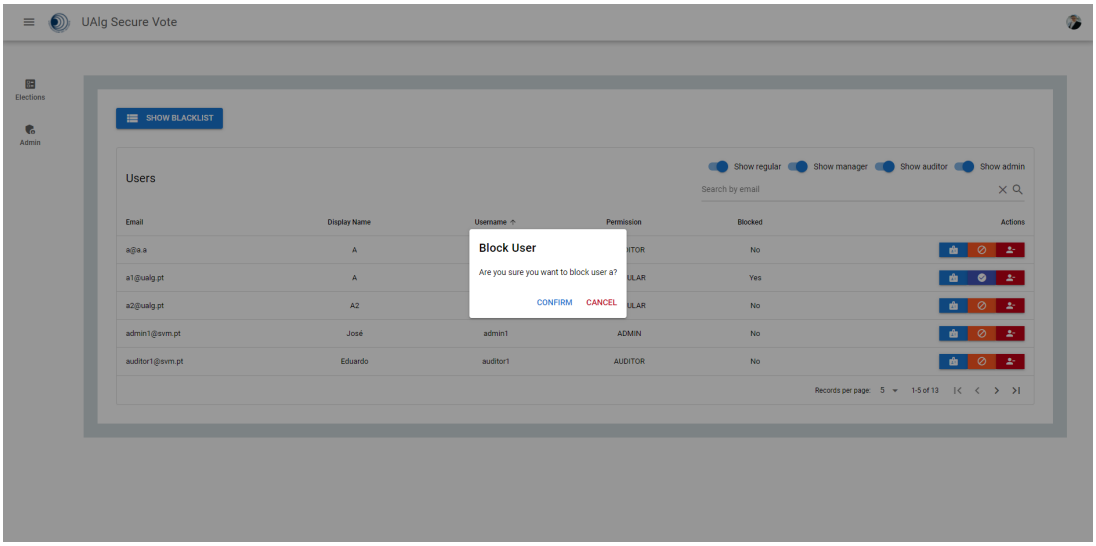

Figure 35: Menu for blocking user.

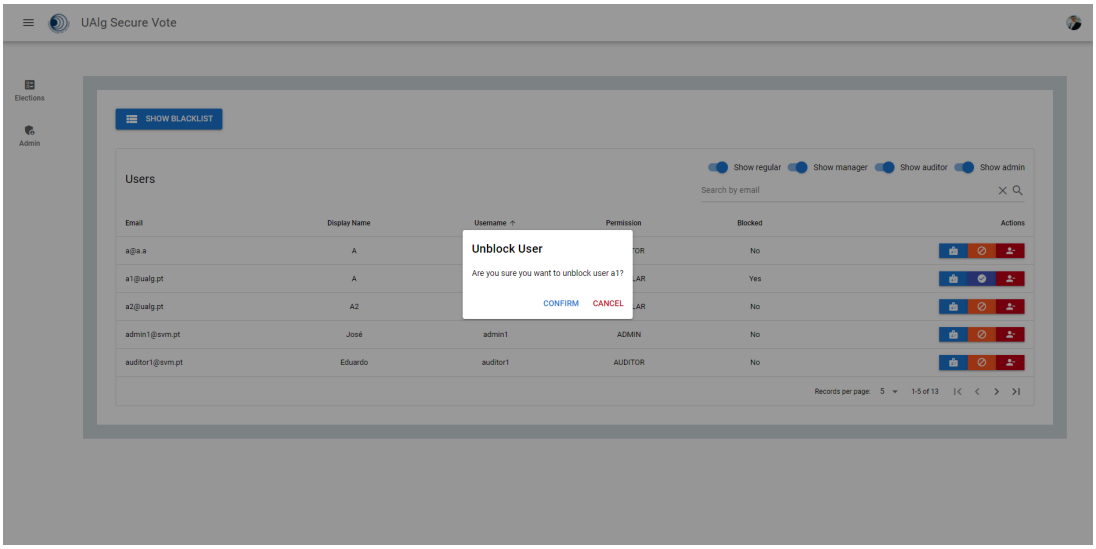

Figure 36: Menu for unlblocking user.

These menus allow authenticated admins to block and unblock user accounts, respectively. Clicking on the Confirm button confirm the block/unblock while the Cancel button cancels the process. A user cannot be blocked if it is a voter on an active election.

4.21 Remove User

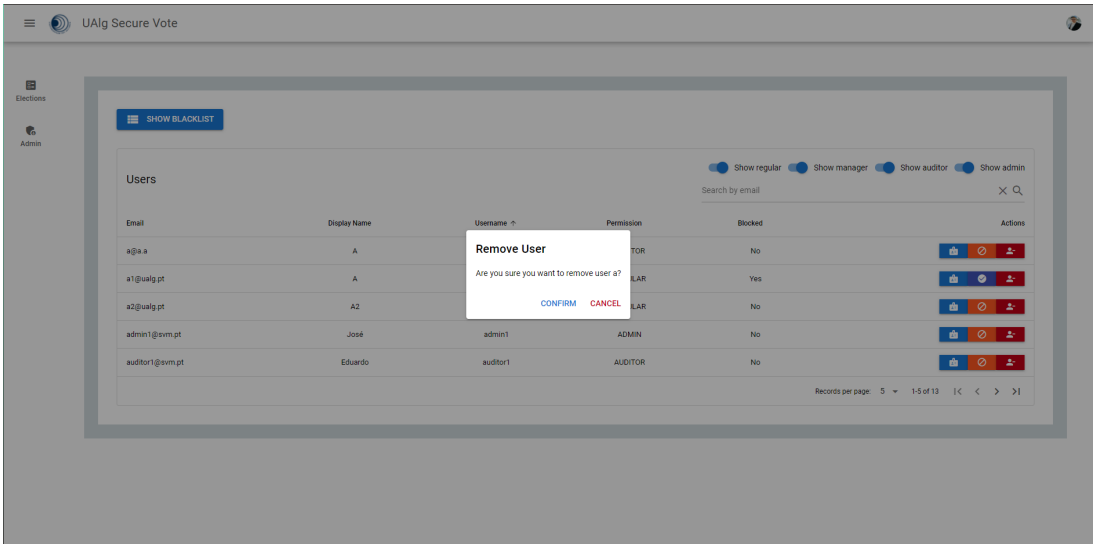

Figure 37: Menu for deleting user.

This menu allows authenticated admins to delete user accounts. Clicking on the Confirm button deletes the user account. The Cancel button cancels the deletion process. A user cannot be deleted if the user is a voter on any election.

4.22 Blacklist Email

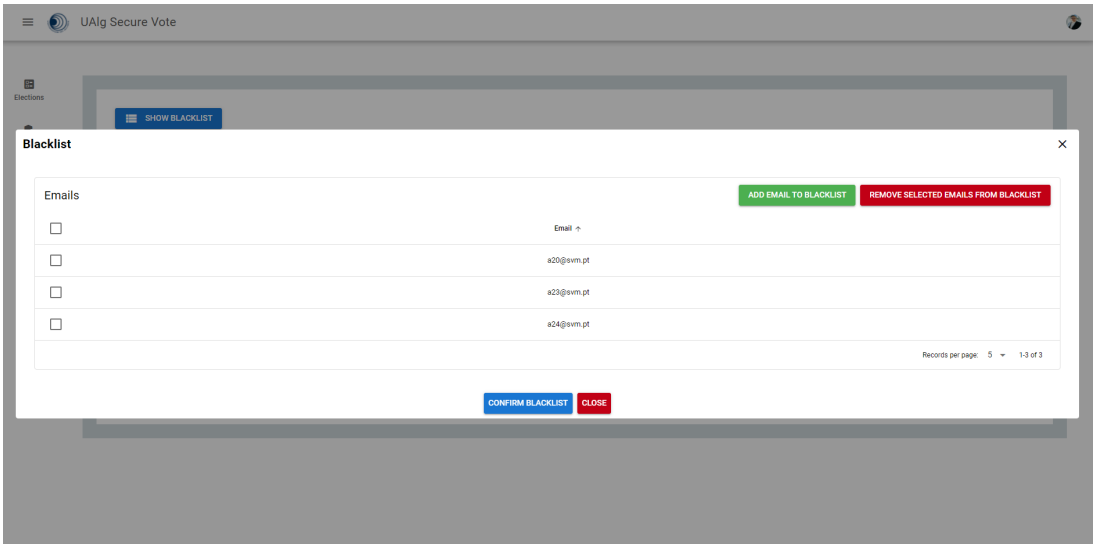

Figure 38: Menu for blacklisting an email.

This menu allows authenticated admins to blacklist emails, preventing said email from registering. Clicking on the Add email to blacklist button prompts the admin to insert the email to blacklist. To remove emails from the blacklist an admin must check their checkbox and click on the Remove selected emails from blacklist button. To confirm the blacklist the admin must click on the Confirm blacklist button. The Close button undoes all non saved changes on the blacklist.

4.23 Profile

Figure 39: Menu for user profile.

This menu allows users to view their profiles. Users can change their account settings, regenerate their vote key and delete their account. Users can change:

- **Display name**: the name the user wants to be displayed in the system, usually the user's name.
- **New password**: the password the user wants to use to access the system. This password must be at least 8 characters long and have upper and lower case characters, numbers and special characters. The ↻ symbol generates and fills the field with a strong password.
- **Confirm new password**: the password entered on the New password field must be reentered in this field to be confirmed.
- **Select Image**: upload an image to be the user's avatar in the system. Only .jpg, .png and .svg files of up to 1MB size are accepted.

The user can change just the information it needs, all others will remain unchanged. Clicking on the Confirm changes button will confirm the changes while the Cancel button cancels all changes made, keeping the information as it was.

To regenerate the vote key the user must fill the following field:

- **New vote key**: the new vote key the user wants to use to confirm its vote. Whenever the user casts a vote the user must enter this key in order to confirm the vote. This new key must be at least 12 characters long and have upper and lower case characters, numbers and special characters. The ↻ symbol generates and fills the field with a strong vote key.
- **Confirm new vote key**: the vote key entered on the New vote key field must be reentered in this field to be confirmed.

Clicking on Confirm changes saves the new vote key while the Cancel button cancels the regeneration process.
To delete the account the user must click on the Delete account button, after which the user is prompted to confirm the decision. Clicking on the Confirm button deletes the account while the Cancel button cancels the deletion process.

CHAPTER 5

Cryptography

5.1 Cipher Suite

A cipher suite is a set of cryptographic algorithms. It usually is used together with the TLS or SSL protocols [43]. Every cipher suite specifies the algorithms for:

- Key exchange: used to protect shared keys;
- Bulk encryption: used to encrypt messages between clients and servers;
- Message authentication: used to verify the messages integrity.

Cipher suites are usually written in the following format:

TLS_ECDH_WITH_AES_256_CBC_SHA512_P384

In which:

- ECDH is the key exchange algorithm;
- ECDSA is the signature algorithm;
- AES_256_GCM is the bulk encryption algorithm;
- SHA512 is the message authentication algorithm;
- P384 is the elliptic curve algorithm.

This module has to be secure [34], therefore a cipher suite has to be used and specified. The following cipher suite is used by both Secure Vote and KMS:

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_RSA_4096_SHA256_K571

In which:

- ECDHE is the key exchange algorithm;
- ECDSA is the signature algorithm;
- AES_256_GCM and RSA_4096 is the bulk encryption algorithm, where some data is encrypted using RSA and other is encrypted using AES;
- SHA256 is the message authentication algorithm;
- K571 is the elliptic curve used.

5.2 Transport Layer Security

Transport Layer Security or TLS is a client/server protocol stacked on top of the reliable transport layer protocol, TCP. TLS is structurally identical with the SSL protocol.

On a lower layer, TLS Record Protocol fragments, optionally compresses, and encrypts high-layer protocol data. TLSPlaintext, TLSCompressed and TLSCiphertext data structures correspond to these fragments respectively and, just as SSL, each has four fields [44]:

- **Type:** refers to the high-layer protocol;
- **Version:** refers to the protocol version;
- **Length:** refers to the fragment byte length;
- **Fragment:** refers to the high-layer protocol data, usually a TLSCiphertext data structure. This field is arbitrarily long, up to 2^{14} bytes;

On a higher layer, TLS is composed of four protocols, similar to SSL [45]:

- **Change Cipher Spec Protocol:** allows the communicating peers to signal transitions in ciphering strategies;
- **Alert Protocol:** allows the communicating peers to exchange alert messages;
- **Handshake Protocol:** allows a client and a server to authenticate each other;
- **Application Data Protocol:** allows communicating peers to exchange data according to some application layer protocol like HTTPS;

TLS protocol uses cipher suites to improve security and, similar to SSL, makes a distinction between session and connection. A connection is a protocol operating environment representation and several connections may correspond to a single session.

Identical to SSL, TLS has four connection states:

- **Compression state:** compression algorithm current state;
- **Cipher state:** encryption algorithm current state;
- **MAC secret:** MAC secret for the connection;
- **Sequence number:** sequence number, starting at zero, for the records transmitted on a specific connection state;

TLS also as four connection states: the current and pending read and write states, TLS records are processed under the current states while security parameters and elements are processed in the pending states and set during TLS Handshake Protocol.

TLS session elements are essentially the same as state elements of an SSL session. These are:

- **Session identifier:** byte sequence chosen by the server to identify and active or resumable session state;
- **Peer certificate:** if available, the X.509v3 peer certificate;
- **Compression method:** data compression algorithm used before encryption;
- **Cipher spec:** data encryption and MAC algorithms used;
- **Master secret:** secret shared between client and server;
- **Is resumable:** indicates if the session is resumable;

TLS and SSL are very similar only differing on certain aspects like at connection level, where TLS distinguishes security parameters from state elements while SSL does not distinguish, only considering state elements. TLS also generates the keying material differently from SSL. SSL uses a master secret and key block. TLS 1.0 also uses a unique construction but it is made around a TLS-specific Pseudo Random Function or PRF. TLS's PRF uses a secret, a seed and a label to generate a long bit sequence. In order to do this, combines MD5 and SHA1 hash functions. PRF is used to generate the keying material required for a TLS connection.

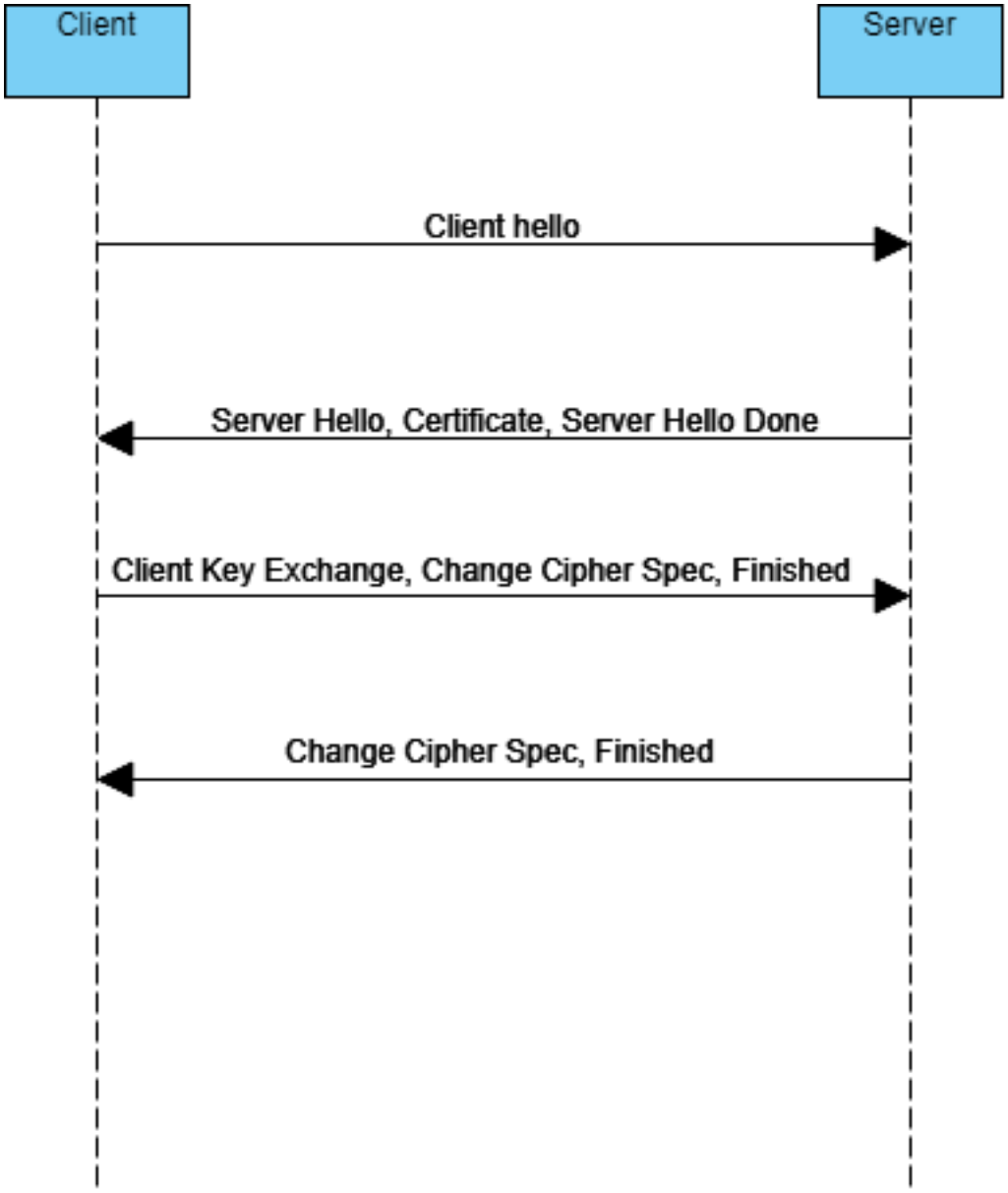

Figure 40: Example of TLS handshake.

A long master secret is required, so to produce such secret, the process uses a PRF with a obtained secret from a key exchange algorithm, the label and a randomly obtained seed, both in the client and server sides. The secret is then used to create a key block with PRF, the label and another obtained seed the same way as before. The key block can then be partitioned into different values with appropriate size. These values are:

- client_write_MAC_secret
- server_write_MAC_secret
- client_write_key
- server_write_key
- client_write_IV
- server_write_IV

Any remaining material is discarded and if the cipher suite is exportable has the is_exportable as true. The write keys are used to create the final write keys following the same process originally used to create the previous write keys, but using those specific write keys instead of the master secret. Also in this case the IVs are derived from the seeds without any secret.

Other protocols may use TLS key establishment and then use the key material for another purpose. In these situations, the keying material is called Exported Keying Material or EKM, created by using a PRF with the master secret, the label and a seed but the seed also has a context value and length from the both client and server [46].

Through the years TLS as been improved and several iterations have been created, the most recent version is TLS 1.3. The process described in this section shows in overall how TLS works and how it may have some changes with each iteration, but the goal remains the same, to secure communications and it is efficient.

5.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography or ECC is a branch of cryptography which uses elliptic curves on finite fields and depends on the Elliptic Curve Discrete Logarithm Problem or ECDLP difficulty, which is a known NP-Hard problem.

ECC uses elliptic curves over finite fields and public and private key pairs, where the private key is an integer and the public key an exact point on the elliptic curve.

Every ECC uses elliptic curves, these are algebraic curves defined as an equation. An elliptic curve equation is defined by:

$$y^2 + xy = x^3 + ax + b$$

The equation varies from curve to curve. NIST curve K-571 is the defined curve on the cipher suite used on Secure Vote module. The equation is:

$$y^2 + xy = x^3 + ax^2 + b$$

Using a specific curve over a finite field will form a cyclic algebraic group, that has all the points on the curve. In these groups, if two points are added or multiplied to an integer number, the result will be another point from this group and curve. The total number of group points on a curve plus a point at infinity is called the **order of the curve**.

While some curves form a single group, other curves form several groups, each with their own points. In this case, the points are spread across multiple subgroups, each with their own order. The order of all groups is calculated with:

$$c = n/s$$

where c is the curves co-factor, calculated by the total amount of subgroups, n is the curve's order, calculated by the total amount of points on the curve, and s is the subgroups order.

ECC is used for cryptography in order to generate the public/private key pair. To do this it is required to set a starting point on the curve, called **generator point or G**, then used to generate all other points of its group or subgroup, if applied, on the curve. This is done by multiplying G by an integer between 0 and the group's order.

Using this process and setting a **generator point G** and a **private key K**, which is an integer, the **public key P** can then be calculated by multiplying G with K, originating a group point on the curve, this point is the public key. Public key is calculated with:

$$P = K \cdot G$$

The process of calculating the public key is extremely fast, the speed varies from curve to curve and the K size, but the reverse process is extremely slow, almost impossible to perform if K is large enough. This is due to the existence of an NP-Hard problem called the Elliptic Curve Discrete Logarithm Problem or ECDPL, that states if it is possible, in a given curve over a finite field, a generator point G on the curve and point P on the curve, find an integer K that satisfies:

$$P = K \cdot G$$

By carefully choosing the curves and finite fields this is made almost impossible.

This encryption types are usually secure and relatively efficient but it may be subject to security issues. As the curve not being "secure", usually causes the cyclic groups to have too few points and can be exploited, or the private key size being too short. [47] [48]. NIST curve K-571 was chosen to be used in this cipher suite because it has over 571 bit of binary field where the curve's finite field will have a large size with a good length for the private key, being secure enough.

Algorithm 1 ECC Key Pair Generation Pseudocode

1: Select an elliptic curve E and a generator point G on the curve
2: Choose a private key K, a random integer in the range $[1, n-1]$, where n is the order of the base point G
3: Compute the public key P as the point multiplication of the generator point G by the private key P : $P = K \cdot G$
4: The public key P is represented as (x, y) coordinates on the curve, and the private key is K
5: Return P and K

5.4 Elliptic Curve Diffie-Hellman Ephemeral Key Exchange

As stated previously, the cipher suite used for the key exchange algorithm is Elliptic Curve Diffie-Hellman Ephemeral or ECDHE.

ECDH works by having two parties generating a public/private key pair using ECC, then sharing the public keys between them and a shared secret is calculated. Different from the regular Diffie-Hellman key exchange, where modular exponentiation is used, ECDH uses point multiplication which is a process that occurs on ECC:

$$(p_1 \cdot G) \cdot p_2 = (p_2 \cdot G) \cdot p_1$$

with **p1** and **p2** are two private keys and **G** a generator point.

$(p_1 \cdot G)$ and $(p_2 \cdot G)$, as shown in 5.3, are both parties public keys, respectively. After both public keys are calculated and shared between parties, each will use the others public key and its own private key to calculate the shared secret:

$$secret = p_1 \cdot d_2 = p_2 \cdot d_1$$

p1 and **p2** are the parties private keys, respectively, and **d1** and **d2**, the parties public keys, respectively [49].

ECDHE and ECDH are essentially the same key exchange algorithm and the main difference is that ECDH uses a single secret to encrypt all exchanges in a session while ECDHE uses a different secret for every exchange, providing better secrecy, not given by ECDH.

Secure Vote and KMS use ECDHE together with AES-256-GCM to encrypt the data sent between each other. The encryption key is calculated on each API using ECDHE where every communication between those APIs always has a distinct encryption key.

Algorithm 2 ECDH Shared Secret Generation Pseudocode

1: Select an elliptic curve E and a generator point G on the curve

2: Calculate Party 1 and Party 2's public/private key pair using algorithm 1

3: Party 1 and Party 2's public/private key pair will be (d_1, p_1) and (d_2, p_2) respectively

4: Party 1 sends d_1 to Party B, and Party B sends d_2 to Party A

5: Calculate the secret: $secret = p_1 \cdot d_2 = p_2 \cdot d_1$

6: Return $secret$

5.5 Secure Hash

Secure Hash or SHA is a hashing function that follows the Merkle-Damgard construction scheme. The scheme extends the original message to a size that is a multiple of an integer, ex. 256 or 512. The message is then divided into blocks with the chosen integer size and in each block a compression function and a vector are applied. After the finalization function is applied to the modified vector, the output is the pretended hash.

SHA follows a simple rule, more bits means more security, the bigger the chosen integer the securer the hash will be, therefore SHA-512 is safer than SHA-256 [50].

Secure vote uses SHA-256 when signing votes and for integrity checks, secure and strong for this processes, together with ECDSA and a SHA-256 hash, it is faster to produce compared to SHA-512 hash, although the difference is barely noticeable with modern hardware.

5.6 Elliptic Curve Digital Signature Algorithm

Elliptic Curve Digital Signature Algorithm or ECDSA is a Digital Signature Algorithm or DSA variant that uses ECC, like ECDH is a variant of an already existing algorithm which uses ECC. It is used to digitally sign and verify messages, checking message origin and integrity.

ECDSA uses a created ECC, described in 5.3, public/private key pair. To sign a message, ECDSA uses the private key and the message to create a signature:

Algorithm 3 ECDSA Signature Generation Pseudocode

1: Select an elliptic curve E and a generator point G on the curve
2: Generate the public/private key pair using algorithm 1
3: Create hash C from the message M to be signed
4: Generate a random number p in the range $[1, n-1]$
5: Calculate a random point $R = p \cdot G$ and extract the x-coordinate
6: Calculate the value $r = x \bmod n$, where n is the order of the curve
7: **if** $r = 0$ **then**
8: Go back to step 4 and choose a different random p
9: **end if**
10: Compute $s = p^{-1} \cdot (C + r \cdot d) \bmod n$, where C is the hash of the message
11: **if** $s = 0$ **then**
12: Go back to step 4 and choose a different random p
13: **end if**
14: Return (r, s)

In order to verify the signature, ECDSA uses the public key. The process to verify a signature is:

Algorithm 4 ECDSA Signature Verification Pseudocode

Require: Public Key P, Signature (r, s), Message M

1: Select an elliptic curve E and a generator point G on the curve
2: Verify that P is a valid point on the curve
3: Verify that P is not the point at infinity
4: Verify that the coordinates of P are in the valid range
5: Verify that r and s are integers in the range $[1, n-1]$, where n is the order of the generator point G
6: Calculate the hash value C of the message M
7: Calculate $v = s^{-1} \mod n$
8: Calculate $u_1 = (C \cdot v) \mod n$ and $u_2 = (r \cdot v) \mod n$
9: Calculate the temporary point $T = (u_1 \cdot G + u_2 \cdot P)$
10: Extract the x-coordinate of P and calculate $V = x \mod n$
11: **if** $V = r$ **then**
12: Return signature is valid
13: **else**
14: Return signature is not valid
15: **end if**

The same hash function and curve must be used when signing and verifying, otherwise the signature cannot be validated and will always be invalid. As stated in 5.3, the curve difficulty and the key length will have a strong impact on how secure these signatures are. If the curve is too weak and the key is too short these signatures may be forged, causing a serious security issue if they are used [51].

Secure Vote uses ECDSA to verify if the cast votes belong to the voters who casted them, without seeing the vote itself. If the signature cannot be validated then the cast vote is either forged or was tempered, in any case, this vote is discarded and the attempt of submitting a forged or tempered vote is logged, therefore auditors will be able to see it, in case an audit is performed.

5.7 Advanced Encryption Standard

Advanced Encryption Standard or AES is a symmetrical encryption block cipher with 128 bits block size algorithm, nowadays widely used because it is considered highly secure. So far, no practical attacks are known and the algorithm has not yet been cracked. When talking about encryption, it is not a question if, but when. Eventually every algorithm will fail, if the method is discovered or the evolving hardware may actually brute-force in time the encryption is yet active, becoming encryption a trivial task.

AES is a symmetrical encryption algorithm, has a single key that encrypts and deciphers data, by comparison asymmetrical encryption algorithms uses a public/private key pair where one key encrypts and the other deciphers. Just as other algorithms, AES also allows different key lengths with 128 bits being secure enough for most cases but in more sensitive information, 256 bits is recommended.

AES is essentially a series of operations performed on an 16 bytes each block array. AES transforms plaintext into ciphertext with a sequence of stages. How many times this sequence of stages is performed depends on the key length.

To encrypt AES follows four different stages:

- **SubBytes:** modifies the bytes in the array independently;
- **ShiftRows:** rotates the four rows of the array independently;
- **MixColumns:** modifies the four columns of the array independently;
- **AddRoundKey:** apply XOR operation with array and round key;

The completion of all these stages makes a round, these rounds are performed a set number of times dependent on the key length. AES requires an initial AddRoundKey and the last round does not contain the MixColumns stage.

During the SubBytes stage, AES uses S-Box, providing a set of 256 possible input bytes permutation to modify the byte values, making sure the modified bytes are neither the same byte or an equivalent byte that compliments it.

During ShiftRows each block array row is rotated to the left a number of times, defined by the row order starting at 0, where the first row rotates 0 times while the last row rotates 3 times.

During MixColumns the block array is regarded as a column vector over the Rijndael field. The block array as a column vector multiplied by a specific 4x4 matrix over the field, updating the block array bytes.

During AddRoundKey the user key set is processed using AES key scheduled to supply the 16 bytes round keys required by AES. The round key is used to modify the block array by performing a XOR in each block using the round key.

AES key schedule works by taking the round r's round key (on the first round the round key is the user supplied key) and creating 4 blocks from the round key $K_{r,i}$, each block having 4 bytes, 16 bytes total starting in $K_{r,0}$ up to $K_{r,15}$. Then 4 temporary words of 4 bytes each T are calculated by:

$$T_0 = S[K_{r,13}] + \theta^r; T_1 = S[K_{r,14}]; T_2 = S[K_{r,15}]; T_3 = S[K_{r,12}]$$

θ is the Rijndael root, S the AES S-Box and i the byte's order on the key. The next key s is then defined by following:

$$K_{s,i} = K_{r,i} + T_i \qquad \text{if} \quad 0 <= i <= 3$$

$$K_{s,i} = K_{r,i} + K_{s,i-4} \qquad \text{if} \quad 4 <= i <= 15$$

To decipher, AES performs the encryption operations in reverse order and using the round keys in reversely. The first operation is also AddRoundKey.

During InvSubBytes, AES S-Box is inverted and the SubBytes process is then applied with this inverted S-Box to modify the bytes.

During InvShiftRows, each block array row is rotated to the right i positions, where i is the row order like in ShiftRows.

During InvMixColumns, like MixColumns, the block array is viewed as a column vector over the Rijndael field. The column vector is multiplied by the specific 4x4 matrix inverse over the field, then updates the column vector. MixColumns is not applied on the last round InvMixColumns and not applied on the first round of the decipher rounds [52].

Algorithm 5 AES-256 Encryption Pseudocode

Require: Plaintext P, Encryption Key K

Initialize AES-256 with K
Divide P into 16-byte blocks $P_1, P_2, \ldots, P_n$
Let C be an empty array
for i from 1 to n **do**
    Generate round keys from K
    Round 1:
        AddRoundKey
    for $round$ from 2 to 10 **do**
        SubBytes
        ShiftRows
        MixColumns
        AddRoundKey
    end for
    Round 11:
        SubBytes
        ShiftRows
        AddRoundKey
    Append the result to C
end for
Return C

Algorithm 6 AES-256 Decryption Pseudocode

Require: Ciphertext C, Decryption Key K

Ensure: Plaintext P

Initialize AES-256 with K
Divide C into 16-byte blocks $C_1, C_2, \ldots, C_n$
Let P be an empty array
for i from 1 to n **do**
    Generate round keys from K
    Round 1:
        AddRoundKey
    for $round$ from 2 to 10 **do**
        AddRoundKey
        InvMixColumns
        InvShiftRows
        InvSubBytes
    end for
    Round 11:
        AddRoundKey
        InvShiftRows
        InvSubBytes
    Append the result to P
end for
Return P

AES has an operation multiple mode, Galois/Counter Mode or GCM. GCM uses a Counter mode variation to provide confidentiality and authenticity proof through a universal hash function defined over a binary Galois field. GCM consists of two functions that are the inverse of one another, these functions are:

- **Authenticated encryption:** encrypts confidential data and produces the authentication tag for both confidential and non-confidential data.
- **Authenticated decryption:** deciphers confidential data, providing an presented and verified authentication tag.

Authenticated encryption function has three parameters: the plaintext **P**, the additional authenticated data **AAD** (ex. ports, sequence numbers, etc.) and the initialization vector **IV**. The P and AAD are data that GCM protects while the IV is a unique value within the specified context that determines an invocation of the authenticated encryption function on the data to be protected, producing a ciphertext and an authentication tag.

Authenticated deciphering functions has four parameters: the IV, the ciphertext, the AAD and the tag. This function will either produce the plaintext from the ciphertext or a special error code. If the correct authentication tag for a specified IV, AAD and ciphertext is not supplied, then the output will have an error. The IV should always be unique to the ciphertext, repeating IVs causes vulnerability allowing third parties the ability to determine the hash subkey, allowing, therefore, to forge the ciphertext [53].

Secure Vote uses AES_256_GCM to encrypt the private keys, both for the signature keys and election keys, afterwards stored on the KMS. It is also used for the module's internal election results encryptions, and with ECDHE, encrypts the communication between Secure Vote and KMS. Private keys are considered sensitive information, so 256 bits key length was chosen to improve security. GCM mode of operation was chosen due to its capabilities of verifying the encrypted data integrity, severely important in either the communication between Secure Vote and KMS and the keys themselves. Since GCM has a problem with repeating IVs, whenever data is encrypted using this algorithm, a new IV is created in order to mitigate this problem.

5.8 Scrypt

To encrypt and decipher using AES a key must be supplied to satisfy the algorithm. Since the Secure Vote and KMS APIs are using AES_256_GCM, a 256 bit key must be supplied. A 256 bit key has a size of 32 bytes or 32 characters long. A 32 character long key is not feasible for a user to insert or memorize, so in order to solve this problem, scrypt was used.

Scrypt is a sequential memory-hard key-derivation function or KDF. A KDF is a function that transforms a string like a password into another string of a predetermined length. Other KDFs exist and all provide protection against attacks like brute-force, scrypt in particular is resistant against ASIC attacks, that uses custom hardware to crack keys [54].

Secure Vote users, when creating either their signature key or the election keys, will have to insert a 12 character long key, enough to mitigate brute-force attacks. Applying KDF, the user inserted key will be 32 characters satisfying AES_256_GCM, allowing the keys encryption and decipher. 12 characters can easily be memorized but a safe storage should be used to save the keys.

5.9 Rivest-Shamir-Adleman

Rivest-Shamir-Adleman algorithm or RSA is an asymmetric key cryptography algorithm that uses a public/private key pair where the one key encrypts and the other deciphers data. RSA uses modular exponentiation to perform its operations. The correctness of RSA can be proven with:

$$(M^e)^d \equiv M \mod N$$

M is the plaintext, e the encryption exponent, d the decryption exponent and N the modulus. The public/private key pair generation is achieved by:

Algorithm 7 RSA Key Generation Pseudocode

Require: Key Length L (in bits)

1: Select two distinct large prime numbers p and q of about $\frac{L}{2}$ bits
2: Calculate the modulus $N = p \cdot q$
3: Calculate the totient $\phi(N) = (p-1)(q-1)$
4: Choose a a random integer e that satisfies $1 < e < \phi(N)$ and gcd$(e, \phi(N)) = 1$
5: Calculate d that satisfies $d \equiv e^{-1} \pmod{\phi(N)}$
6: Public Key will be (N, e) and Private Key will be (N, d)
7: Return $(N, e), (N, d)$

RSA encrypts a plaintext M into a ciphertext C by using:

$$C \equiv M^e \mod N$$

RSA deciphers a ciphertext C into a plaintext M by using:

$$M \equiv C^d \mod N$$

The modulus length N determines how strong an RSA cipher will be, but if the modulus length is too high it will take a considerable amount of time to generate the public/private key pair, so the circumstance in which RSA is applied has to be properly assessed to conclude just how strong the encryption must be [55].

Secure Vote uses RSA to encrypt the votes cast on the elections, the public key used to encrypt the vote and the private key to decipher votes. Vote encryption, using the public key, created on the client side to maximize vote privacy, so Secure Vote's API, in no circumstances, accesses the plaintext vote associated with the voter who cast it.

5.10 Hash-based Message Authentication Code

Hash-based Message Authentication Code or HMAC is an algorithm used to create Message Authentication Codes or MACs to verify data integrity. A method to create MACs is to use an hash function but HMAC differs by using a secret key while hash function do not require such keys. HMAC still requires a hash function to properly operate and this must be secure.

In order to create a hash for a HMAC message:

Algorithm 8 HMAC Pseudocode

Require: Message M, Key K, Hash Function H

if K length > H block size **then**
    Hash K with H and use result as the key
else
    if K length < H block size **then**
        Add zeros to the right of K until it reaches the block size and use result as key
    else
        Use K as key
    end if
end if
Perform a XOR operation with the key and 0x36 to create the inner pad $ipad$.
Perform a XOR operation with the key and 0x5C to create the outer pad $opad$.
Concatenate $ipad$ and M and hash the result using H to create $h1$.
Concatenate $opad$ and $h1$ and hash the result using H to create $h2$.
Return $h2$.

The result $h2$ is a hash that can be used to verify message integrity and if it has been compromised [56].

Secure Vote uses HMAC to verify if the user submitted casted vote is exactly the same that arrives on the API. ECDSA already verifys integrity, but HMAC is used to reinforce it because casting a vote is a sensitive process and all cast votes should be treated with utmost care, so the vote is not tempered.

CHAPTER 6

System Security

6.1 JSON Web Token

When developing REST APIs, JSON Web Tokens (JWT) are commonly used to identify and authenticate users and services that are using the API[32][33]. A JWT is a token represented as a string created by encoding a stringifyied JSON object. Stringifying is a JSON object specific process that converts a JSON object into a string, and then is digitally signed using algorithms like HMAC, so, JWT requires a key to be produced. A JWT has three main components, these are:

- Header: contains the type of token and the signature algorithm.
- Payload: contains the data
- Signature: contains the hashing function

JWTs can be decoded using the key in order to obtain its payload[57].

Secure Vote uses JWTs to authenticate users and services to access their endpoints. User Permission endpoints can only be accessed after the token is verified and the database is queried to retrieve user permissions, because they are not a part of the token payload. KMS also uses JWTs to authenticate requests. Secure Vote has endpoints that do not require any authentication such as login, KMS requires authentication in all its endpoints.

6.2 Stored Procedures and Prepared Statements

Secure Vote and KMS manipulate and store data in databases. It is required that those operations are not liable to attacks. In order to mitigate possible SQL injection by sending malicious SQL code on the data sent to the database, triggering a database query, an action that is not supposed, compromising the stored data or even hinder the database performance. To solve this problem, stored procedures should be used.

A stored procedure is a set of SQL queries designed to perform a specific task. Some DBMSs have their own pre-defined stored procedures. Stored procedures help mitigate possible SQL injection attacks, but they alone are not enough. Before running a stored procedure or any query, data should be verified to block any attempts to send a malicious characters to the query, if it is valid then can be stored and not cause bad data issues.

Another way to mitigate SQL injection attacks is using Parameterized Queries also known as Prepared Statements. Prepared Statements are SQL queries that contain placeholders for the user variables. A DBMS will first compile the SQL statement without considering the placeholders and then store the result. The DBMS then adds the variables and compiles again the statement. Even if an attack occurs, the malicious code will be treated as a regular string and will not alter the query itself. Stored procedures and Prepared Statements are not mutually exclusive and can both be used at the same time, if the DBMS allows stored procedures, because not all allow stored procedures, like CassandraDB [58].

Both Secure Vote and KMS use stored procedures and prepared statements to mitigate possible SQL and NoSQL injection attacks, because these types of attacks can result in compromised data which is a severe problem, specially in this kind of applications.

6.3 Rate Limiter

Denial of Service (DoS) attacks are one of the biggest threats on the internet. In these attacks, the attacker consumes all of computing resources, leading for the affected services not being available. A common DoS attack is TCP sync flooding that takes advantage of the TCP's handshake and fills the queue with a large amount of requests in quick succession, causing the queue to fill up and any new request is rejected, thus blocking any future requests from entering, denying all services dependent on TCP. A way to mitigate DoS attacks is to use the request source IP address and provide a more strict timeout for the requests. If a timeout occurs the request is dropped. Defining trusted sources is also another method to mitigate DoS attacks [59].

API endpoints can be subjected to DoS attacks. An attacker can repeatedly send requests to the API's endpoints, the queue will fill up and all subsequent requests are discarded[32][33]. In order to mitigate this, a rate limiter is recommended, essentially defining a maximum number of requests from an IP address in a time frame, if this limit is achieved then all subsequent requests will be dropped and an HTTP 429 error: Too many requests will be sent to the IP address [60]. DoS attacks cannot be entirely prevented, but mitigation is possible and should be implemented whenever possible, the same should be attempted with Distributed Denial of Service attacks or DDoS attacks, which are a more severe variant of the DoS attack.

Both Secure Vote and KMS have been developed using Express.js framework, this framework integrates of a rate limiter on the endpoints to mitigate possible DoS/DDoS attacks.

6.4 HTTP Headers and Policies

When dealing with authenticated access, Cross-Site Request Forgery may occur. Cross-Site Request Forgery or CSRF are requests from dubious sources to access privileged information to non-authorized sources. These kinds of attacks can basically be described as an exploitation of the trust between the application and the browser. The application is unable to distinguish genuine from malicious requests causing data to be compromised.

HTTP is a stateless protocol and almost all applications require a unique state for each user, so it is common to use sessions, allowing the stored data to be accessed and altered throughout the user interaction with the application. Storing this data in cookies is commonly used to secure the session integrity and confidentiality. Compromising the stored data in cookies is severe, because cookies allow applications to store key/value pairs on the HTTP header Set-Cookie, sending the cookies stored data whenever a request is made.

HTTP header Set-Cookie accepts several attributes, one of these is the SameSite attribute, used to set restrictions on the cookies scope, telling the browsers to either include them on request or not. For SameSite, three different policies exist:

- **None:** this policy specifies that cookies are sent in all requests.
- **Lax:** this policy specifies that cookies are sent on requests issued by top-level navigation such as clicking on link but will not be sent on sub-requests such as loading media files.
- **Strict:** this policy specifies that cookies are only sent on requests that originate from the same application, never being sent on cross-site browsing contexts.

Another attribute for the Set-Cookie header is the Secure attribute that restricts the cookies usage only to secure connections, like HTTPS connections.

Most applications use web cache, storing responses to specific requests for a certain amount of time, reducing workload by not having to process the same request. Web cache poisoning attacks exploits vulnerabilities like Cross-Site Scripting or XSS, that alters the cache and, in consequence, users may receive poisoned responses resulting in an impact that may vary on what was changed in the cache. Verifying requests should be done to mitigate possible XSS attacks.

The Same-Origin Policy or SOP is another policy that can be used to mitigate possible CSRF attacks. This policy consists in restricting the access on web resources, isolating and providing boundaries from other different origin applications. Two applications have the same origin if both use the same protocol, host and port, if anything is different, then they belong to different origins. SOP creates protective walls for applications, so it may hinder functionalities and be too restrictive, blocking some shared resources.

In order to alleviate these restrictions, some solutions were presented like Cross Origin Resource Sharing or CORS [31][32][33][34]. CORS is an extension of the XMLHttpRequest API that provides permissions to access resources in cross-origin requests through a set of HTTP headers enforced by the client. Header validation is performed server side. To configure SOP in CORS, the Access-Control-Allow-Origin header is set and only applications belonging to a whitelist can request resources, all others will be denied under the CORS Access-Control-Allow-Origin. CORS also supplies a Access-Control-Request-Method header where it is possible to define what methods are accepted, all others are discarded. Access-Control-Allow-Credentials allows credentials like authentication cookies to be sent on requests. Using CORS it is possible to choose what headers are accepted on request, therefore if the request contains a header that does not belong to the accepted headers, it will be discarded. To allow CORS, preflight must be enabled allowing the browser to send HTTP request OPTIONS to obtain the CORS defined headers [61].

Both Secure Vote and KMS use these mechanisms to control access and mitigate possible CSRF attacks. Secure Vote only allows requests from specific methods, these methods are allowed, all others are blocked, just the required specific headers for its operation are permitted. Secure Vote's API has Access-Control-Allow-Credentials header set to true, receiving the authentication token as a cookie.

Secure Vote also uses Lax SameSite policy, because the interface is a different application, on the whitelist of Access-Control-Allow-Origin, and defines a maximum time for cached responses. KMS uses the same mechanisms and policies as Secure Vote with small differences. The Access-

Control-Allow-Credentials is set to false because the authentication token is sent as a header and not as a cookie. The header that contains this token, is allowed and required to successfully authenticate in Secure Vote. All other policies are the exact same functionalities as Secure Vote. Both Secure Vote and KMS also apply security through obscurity or STO, taking advantage of the X-Powered-By header that specifies what technology was used to create the application and changes it to another value, giving the illusion that a different technology was used to create the application.

CHAPTER 7

Conclusions and Future Work

7.1 Conclusions

As long as democracies exist, there will always exist the need for elections, commonly associated with democratic governments and entities providing freedom of choice to populations. As a consequence, more freedom of opinion exists. With the advances in technology, the electoral process must be able to coexist and even thrive together. Technology exists to simplify and improve people's lives. Elections should not be an exception.

Currently, there are several technological and remote election solutions, via electronic and/or using the internet, for example, to determine a country's government. [3] [35]. One of the main issues with these elections is public trust [3], the voters and parties must trust the system and respect voter's privacy. This might not be an easy task, and in some cases, may lead to security flaws found in the source code that can be made publicly available, destroying the trust on the system.

As electoral procedures and technology evolve, so should election systems. The developed system presented in this thesis follows the current electoral procedures, implementing several security algorithms and secure infrastructure.

With the evolution of technology, it is inevitable that, in the future, this system security will be put in jeopardy and it's cipher suite will no longer be strong enough to protect the system generated data, privacy and anonymity. If the system and data can be compromised, then it should not be used to held elections and should be upgraded, secured and new features implemented to avoid security issues and data compromising. Regardless, these upgrades must never compromise the user's privacy which is the key point of any democratic election.

The developed system may be expensive to implement following the diagram in 14, nevertheless it is a safer way to be deployed. The diagram divides all components into different servers and if correctly implemented will lead to a system that cannot easily be entirely compromised. Redundancy should also be implemented, although being expensive, but if any server is compromised, another server will carry on continuing the system's operations.

To conclude, regardless of how the system is implemented, with or without multiple servers and/or redundancy, this system achieves it's goal to provide a way to hold elections in a secure and private environment.

7.2 Future Work

Despite the system being secure and private in its operations and elections being held reliably, the matter of public trust still exists. Users may not want to use a system if they do not fully trust the entity that is using it. Even by releasing the source code to the public there is no guarantee that the released code is the one being used on the deployed system or that it will not be changed before the election.

A system such as the one described on this document should be deployed on the Blockchain. Using the Blockchain provides a decentralized system assuring security to voters in their identification, authentication and authorization, data transmission and data verifiability. By using the Blockchain, system voters are assured that their identification is never compromised due to how the Blockchain works. It is almost impossible to edit and hack the system at the moment the smart contract is minted, assuring users that the code will not be altered before or mid-election [26][27][29][21][23][24][62]. This Blockchain feature created the catchphrase "Code is law". The major downside of using the Blockchain is that is too expensive to implement and maintain, minting a smart contract on the Ethereum Blockchain, the most well know Blockchain, is very expensive. To maintain a system on the Blockchain requires the payment of the "gas fees" which are also expensive. An attempt to implement this system on a Blockchain will be addressed in the future.

Bibliography

[1] Serdult, Uwe and Germann, Micha and Mendez, Fernando and Portenier, Alicia and Wellig, Christoph. Fifteen years of internet voting in Switzerland [History, Governance and Use]. 2015 Second International Conference on eDemocracy & eGovernment (ICEDEG), 2015, pp. 126-132

[2] Heiberg, Sven and Willemson, Jan. Verifiable internet voting in Estonia. 2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), 2014, pp. 1-8

[3] R. Markussen, L. Ronquillo and C. SchÃ¼rmann. Trust in internet election observing the Norwegian decryption and counting ceremony. 2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), Lochau / Bregenz, Austria, 2014, pp. 1-8.

[4] Dyachkova, Irina and Rakitskiy, Anton. Development of the Remote Anonimous Voting System and the Analysis of its Vulnerabilities. 2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), 2021, pp. 485-487.

[5] Varshney, Karishma and Johari, Rahul and Ujjwal, R. L. Remote Online Voting Systemusing Aneka Platform. 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018, pp. 401-406.

[6] Alamleh, Hosam and AlQahtani, Ali Abdullah S. Analysis of the Design Requirements for Remote Internet-Based E-Voting Systems. 2021 IEEE World AI IoT Congress (AIIoT), 2021, pp. 0386-0390.

[7] Sliusar, Valentin and Fyodorov, Aleksei and Volkov, Aleksandr and Fyodorov, Pyotr and Pascari, Vladislav. Blockchain Technology Application for Electronic Voting Sys-

tems. 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 2021, pp. 2257-2261.

[8] Schmidt, Axel and Langer, Lucie and Buchmann, Johannes and Volkamer, Melanie. Specification of a Voting Service Provider. 2021 2009 First International Workshop on Requirements Engineering for e-Voting Systems, 2009, pp. 9-18.

[9] Ruth, Stephen and Mercer, David. Voting from the Home or Office? Don't Hold Your Breath. IEEE Internet Computing, 2007, volume 11, number 4, pp. 68-71.

[10] Kate, Nilam and Katti, J.V. Security of remote voting system based on Visual Cryptography and SHA. 2016 International Conference on Computing Communication Control and automation (ICCUBEA), 2016, pp. 1-6.

[11] Yang, Jun and Jing, Siyuan and Jia, Liping. RVBT: A Remote Voting Scheme Based on Three-Ballot. 2020 16th International Conference on Computational Intelligence and Security (CIS), 2020, pp. 303-307.

[12] Agarwal, Samarth and Haider, Afreen and Jamwal, Abhishek and Dev, Param and Chandel, Rajeevan. Biometric Based Secured Remote Electronic Voting System. 2020 7th International Conference on Smart Structures and Systems (ICSSS), 2020, pp. 1-5.

[13] Foster, David and Stapleton, Laura and Fu, Huirong. Secure Remote Electronic Voting. 2006 IEEE International Conference on Electro/Information Technology, 2006, pp. 591-596.

[14] Shirazi, Fateme and Neumann, Stephan and Ciolacu, Ines and Volkamer, Melanie. Robust electronic voting: Introducing robustness in Civitas. 2011 International Workshop on Requirements Engineering for Electronic Voting Systems, 2011, pp. 47-55.

[15] Usmani, Z.A. and Patanwala, Kaif and Panigrahi, Mukesh and Nair, Ajay. Multi-purpose platform independent online voting system. 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017, pp. 1-5.

[16] Lambrinoudakis, C. and Kokolakis, S. and Karyda, M. and Tsoumas, V. and Gritzalis, D. and Katsikas, S. Electronic voting systems: security implications of the administrative workflow. 14th International Workshop on Database and Expert Systems Applications, 2003. Proceedings., 2003, pp. 467-471.

[17] Rodiana, Irham Mulkan and Rahardjo, Budi and W., Aciek Ida. Design of a Public Key Infrastructure-based Single Ballot E-Voting System. 2018 International Conference on Information Technology Systems and Innovation (ICITSI), 2018, pp. 6-9.

[18] P, Annar Shankar and S, Harsshanth and V N, Jaswanth Solai and A P, Geetha and A, Sree Haran and Jemima D, Darling. Migrant Voting System-Solution to Gain the Lost Votes. 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 770-774.

[19] Cetinkaya, Orhan. Analysis of Security Requirements for Cryptographic Voting Protocols (Extended Abstract). 2008 Third International Conference on Availability, Reliability and Security, 2008, pp. 1451-1456.

[20] Kumar, Mahender and Chand, Satish and Katti, C. P. A Secure End-to-End Verifiable Internet-Voting System Using Identity-Based Blind Signature. Journal: IEEE Systems Journal, 2020, volume 14, number 2, pp. 2032-2041.

[21] Li, Yannan and Susilo, Willy and Yang, Guomin and Yu, Yong and Liu, Dongxi and Du, Xiaojiang and Guizani, Mohsen. A Blockchain-Based Self-Tallying Voting Protocol in Decentralized IoT. Journal: IEEE Transactions on Dependable and Secure Computing, 2022, volume 19, number 1, pp. 119-130.

[22] Salman, Wasan and Yakovlev, Viktor and Alani, Sameer. Analysis of the traditional voting system and transition to the online voting system in the republic of Iraq. 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2021, pp. 1-5.

[23] Al-madani, Ali Mansour and Gaikwad, Ashok T. and Mahale, Vivek and Ahmed, Zeyad A.T. Decentralized E-voting system based on Smart Contract by using Blockchain Technology. 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), 2020, pp. 176-180.

[24] Rosasooria, Yamuna and Mahamad, Abd Kadir and Saon, Sharifah and Isa, Mohd Anuar Mat and Yamaguchi, Shingo and Ahmadon, Mohd Anuaruddin. E-Voting on Blockchain using Solidity Language. 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE), 2020, pp. 1-6.

[25] Kiayias, Aggelos and Korman, Michael and Walluck, David. An Internet Voting System Supporting User Privacy. 2006 22nd Annual Computer Security Applications Conference (ACSAC'06), 2006, pp. 165-174.

[26] Kumar, Durgesh and Kumar Dwivedi, Rajendra. Blockchain and Internet of Things (IoT) Enabled Smart E-Voting System. 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), 2023, pp. 28-34.

[27] Jangada, Atharva and Dadlani, Nimish and Raina, Sanchit and Sooraj, VS and Buchade, A.R. De-Centralized Voting System using Blockchain. 2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), 2022, pp. 1-5.

[28] Peter, Geno and Stonier, Albert Alexander and Sherine, Anli. Development of Mobile Application For E-Voting System Using 3-step Security for preventing phishing attack. 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2022, pp. 1173-1177.

[29] Pranitha, G. and Rukmini, T. and Shankar, T. N. and Sah, Basant and Kumar, Naween and Padhy, Sasmita. Utilization of Blockchain in E-Voting System. 2022 2nd International Conference on Intelligent Technologies (CONIT), 2022, pp. 1-5.

[30] Dandekar, DB and Umratkar, Girish and Manwar, Pranali and Sute, Sneha and Ansari, Zain and Khan, Hamza and Ansari, Baseer and Joshi, Bhuvanesh. Online Voting System Using Cloud Computing. Journal: International Research Journal of Modernization in Engineering Technology and Science, 2022, volume 4, number 4.

[31] Culnane, Chris and Essex, Aleksander and Lewis, Sarah Jamie and Pereira, Olivier and Teague, Vanessa. Knights and Knaves Run Elections: Internet Voting and Undetectable Electoral Fraud. Journal: IEEE Security & Privacy, 2019, volume 17, number 4, pp. 62-70.

[32] Lauer, Thomas W. The Risk of e-Voting. Journal: Electronic Journal of E-government, 2004, volume 2, number 3, pp. 169-178.

[33] Jefferson, David and Rubin, Aviel D. and Simons, Barbara and Wagner, David. Analyzing Internet Voting Security. Journal: Commun. ACM, 2004, volume 47, number 10, pp. 59-64.

[34] Schryen, G. Security aspects of Internet voting. 37th Annual Hawaii International Conference on System Sciences.

[35] O. Kulyk, S. Neumann, M. Volkamer, C. Feier and T. Koster. Electronic voting with fully distributed trust and maximized flexibility regarding ballot design. 2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), Lochau / Bregenz, Austria, 2014, pp. 1-10.

[36] Stanley Kelley, Jr. Interpreting Elections. ISBN: 9780691022161. pp. 10-11, 1983, Publisher: Princeton University Press.

[37] Eilertsen, Anna Maria and Bagge, Anya Helene. Exploring API/Client Co-Evolution," 2018 IEEE/ACM 2nd International Workshop on API Usage and Evolution (WAPI). Gothenburg, Sweden, 2018, pp. 10-13.

[38] Valentin, Bojinov. RESTful Web API Design with Node.js - Second Edition. ISBN: 9781786469137. pp. 6-17, 2016, Publisher: Packt Publishing.

[39] Roman, Rodrigo and Alcaraz, Cristina and Lopez, Javier and Sklavos, Nicolas. Key management systems for sensor networks in the context of the Internet of Things. Journal: Computers & Electrical Engineering, 2011, volume 37, number 2, pp. 147-159. Publisher: Elsevier.

[40] Gupta, Satinder Bal and Mittal, Aditya. Introduction to Database Management System, second edition. ISBN: 9789381159316. pp. 3-4, 2017, Publisher: Laxmi Publications Pvt Ltd.

[41] Lena, Wiese. Advanced Data Management : For SQL, NoSQL, Cloud and Distributed Databases. ISBN: 9783110441406. pp. 17-26, 2015, Publisher: De Gruyter.

[42] Lena, Wiese. Advanced Data Management : For SQL, NoSQL, Cloud and Distributed Databases. ISBN: 9783110441406. pp. 33-39, 2015, Publisher: De Gruyter.

[43] Microsoft. Cipher Suites in TLS/SSL (Schannel SSP). URI: https://learn.microsoft.com/en-au/windows/win32/secauthn/cipher-suites-in-schannel. 2023

[44] Oppliger, Rolf. SSL and TLS : Theory and Practice. ISBN: 9781596934474. pp. 133-134, 2009, Publisher: Artech House, Inc.

[45] Oppliger, Rolf. SSL and TLS : Theory and Practice. ISBN: 9781596934474. pp. 94-120, 2009, Publisher: Artech House, Inc.

[46] Oppliger, Rolf. SSL and TLS : Theory and Practice. ISBN: 9781596934474. pp. 135-140, 2009, Publisher: Artech House, Inc.

[47] I., Blake and G., Seroussi and N., Smart. Elliptic Curves in Cryptography. pp. 11-100, 1999, Publisher: Cambridge University Press.

[48] Bos, Joppe W and Halderman, J Alex and Heninger, Nadia and Moore, Jonathan and Naehrig, Michael and Wustrow, Eric. Financial Cryptography and Data Security: 18th International Conference. pp. 157-175, 2014.

[49] Sankar, R. and Subashri, T. and Vaidehi, V. Implementation and integration of efficient ECDH key exchanging mechanism in software based VoIP network. pp. 124-128, 2011.

[50] Dang, Q Secure hash standard. URI: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. pp. 10-17, 2015. NIST Special Publication.

[51] Kodali, Ravi Kishore. "Implementation of ECDSA in WSN," 2013 International Conference on Control Communication and Computing (ICCC). pp. 310-314, 2013.

[52] Cid, C. and Murphy, S. and Robshaw, M. Algebraic Aspects of the Advanced Encryption Standard. ISBN: 9780387243634. pp. 35-42, 2006. Publisher: Springer New York.

[53] Dworkin, M. A Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. URI: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf. pp. 7-27, 2007. NIST Special Publication.

[54] Duong Le, Vu Trung and Tran, Thi Hong and Pham, Hoai Luan and Lam, Duc Khai and Nakashima, Yasuhiko. MRSA: A High-Efficiency Multi ROMix Scrypt Accelerator for Cryptocurrency Mining and Data Security. pp. 168383-168396, 2021. Journal: IEEE Access.

[55] Pavani, K. and Sriramya, P. "Enhancing Public Key Cryptography using RSA, RSA-CRT and N-Prime RSA with Multiple Keys," 2021 Third International Conference on

Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). pp. 1-6, 2021.

[56] Bellare, Mihir and Canetti, Ran and Krawczyk, Hugo. Message authentication using hash functions: The HMAC construction, vol. 2. pp. 12-15, 1996. Journal: RSA Laboratories CryptoBytes.

[57] Adam, Stenly Ibrahim and Moedjahedy, Jimmy H and Maramis, Jeremiah. "RESTful Web Service Implementation on Unklab Information System Using JSON Web Token (JWT)," 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS) pp. 1-6, 2020.

[58] Sadeghian, Amirmohammad and Zamani, Mazdak and Abdullah, Shahidan M. "A Taxonomy of SQL Injection Attacks," 2013 International Conference on Informatics and Creative Multimedia. pp. 269-273, 2013.

[59] Chao-yang, Zhang. "DOS Attack Analysis and Study of New Measures to Prevent," 2011 International Conference on Intelligence Science and Information Engineering. pp. 426-429, 2011.

[60] Soliman, Mostafa and Azer, Marianne A. "Web Application API Blind Denial of Service Attacks," 2018 14th International Computer Engineering Conference (ICENCO). pp. 249-253, 2018.

[61] Arshad, Elham and others. Analysis of Oauth and CORS vulnerabilities in the wild. pp. 25-39, 2022. Publisher: Università degli studi di Trento

[62] Khanpara, Pimal and Patel, Shivam and Valiveti, Sharada. "Blockchain-based E-Voting Technology: Opportunities and Challenges," 2022 7th International Conference on Communication and Electronics Systems (ICCES). pp. 855-861, 2022.