Remote Secure Online Voting System Development

T. Matos[0009-0008-2665-4332] and J. Guerreiro[1][0000-0001-7471-4928]

NOVA LINCS & FCT & ISE , University of the Algarve, 8005-139 Faro, Portugal
a52888@ualg.pt jdguerreiro@ualg.pt

Abstract Remote online elections have been studied and applied throughout the years in several countries with different approaches and distinct secure mechanisms. Online elections collect sensitive information therefore, the system, must guarantee confidentiality, integrity, privacy, security and be auditable to ensure a successful and creditable election process.
A distinct web-based system approach, as far known, was developed using data processing, backend, storage and frontend components secured by cipher suites and an independent developed asymmetric cryptography key management system. Both, cipher suites and the key manage system, ensure authentication, authorization and vote encryption before storage, making it impossible to visualize election results primary to vote counting initiation and user's votes choices.
The developed system implements a mechanism to verify enabled users accesses in a specific election and ensure if they already voted, avoiding repeated votes. Encrypts, stores votes and users in distinct database tables to avoid, even with direct database access, the knowledge of who has voted so far and what was the vote direction, displaying statistics information of the already casted number of votes, being, the election committee, able to monitor effectiveness, before the voting period ends.
The developed and implemented databases follows ACID (Atomicity, Consistency, Isolation and Durability) properties, assuring each statement in a transaction is treated as a single unit preventing data loss and corruption, ensuring consistency, isolating user transactions, guaranteeing no transaction interferes with another and, on the event of a system failure, is assured that all executed transactions are saved and no data is lost.

Keywords: Security, Cibersecurity, Remote Online Voting, System Development

1 Introduction

Elections are a key point in any democracy allowing the people to individually choose their representatives. Nowadays, elections are, in every democratic country, used to select the ruling or regional governments for a mandate. Paper elections are the most common ones where every voter selects

a candidate from a ballot and places the vote paper into an urn. Votes are never associated with the voter and results are counted and shown to the population.
Recently, different forms of election have been addressed, elections via letter using post offices that deliver votes to the counting tables or the use of electronic elections for voters to submit their choices, by either using a predefined machine set and localization, similar to paper elections, or by using remote election software and internet technologies.
A remote elections software must be reliable, secure and ensure that user's identity and the privacy of their vote choices are not compromised, therefore cryptography and secured communications assume an relevant importance.

In this article, integrated on the Accessible security and privacy topic, a research study and a remote election software was developed that can be used in any type of elections, following the best security and privacy practices to ensure voters can cast their votes without being compromised and the election follows a predefined secure workflow until the results are made public.

The systems requirements to properly define the system architecture were [1] [2]:

- **Ballot flexibility**: The system must be able to have different type of ballots, from yes or no to multiple parties ballots;
- **Voter flexibility**: Must be able to have different list of eligible voters for distinct elections;
- **Spontaneity**: Must require as little preparation as possible;
- **Remote participation**: Possibility to cast a vote from wherever the voter chooses;
- **Usability**: The system must be usable for everybody;
- **Strict Voting**: Cast votes only on the ballot candidates or by not selecting any candidate, resulting in a blank vote;
- **logging**: There must be registered log information without compromising individual privacy for the system to be auditable.

An election is a sensitive procedure that differs in candidates and eligible voters on each election, therefore several issues were considered for the system development:

- **Eligibility**: Only eligible voters are accepted;
- **Uniqueness**: Only a single vote can be accepted from each voter;
- **Equality**: Every eligible voter can access the election results after the results are public, regardless if the voter casted or not a vote;
- **Secrecy**: The system cannot link a voter to his vote, even with direct access to the database, nevertheless the voter must only vote once;
- **Integrity**: Impossibility to replace a casted vote with another vote;

– **Robustness**: The system must always perform its functions regardless of any eventual problem.

By following all these requirements, the system will successfully conduct a remote election.

The remainder of this article is organized as follows: In Section 2 Related work is presented. Section 3 describes the System Architecture. The developed Application Programmable Interface (API) Module is presented in Subsection 3.1, the databases are presented in Subsection 3.2 while Section 4 presents the System Security and Section 5 concludes the article.

2 Related Work

Throughout the years election systems have been used and are a reality in several countries. Switzerland [3], Estonia [4] and Norway [5] are perfect examples of such use.
Dyachkova et al [6] developed a completely anonymous real time public network voting system. The authors system uses two different user's permissions, regular users and administrators. The first can cast a vote and visualize results after they became public, while the administrators cannot vote, but can create, delete elections, define users and groups, and publish results. Users authenticate using a single password and for the ballot creation and for signature the authors used an algorithm named "blind", based on RSA encryption. The ring signature protocol was used for data transmission to protect and anonymize eligible voters and prevent repeated votes. The system architecture is a node network connected to each other and a storage server that receives the nodes data, maintaining voter's anonymity.
Varshney et al [7] proposed a system where the voter registers and authenticates identifying themselves with an random id and password generated by the system after a form fulfillment. On the election day, voters may cast their vote within a 10 second window to submit their choice. After a successful submission, a numeric code is automatically generated to identify the voter's voting location and choice which is encrypted by AES and stored on the database. Due to the AES complexity, even with a smaller key size of 128 bits, the generated numeric code maintains its confidentiality. The presented system does not allow, after the vote is casted, the same user to access the ballot again. To ensure integrity, any election system must be auditable, Alamleh et al [8] mentioned. The authors also refereed that an election system must be stored in a safe environment to maintain confidentiality, integrity, availability and also logs must be made available for eventual audits.
The Sliusar et al [9] proposed system is based on Blockchain technology. The system requires an identification confirmation, where the user presents a document and automatically the system generates a QR code, which is verified and registered by an authorized independent third party operator into

the Blockchain, followed by the a token generation that will function as a ballot associated to the user, like a virtual wallet with a time limit. When the user casts his vote, a transaction between the user's wallet and the candidate's wallet is established, maintaining anonymity, because it cannot be traced back to the user's identity. The vote is stored and the user can visualize his vote at any time. The Blockchain technology guarantees transparency, anonymity, reliability, and security due to the fact that there is no need for electoral authorities, only the voter is able to see his vote and it is impossible to hack a chain block without affecting all others, avoiding, therefore, eventual vote tampering. Blockchain, with its a decentralized technology, assures fault tolerance, increases efficiency and faster processing, increasing civic engagement where the voter can participate in an election regardless of his location. The only problem of Sliusar et al system is the need of guaranteeing the voter's identification assuring the voter is indeed the same person of the identification document.

Schmidt et al [10] state that an election system should be outsourced to verified third parties called Voting Service Providers (VSP). A VSP should provide secure building, hardware, software and skilled personnel to ensure a secure electronic election. The authors refer that a VSP must be trustworthy, regulated and supervised by proper authorities. The hosting and operational processing are provided by the VSP therefore they must also secure communications via SSL/TLS. Some servers are connected to the external unprotected network (internet) to receive the votes and other are not, therefore it is required the implementation of all trusted and secured components like communication, storage and erasure, cryptography, logging and monitoring, installation, configuration and personnel for a proper electronic election to be held, maintaining system availability and protection.

An evaluation of remote internet voting as the next step in governance was performed by Ruth et al [11]. The authors stated that the challenge was not technological but political, despite handling problems like data security, password protection, anonymity and server overload. The constituencies representation, discrimination and other political issues are concerns to be dealed by public officials. The authors also mentioned that remote voting elections were held in the United States and in Europe. In the last case, countries noticed that voters who normally describe themselves as abstainers choose internet voting. Different identification ways were described by the authors, in the USA, voters used their social security number, in Spain a smart card with a PIN code was used but it was counterproductive because only a small number of voters used the system. In Estonia a digital identification with two PINs was implemented where the first was used as a credential to access and cast the ballot, the second to encrypt and store it into the server. Voter anonymity was kept using "double envelope" security in which the digital ballot is not decrypted until the voter's identity was separated from it. Several Estonian parties expressed concerns over the system usage, especially regarding vote secrecy and, due to the unsupervised

nature of the system, concerns of potentiate illegal pressure, coercion, or inducement of voters. Kate el al also mentioned that the Estonia System did not meet the expectations and only a small percentage of electors used the system and stated that these type of systems are convenient and can be secured in maintaining the voters anonymity but the assurance that votes are genuinely free of choice is a problem that does not occur in physical regular paper elections.

Kate et al [12] used a message digest embedded in a random image file as an authentication system. The message digest used a SHA2 hashing algorithm with a generated 256 bit string and the vote was encrypted with AES protocol and then stored. When elections terminate, the votes are decrypted and results are announced. Yang et al [13] implemented a complex three ballot model for end-to-end verifiability where every voter holds one of the ballots that can be used to verify the bulletin board results, calculate the final result and compare it with the published results. The system is resistant to coercion and does not require any cryptography while votes are being casted, they are encrypted and signed when stored. On tallying, the signature is verified and the vote is decrypted and counted.

The use of biometric fingerprint to authenticate voters was presented by Agarwal et al [14] in their developed system. The reading of the biometrical is physical, so the voters address themselves to a voting station, and a comparison between the stored and the voter fingerprint is done by hashing. The votes are not encrypted and only the vote casting is stored to prevent vote duplication.

Annar Shankar et al [15] proposed a system for migrants to cast votes, consisting in three modules: a mobile application, a backend server and a web application. The mobile application verifies the voter's ID requiring a One Time Password (OTP). Voters can temporarily change their voting location, the system requests a voter photo, a digital copy of its ID and the new location. Voters can also verify their activity by scanning an encrypted QR code present on the web application in the voting station. The backend server is responsible for storing the voter personal details in a database and respond to voter requests. Generating parties list and counting votes are also another responsibility of the backend server. On election day the voter scans a QR code on the website and sends their details to the server, triggering an event on the voting location. The web application on the polling station listens to the trigger and displays the constituency list allowing the voter to vote which is then stored anonymously by encrypting it 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 face is detected the vote buttons are enabled. After a vote has been casted it is encrypted and sent to a distributed server to avoid data tampering. Each voter must be present on election booth to cast the vote to avoid forgery.

All of these systems have pros and cons, the encryption before storing makes

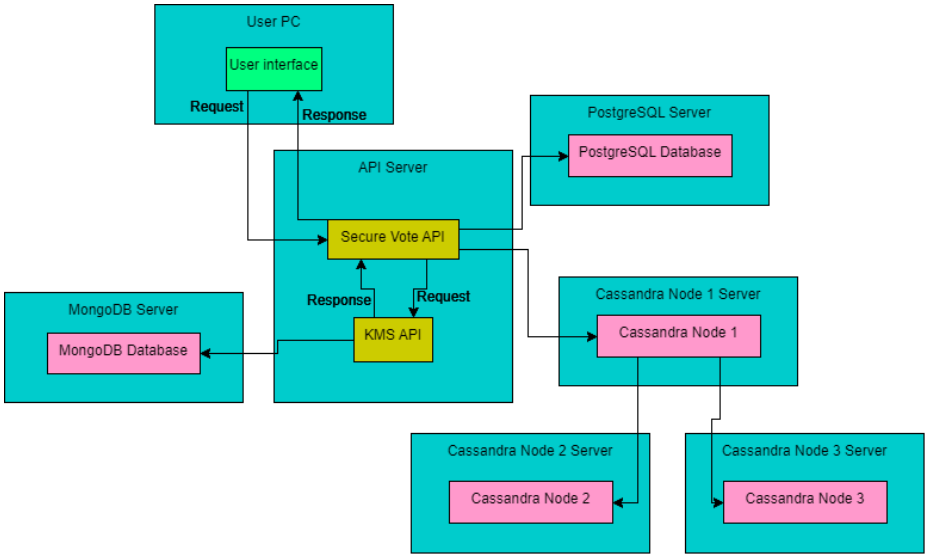

Figure 1: System Architecture.

it harder to change data, using the decentralized Blockchain functionalities is a good option to guarantee security and and anonymity, using a VSP may also be a good solution, nevertheless some counter backs in the presented systems like the obligation to physically access the voting system, or to physically authenticate to vote, or even having a small windows to select the voter choice may be seen as an issue.
The developed Remote Secure Online Voting system presented in this article is a different approach contributing to remote election systems that can be applied to any kind of elections assuring security and anonymity. In the next Section the developed System Architecture is presented.

3 System Architecture

The system architecture is presented in Figure 1, composed by an user interface, a Application Programming Interface (API) module with two API, Secure Vote and the Key Management System (KMS), a relational and two non-relational databases, one using Cassandra database system distributed nodes and mongoDB. Figure 1 diagram shows the APIs and servers usage for each used database system. Taking advantage of the decentralized nature of Cassandra the server usage of every node is optimal and if a crash or Denial of Service(DOS)/Distributed Denial of Service(DDoS) attacks occurs all will remain working. MongoDB and PostgreSQL separate servers mitigates possible data compromises, isolating such attempts to specific

servers.
An election system requires a logical component to process data [16] [17], so the development of a API module was essential to be used as an interface between users, databases storage and the Cassandra nodes in order to secure data. In subsection 3.1 the API module will be detailed while the databases will be presented in subsection 3.2.

3.1 Application Programming Interface Module

As stated before, an API is a interface to access distinct software modules, supplying end-points to invoke a specific method or class [18]. The most used API model is the Representational State Transfer, also known as REST, which was the logical choice for the developed remote election system in order to access the electoral module in a indirect, secured and encrypted way. Two distinct API were developed in this module, the first, Secure Vote API, that allows the creation of an election, vote casting, vote tallying and results presentation. All accesses and operations are stored in a log for eventual auditing. Each user has dedicated endpoints, forms and permissions. The system defines a set of permissions, Regular for voters, Manager to create elections, ballots, initiate and end election periods, tally and display results, Auditor to visualize all logs and audit the system and the last, Admin, accesses all system options.

The system also defines, for each election, a set of data entities, defined and properly mapped to their properties [19] [20] [21] [16] [22] [17] [23] [24] [25], each with a distinct role:

- **User**: The user class contains all operations regarding users like user registration, account verification, update and delete, login, password recovery, regenerate signature key, change permission, block and unblock user, blacklist, user verification. All of these options depend on the user permissions previously described;
- **Election**: This class has operations to create, update and delete elections, generate and regenerate the election keys that are responsible for vote casting encryption and decryption. The generation and regeneration of election key are only available before the start date/time of the election. Every user can visualize the results, after the election end date, just managers can access the voters list but not on which candidate or group of candidates they have voted on;
- **Vote**: The vote class has only one operation, the vote itself. The votes are directly associated to an election but never to a user, so no one can visualize, not even in the database, the users vote. Tallying is also the vote class responsibility, counting each candidate or group of candidates votes;
- **Log**: All interactions in the system are stored in two types of logs, internal and election logs. The first, auditors can visualize accesses, admins and system actions, while the second auditors can visualize all

election actions performed by all users while the election period is available;

– **Blacklist**: This class supplies operations to add and remove emails to and from the blacklist. The blacklist blocks emails from registering an account in order to block eventual email spam.

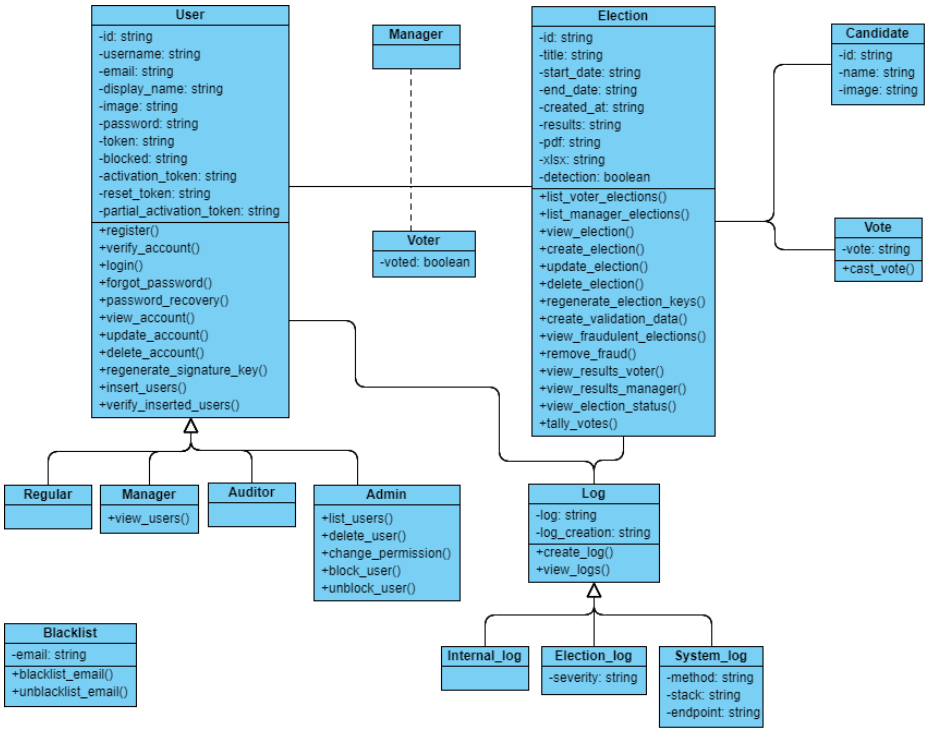

Figure 2: Secure Vote module API Class Diagram.

The second API is KMS, developed to secure encryption keys since they are a major concern and cannot be compromised [1]. A key management system allows its users to manage their encryption keys. Users can create, retrieve, update and delete keys [26]. 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 case, the KMS will have its own API to allow the Secure Vote API to create, retrieve, update and delete keys regarding elections and user signatures. Due to the sensitive nature of the private keys, they are encrypted when stored while the public keys do not require such encryption.

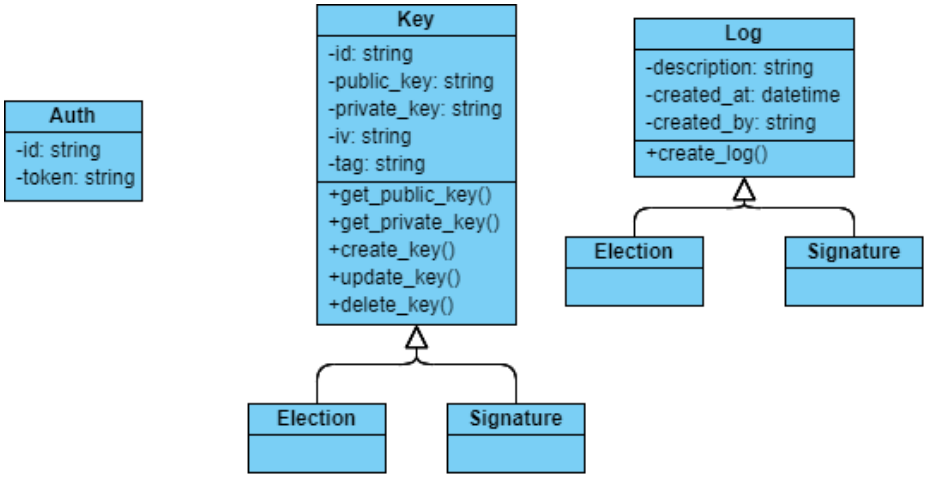

Figure 3: KMS API Class Diagram.

In Figure 3 it is possible to visualize the lack of relationships between entities because they are not required to properly execute their operations. The KMS API main functionality is to secure the public and private keys and operations, like key creation, retrieval, update or delete. This API also stores logs of all actions performed between the Secure Vote API, the non-relational database and the KMS API. The KMS API has its own entities and operations:

- **Key**: The Secure Vote API uses the KMS API to create, retrieve, update and delete keys. Voter signature keys are used while casting the vote and election keys are used to encrypt and decrypt votes;
- **Log**: This class stores all actions performed by the system to be audit and checked if any unauthorized access occurred;
- **Auth**: The auth class has a middle-ware in all endpoints checking if the token exists and if is a valid one. Each token is generated with a secret associated to each system. If one of these tokens is compromised, the keys cannot be retrieved because they are encrypted and also of because the API answer is encrypted before it is sent.

3.2 Databases

The databases architecture are a division in two of the diagram presented in Figure 2, a relational and non-relational databases for data decentralization ans security.
The relational database was implemented to deal with data manipulation on multiples tables and consistency also because of the transactions amount

needed to ensure candidates, voters, managers, election data storage. PostgreSQL was chosen because is ACID-complaint features and automatic updates to stored procedures preventing therefore SQL injection. Figure 4 describe the schema used for the Secure Vote user, candidate, manager, voters and elections data, while Figure 5 describe the Secure Vote non-relational data schema where the votes and logs are stored.

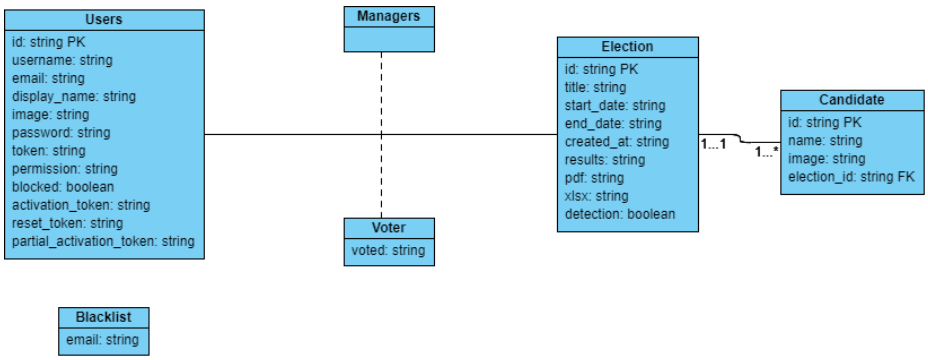

Figure 4: Secure Vote module relational data schema.

Votes and log operations, performed by both API, cannot fail during vote casting, therefore the databases must be, always available. To mitigate this problem, Cassandra database system was implemented because of its distributed NoSQL wide-column storage system developed by Apache for high availability, and its architecture that does not present a single point of failure, so far. Also because Cassandra prepared statements prevents NoSQL injection attacks, assigning extra security to the election system. However, Cassandra is not ACID-complaint, due to the lack of transactions, but the implemented operations do not require any transactions, they are single not sequenced queries.

The non-relational data schema presented in Figure 5 was implemented in Cassandra distributed database system for the votes of the electors, internal operations log, election transactions log and system logs. The votes were implemented in the non-relational database without any identification of the voter, so it cannot be back traced which voter voted on what candidate, not even with database access, because there is no registration of the user, nevertheless the user cannot vote a second time, managed in the transaction database, assuring voter's anonymity and confidentiality.
Comparing the diagram presented in Figure 3.1 with the schema in Figure 4, some attributes are no present because permissions have to be stored with the users data. Several tokens are also stored, a reset token is used when a user recovers its password and a activation token is used to verify

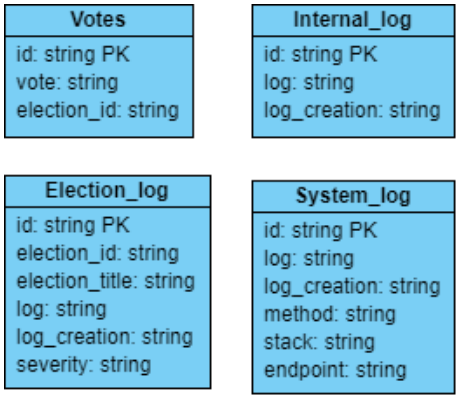

Figure 5: Secure Vote module non-relational data schema.

if the submitted email is actually a real user email, accomplished with a activation email sent to the registered user email.
The KMS database is stored in MongoDB because is schemeless, where data can change the schema without having to alter the database itself, allowing keys to be stored regardless of what encryption type is used.

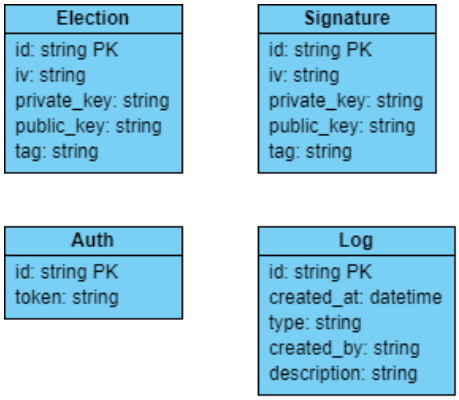

Figure 6: KMS data schema.

Figure 6 represents the developed schema to store KMS operations where each table is a key type to improve data organization. All election, signatures, authentication keys and all transactions logs performed are stored in the schema.

All data must be secured, therefore in section 4 all applied system security is detailed.

4 System Security

Nowadays there are several cipher suites that can be applied to secure applications, supplying algorithms for key exchange to protect shared keys, bulk encryption to encrypt client/server messages and message authentication to verify integrity, combined with TLS or SSL protocols [27]. Remote elections must be secured [28], therefore a
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_RSA_4096_SHA256_K571
cipher suite was used for both Secure Vote and KMS APIs, where the algorithms ECDHE was applied for key exchange, ECDSA for signature verification, AES_256_GCM and RSA_4096 are the bulk encryption algorithms used, HMAC is for message authentication and K571 is the elliptic curve used. The cipher suite used was combined with TLS over TCP for secure transport connections.

4.1 Cipher Suite implementation in the Remote Secure Online System

Secure Vote and KMS uses ECDHE together with AES_256_GCM to encrypt the data exchanged between themselves where every API communication has always a different encryption keys. For signing votes and integrity check, Secure Vote API uses SHA_256 together with ECDSA to verify if a vote cast belong to the actual voter who casted it. If the vote was forged or tampered, is discarded, but the attempt to submit a forged or tempered vote is logged for auditors and authorities to confirm and eventually act.
AES_256_GCM is used encrypt the signature and election private keys to be stored in the KMS database. The election results also use this algorithm combined with ECDHE communication encryption. The 256 bit key was chosen instead of 128 bit key because private keys are sensitive information and GCM mode of operation was due to its encrypted data integrity capabilities for communication and private keys security. To encrypt and decrypt using AES, a 256 bit key must be supplied, therefore the Key Derivation Function (KDF) was applied to transform a string into another string of a predetermined size. The function was chosen because is particularly resistant against ASIC and brute-force attacks to decipher encrypted messages and passwords [29]. The KDF or scrypt was applied when creating signature or election keys, where the user must insert a 12-character key, transformed by the function in a 256 bit or 32 characters long string to be used with AES. RSA is used in Secure Vote to encrypt the vote casting, using on the client side, the public key, and on the server side a private key to decrypt the votes while HMAC is used to provide a way to verify if the submitted cast is the same that arrives in the API. ECDSA already provides a

way to verify integrity, but HMAC reinforces these transactions because it is a sensitive process and must be handled with special care to avoid tampered votes.

4.2 Complementary security applied in the Remote Secure Online System

To identify and authenticate both users and services consumed by the Secure Vote API [30] [31] a JSON Web Token (JWT), which is a token represented as a string, converted from a JSON object, to digitally sign using HMAC, was used. These tokens in the Secure Vote API are used to verify permissions in each endpoint checking on the database if a user or service has permissions to access. KMS API also uses JWT to authenticate requests while always requiring authentication in all its endpoints.
Stored procedures and parameterized statements, which are not mutually exclusive and can be both used at the same time [32], were also used in the system to mitigate possible SQL and NoSQL injection attacks.
DOS attacks are one of the biggest threats in the internet nowadays, API endpoints can be subjected to such attacks, and an attacker can send constantly requests [30] [31] therefore a rate limiter that defines the maximum number of IP address requests is recommended. Both API, Secure Vote and KMS uses Express.js framework that integrates a rate limiter on the endpoints to mitigate this kind of attacks.
To deal with Cross Site Request Forgery (CSRF), exploitation of the existing trust between application and browser, when requests arrive on an API from dubious sources, an extension of the XML-HttpRequest, Cross Origin Resource Sharing (CORS) is applied, providing permissions to access resources in cross origin requests through a set of HTTP headers enforced by the client. The validation of these headers is performed in the server site. Both API, Secure Vote and KMS uses these mechanisms to control access and prevent CSRF attacks.

4.3 Resilient and Security Tests

The Remote Secure Online Voting System is now being tested by security personal in the University of Algarve, where several tests have been applied. Tests like brute force, ASIC and dictionary attacks have been performed and as far, unsuccessful. Tools like DirBuster, Dirb, Webroot.pl and Burp Suite have been unsuccessfully used for brute force attacks. DOS and DDOS attacks have been performed with tools like LOIC, HOIC, Slowloris and RUDY. Despite of the attacks diminish communications, the system never stopped and still was able to continue working. Several other tests are being prepared to be enforced on the system to test the resiliency and integrity.

5 Conclusions and Future Work

All democratic democracies need elections for people free choice to select their rulers. So far, several attempts have been made to implement electronic, digital, and remote election systems, but not all of them were successful. One of the main issues is public trust, so the authors intended to contribute by providing a possible solution that can be applied in remote secure online elections. The design architecture, ciphers, redundancy, and complementary security applied on the system ensures that a possible solution can be used in remote elections, providing integrity, data segregation, confidentiality, privacy and secure transactions. The system is completely auditable, all transactions, internal, external, election and system logs are saved and accessible. Several security tests and cyberattacks have been made and yet unsuccessful, others will follow, as future work, to ensure that the product is secure to be on production.

ACKNOWLEDGEMENTS

This work was supported by NOVA LINCS (UIDB/04516/2020) with the finantial support of FCT.IP.

References

1. A. Kiayias, M. Korman, and D. Walluck, “An internet voting system supporting user privacy,” in *2006 22nd Annual Computer Security Applications Conference (ACSAC’06)*, IEEE, Dec. 2006.
2. O. Kulyk, S. Neumann, M. Volkamer, C. Feier, and T. Koster, “Electronic voting with fully distributed trust and maximized flexibility regarding ballot design,” in *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, IEEE, Oct. 2014.
3. U. Serdult, M. Germann, F. Mendez, A. Portenier, and C. Wellig, “Fifteen years of internet voting in switzerland [history, governance and use],” in *2015 Second International Conference on eDemocracy; eGovernment (ICEDEG)*, IEEE, Apr. 2015.
4. S. Heiberg and J. Willemson, “Verifiable internet voting in estonia,” in *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, IEEE, Oct. 2014.
5. R. Markussen, L. Ronquillo, and C. Schürmann, “Trust in internet election observing the norwegian decryption and counting ceremony,” in *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, IEEE, Oct. 2014.
6. I. Dyachkova and A. Rakitskiy, “Development of the remote anonimous voting system and the analysis of its vulnerabilities,” in *2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, IEEE, May 2021.
7. K. Varshney, R. Johari, and R. L. Ujjwal, “Remote online voting systemusing aneka platform,” in *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, IEEE, Aug. 2018.

8. H. Alamleh and A. A. S. AlQahtani, "Analysis of the design requirements for remote internet-based e-voting systems," in *2021 IEEE World AI IoT Congress (AIIoT)*, IEEE, May 2021.
9. V. Sliusar, A. Fyodorov, A. Volkov, P. Fyodorov, and V. Pascari, "Blockchain technology application for electronic voting systems," in *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, IEEE, Jan. 2021.
10. A. Schmidt, L. Langer, J. Buchmann, and M. Volkamer, "Specification of a voting service provider," in *2009 First International Workshop on Requirements Engineering for e-Voting Systems*, IEEE, Aug. 2009.
11. S. Ruth and D. Mercer, "Voting from the home or office? don't hold your breath," *IEEE Internet Computing*, vol. 11, pp. 68–71, July 2007.
12. N. Kate and J. Katti, "Security of remote voting system based on visual cryptography and sha," in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, IEEE, Aug. 2016.
13. J. Yang, S. Jing, and L. Jia, "Rvbt: A remote voting scheme based on three-ballot," in *2020 16th International Conference on Computational Intelligence and Security (CIS)*, IEEE, Nov. 2020.
14. S. Agarwal, A. Haider, A. Jamwal, P. Dev, and R. Chandel, "Biometric based secured remote electronic voting system," in *2020 7th International Conference on Smart Structures and Systems (ICSSS)*, IEEE, July 2020.
15. A. S. P, H. S, J. S. V N, G. A P, S. H. A, and D. Jemima D, "Migrant voting system-solution to gain the lost votes," in *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, IEEE, Oct. 2021.
16. Y. Rosasooria, A. K. Mahamad, S. Saon, M. A. M. Isa, S. Yamaguchi, and M. A. Ahmadon, "E-voting on blockchain using solidity language," in *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, IEEE, Oct. 2020.
17. A. Jangada, N. Dadlani, S. Raina, V. Sooraj, and A. Buchade, "De-centralized voting system using blockchain," in *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, IEEE, Sept. 2022.
18. A. M. Eilertsen and A. H. Bagge, "Exploring api/client co-evolution," in *2018 IEEE/ACM 2nd International Workshop on API Usage and Evolution (WAPI)*, pp. 10–13, 2018.
19. Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, X. Du, and M. Guizani, "A blockchain-based self-tallying voting protocol in decentralized iot," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, pp. 119–130, Jan. 2022.
20. W. Salman, V. Yakovlev, and S. Alani, "Analysis of the traditional voting system and transition to the online voting system in the republic of iraq," in *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, June 2021.
21. A. M. Al-madani, A. T. Gaikwad, V. Mahale, and Z. A. Ahmed, "Decentralized e-voting system based on smart contract by using blockchain technology," in *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, IEEE, Oct. 2020.
22. D. Kumar and R. Kumar Dwivedi, "Blockchain and internet of things (iot) enabled smart e-voting system," in *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, IEEE, Jan. 2023.

23. G. Peter, A. A. Stonier, and A. Sherine, “Development of mobile application for e-voting system using 3-step security for preventing phishing attack,” in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, IEEE, Apr. 2022.
24. G. Pranitha, T. Rukmini, T. N. Shankar, B. Sah, N. Kumar, and S. Padhy, “Utilization of blockchain in e-voting system,” in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, IEEE, June 2022.
25. G. U. P. M. S. S. Z. A. H. K. B. A. Dandekar, D. B. and B. Joshi., “Online voting system using cloud computing,” *International Research Journal of Modernization in Engineering*, 2022.
26. R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, “Key management systems for sensor networks in the context of the internet of things,” *Computers; Electrical Engineering*, vol. 37, pp. 147–159, Mar. 2011.
27. Microsoft, “Cipher suites in tls/ssl (schannel ssp).” URI:https://learn.microsoft.com/en-au/windows/win32/secauthn/ciphersuites-in-schannel, 2023.
28. G. Schryen, “Security aspects of internet voting,” in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, IEEE, 2004.
29. V. T. Duong Le, T. H. Tran, H. L. Pham, D. K. Lam, and Y. Nakashima, “Mrsa: A high-efficiency multi romix scrypt accelerator for cryptocurrency mining and data security,” *IEEE Access*, vol. 9, pp. 168383–168396, 2021.
30. T. W. Lauer, “The risk of e-voting,” 2004.
31. D. Jefferson, A. D. Rubin, B. Simons, and D. Wagner, “Analyzing internet voting security,” *Communications of the ACM*, vol. 47, pp. 59–64, Oct. 2004.
32. Z. Chao-yang, “Dos attack analysis and study of new measures to prevent,” in *2011 International Conference on Intelligence Science and Information Engineering*, IEEE, Aug. 2011.