Information Security Final Report

Theo Depraetere, Floris Kornelis van Dijken, Miel Peeters,
Gadir Suleymanli, Matisse Sulzer, Ruben Vandamme

April 2024

1 Introduction

In this report, we delve into various dimensions of information security concerning online voting systems. We begin by elucidating the fundamental requirements for fair elections, exploring the principles of authorization, accessibility, confidentiality, and transparency. Next, we compare the inherent properties of online systems with the status quo. Before describing technical solutions to the online voting problem, we then discuss the security services they must achieve, and the categories of attacks and vulnerabilities they need to be protected against. Then, we propose our design for an online voting system, and motivate our design choices along the way.

2 Requirements for Fair Elections

Every adult citizen has the right to vote. Someone who is not authorized to vote should not be able to vote either(1). Clear criteria should be in place for this. Every authorized person should be able to vote, without fear or intimidation, in an accessible and equal manner(2). A person may only vote once(3), and their vote should be treated equally compared to other votes; they carry the same weight. Secrecy of a vote is to be maintained(4). The voting process must be entirely transparent. The procedures should be public and accessible, and everyone should to some extent be familiar with them. Those responsible for the electoral process must have the qualifications and be impartial. The correctness of an election must, of course, be verifiable(3) [1].

(1) Authorization + authentication, (2) Availability, (3) Data integrity + non-repudiation, (4) Confidentiality

3 Traditional vs. Online Voting: Problems Associated with Online Elections

3.1 Security in Traditional Voting Systems

Traditional voting systems possess inherent security features due to their tangible nature. The physical components of these systems, such as ballot boxes and voting booths, are guarded and monitored, making unauthorized access and tampering difficult. This aspect of physical security is a significant advantage of traditional voting systems.

Moreover, traditional voting systems require voters to be physically present, allowing their identities to be verified using reliable, tangible means such as photo identification. This process significantly reduces the risk of voter impersonation, further enhancing the security of these systems.

Another crucial feature of traditional voting systems is the audit trail they leave behind. The physical trail, in the form of ballot papers, can be manually counted and verified in case of any disputes regarding the election results. This auditability is a critical aspect of ensuring the integrity of the voting process.

3.2 Challenges in Emulating Security in Online Voting Systems

While online voting systems offer the convenience of remote voting, they face significant challenges in emulating the security features inherent in traditional voting systems. The internet is fraught with various threats such as hacking, malware, and DDoS attacks. These threats can compromise the security of online voting systems, leading to unauthorized access, data breaches, and even manipulation of votes.

Ensuring voter anonymity and privacy is another significant challenge in online voting. While traditional voting allows voters to cast their votes in privacy, emulating this level of privacy online is difficult due to digital footprints. This challenge poses a significant hurdle in the widespread adoption of online voting systems.

In online voting, there's no physical trail that can be audited, making the verification of results challenging. Luckily, there is a way to simulate auditing with a digital voting system, which will be elaborated upon later in the report.

Lastly, due to the intangible nature of online systems and the prevalence of cyber threats, voters may find it hard to trust online voting systems. This lack of trust can impact voter turnout and the overall credibility of the election, posing a significant challenge to the implementation of secure and reliable online voting systems.

4 Required Security Services

Before discussing the security mechanisms, we will first address the required security services that they must achieve. From a research perspective, we have divided the security challenge of online voting into three subproblems: authentication, data transmission, and data storage. For each subproblem, we investigated the required security services. Considering the security policy for online voting, we discuss the necessary security services and how each of these services contributes to the robustness and trustworthiness of the online voting system.

4.1 Confidentiality

Confidentiality ensures that sensitive information, such as voter identities and choices, remains secret from unauthorized parties. Obviously, anonymity is a critical part of voting; a vote must remain private and untraceable. This requires that all transmitted data be encrypted to ensure that an internet ballot sent by a voter does not reveal anything about the vote. In addition, the database containing all the votes must also be encrypted and secured to prevent the disclosure of how individuals voted, thus ensuring trustworthiness.

4.2 Authentication

Authentication verifies the identity of each voter to prevent unauthorized individuals from casting votes. It is essential that a voter cannot vote using another person's identity.

4.3 Access control and authorisation

Access control and authorization ensure that only authorized individuals have access to certain data or functions within the voting system. In this context, only personnel responsible for maintaining the IT infrastructure should have access privileges. Importantly, voters must also be assured of their access to the online voting system. Depriving a voter of access to the actual voting system would compromise the election.

4.4 Data integrity

Data integrity guarantees that the information has not been altered or tampered with throughout its entire lifecycle. This is crucial for online voting to ensure that votes are accurately recorded and counted. An internet ballot sent by a voter should not be able to be altered without it being detected. Further, the database must maintain consistency and prohibit any modifications to the collected votes.

4.5 Non-repudiation

Non-repudiation prevents an individual or entity from denying having performed a particular action. For online voting, this means ensuring that once a vote has been cast, the voter cannot deny their action, and the system cannot deny having received the vote.

4.6 Availability

Availability ensures that the voting system is accessible to all authorized voters during the defined voting period, without interruption. An election could be compromised in the event that voters are unable to access the voting service when needed, underscoring the importance of maintaining continuous availability to uphold the integrity of the election process.

5 Attacks and Vulnerabilities

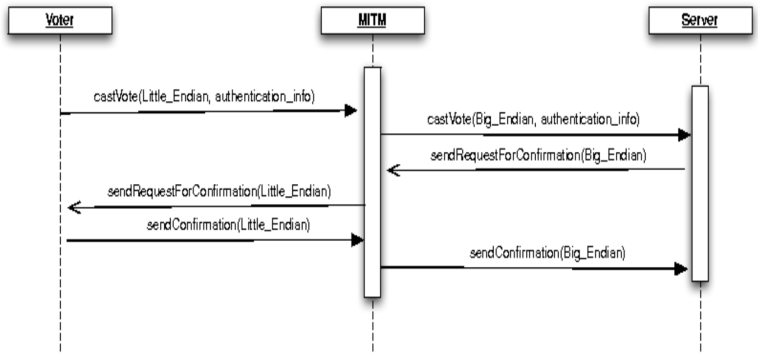
Online voting systems are inherently vulnerable to a variety of different attacks, presenting significant challenges to the integrity, confidentiality, and availability of the voting process. These vulnerabilities can broadly be categorized into two subgroups: those targeting data, including confidentiality and data integrity, and those aiming to disrupt the availability of resources [2]. Each poses unique risks that must be addressed to ensure the trustworthiness of online voting platforms. Although there are a plethora of attacks that can be carried out on online voting systems, some of the most relevant ones are described in this section.

One common form of attack targeting data integrity is the Database Injection Attack, which exploits vulnerabilities in input fields to insert malicious SQL queries [3]. While such attacks can potentially compromise user data stored in the backend, they can be avoided relatively easily. These attacks tend to occur due to a lack of awareness of the topic, resulting in database code that can be easily

injected and altered. It is not necessarily an attack that will be a major point of concern, since the methodologies we are using will be sufficient to avoid the use of input fields where data can easily be inserted into the database. However, it is still a point worth mentioning that is relevant for any system that requires some sort of user input.

Another very common category of attacks is man-in-the-middle attacks (MITM), where a malicious actor intercepts and possibly alters communication between two parties without their knowledge or consent. It is an attack that could potentially damage the integrity and confidentiality of our system. In a typical e-voting scenario, a legitimate voter connects to a remote server via their personal computer to cast their ballot. However, malicious actors can exploit vulnerabilities in the system to carry out MITM attacks. The attack scenario can be visualized using the given sequence diagram in Figure 1. These attacks can occur either locally, through the use of downloaded Trojan horses

Figure 1: Man-in-the-middle Attack on the System [4]



that hijack authenticated sessions, or remotely, by redirecting traffic to a rogue server via methods such as email phishing or ARP/DNS spoofing [4]. In Estonia, where online voting is widely adopted, the security of such systems was analyzed by several papers and found to be at risk. Observations of the pre-election setup process in the National Electoral Committee's offices revealed potential vulnerabilities, such as downloading software over unsecured HTTP connections, which could be exploited by network-based MITM attackers to introduce malware into the configuration process [5]. Consequently, the attackers could not only count and detect the votes by listening to connections, but also manipulate the voting process.

Furthermore, it is impossible to ignore the importance of social factors, as demonstrated by phishing attacks, which use false emails or messages to deceive people into disclosing their passwords or casting wrong votes. Even if these attacks might not specifically target system flaws, they still highlight how crucial user knowledge and education are to reducing the risks connected with online voting. While these attacks are very difficult to overcome, since there is more of a human factor involved than a technology factor, we will also consider non-repudiation of not only the voters, but also our system so that the users can be sure that the website or application they are entering to vote is indeed the official location to cast their vote. We should ensure that it is similar to traditional voting where there are pre-determined official polling stations for the public.

Yet another and probably one of the biggest threats to online voting systems is the Distributed-

Denial-of-Service (DDoS) attacks. These attacks could potentially disrupt access to crucial election information and services. Both application and protocol layer attacks can render election office websites or mobile applications temporarily inaccessible, taking away the voters' ability to participate in the electoral process. According to the Cybersecurity and Infrastructure Security Agency, during the 2022 midterm elections, multiple state and local election offices experienced temporary website outages due to such attacks [6]. Apart from disturbing the election process, such attacks also create opportunities for foreign actors to spread disinformation about the voting and lead to public distrust in the voting system.

Our system will attempt to consider these vulnerabilities and attacks in order to ensure an e-voting process that can be secure and trustworthy for all of the involved parties.

6 Methodology

Different techniques were considered to have a secure online voting system, which are discussed in this section. Before going into the specific details regarding our, the entire process from start to finish will be explained, which are further elaborated on later.

First, when the back-end database has been built and tested, every voter receives an invitation letter, preferably through the eBox service. This letter contains practical information about the voting process, including a link to the platform, in order to make sure that everyone will be ready to use the platform on the day of voting. This information should also include a detailed guide for setting up Itsme, since it will be required for authentication. Along with this practical information, the anti-phishing secret will be shared, and its purpose should be explained clearly, so people know it is there only to help them.

On the day of voting, voters go to the URL provided in their invitation letter (it is the same URL for everyone, of course). All communication with the back-end happens over HTTPS. If all goes well, the user is presented with an authentication page, where they get a button to log in using the Itsme Service. If that has been completed, the back-end can verify the identity of the voter, and verify if they have not voted yet. If they have, the server returns a page which tells them they are only allowed to vote once. If they haven't, the server responds with a voting page, showing all the different options and the anti-phishing code. The user can now verify the legitimacy of the site, before they enter their answers. Their voting request is sent to the server, which updates the database as explained in 6.6.

6.1 Authentication (Itsme)

Authentication is one of the most important aspects that we need to consider in online voting to verify the identity of voters and maintain the integrity of the electoral process. By ensuring that only authorized individuals can access and participate in voting, authentication helps prevent fraud and manipulation of election results. Robust authentication measures must be put in place to reinforce the legitimacy of our online voting system. To ensure that authentication is properly implemented, we have decided to use the Itsme platform.

6.1.1 Why Itsme?

Initially, we were planning on using other methods, such as two-factor authentication (password and biometric or password and token/SMS code) with encryption. However, after brainstorming more, we have realized that all of these methods pose some serious problems in the context of authentication. Eventually, after some more research and discussions, we have come up with the idea of making use of Itsme. Utilizing Itsme for authentication offers numerous benefits for our project. Its passwordless login feature not only enhances security by reducing the risk associated with passwords but also simplifies the user experience. Additionally, its use of asymmetric-key encryption ensures secure communication of ID data across various channels, while its integration of two-factor authentication further strengthens the overall security of our online voting system. Moreover, Itsme provides consistency across different devices and channels, ensuring accessibility for a wide range of users, which is crucial for e-voting.

Itsme is a privately-owned authentication system accessible to individuals with a Belgian eID or a Belgian bank account. Utilizing the Itsme system, individuals can be reliably identified, and their voting status can be verified by cross-referencing their unique identification with their ID card. The integration of this system ensures that the creation of fraudulent Itsme accounts or accounts by non-Belgian citizens is impossible. Operating via a mobile app, users verify their identity using a PIN, with authentication processes including checks for the app's original installation device and SIM card changes. Itsme provides a robust authentication process by requiring both physical possession of the user's phone and knowledge of their PIN, ensuring two-factor authentication. Furthermore, Itsme's security measures, such as the option to block accounts in case of phone loss or theft, coupled with its adherence to ISO 27001 certification standards, give more confidence in the reliability and integrity of the authentication system. [7].

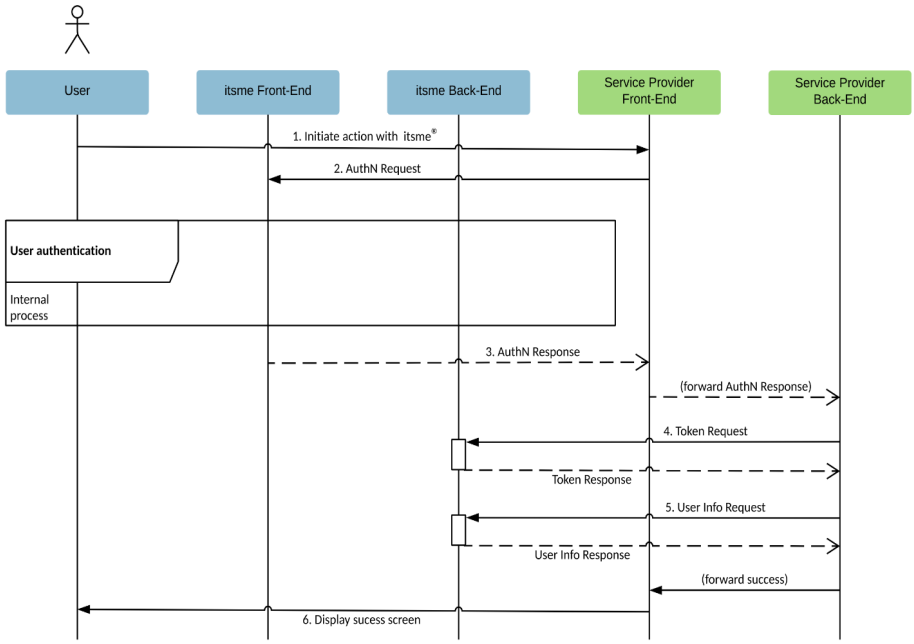
6.1.2 How does Authentication with Itsme Work?

The diagram in Figure 2 summarizes how the Itsme authentication process works. When the user performs the action that causes the Authorization Request to be initiated, they are redirected to the Itsme page. There, you'll verify your phone number, prompting the Itsme app on your phone to spring into action. The app will ask you to confirm the information being shared, giving you the option to use your Itsme code, fingerprint, or FaceID for added security. Upon successful authentication, the platform collects the Authorization Code (given as AuthN in the Figure 2), exchanges it for an ID token and Access Token, retrieves additional claims if necessary, and confirms the operation's success to display a success message to the user.

6.1.3 Use of Encryption

To ensure secure communication between our backend and Itsme, we have the option to choose between two authentication methods: "Private key JWT" and "Client secret." "Private key JWT" utilizes a public-private key pair for asymmetric encryption, offering the highest level of security. This method requires exposing public keys via a JWK Set document and keeping private keys secure. Alternatively, the "Client secret" method employs a shared secret key for symmetric encryption, which can be simpler to implement but may not offer the same level of security. In our application, we would enforce asymmetric encryption with at least 512 bits (RSA512) to enhance security. For key rotation in the "Private key JWT" method, we can add new keys to the JWK Set document and gradually transition to using the new key while keeping the old one temporarily. Once validated, the old key can be removed.

Figure 2: Itsme Authentication Process [8]



Itsme offers an additional security layer called Proof of Key for Code Exchange (PKCE) to enhance the security of authentication flows. PKCE mitigates potential Authorization Code interception attacks by requiring the use of a random string called code_verifier, which is hashed into a code_challenge using SHA256. This code_challenge is included in the Authorization Request, while the code_verifier is sent with the Token Request, ensuring that both requests originate from the same source.

6.1.4 Certificates

Itsme enforces the use of HTTPS connections to ensure robust security for its authentication services. Websites seeking integration with Itsme must obtain SSL/TLS certificates from certificate authorities, which involves submitting a certificate signing request containing site details and a public key. These certificates are categorized into Domain Validation, Organization Validation, and Extended Validation. Itsme only supports Organization Validation and Extended Validation certificates due to their higher identity guarantees. Domain Validation (DV) certificates offer the lowest level of assurance for clients due to their automated registration process. These certificates primarily verify that the domain name is registered and do not conduct thorough identity checks on the entity behind the website. The confirmation process relies on email responses and setting up DNS records, which can be vulnerable to exploitation by attackers [9]. In the case of Itsme, the chain of trust for these certificates must be publicly accessible. Additionally, the implementation of the Server Name Indication (SNI) extension in TLS protocols further enhances security by enabling servers to provide the correct certificate based on the queried endpoint, ensuring secure and accurate authentication exchanges [8].

6.2 Website Verification

Phishing attacks are increasing in numbers each year. It is very hard to verify the trustworthiness of any online resource. For example, say the voting website resides on the domain "https://vote.be". Even if the voter enters this URL into their browser correctly, the systems that are in place to route the user to our trusted back-end are not totally infallible (e.g. DNS spoofing [10]).

This could impair voters' confidence in the entire system, if they are even aware of the threats. We therefore propose an anti-phishing technique in which users get the ability to verify the voting platform they see on their screens. To do this, the platform will show a secret which is specific to each user, such as a 6-character code. This requires some secure communication channel, such that the users can receive the code beforehand and trust that this is not a phishing attempt.

We suggest using the eBox digital mailbox service, which is a secure communication channel provided by the Belgian federal government, reserved for official documents.

This way, when a user ends up on a site that looks legitimate, but shows an incorrect 6-character code, they know this is not the actual voting platform, but some malicious attempt at getting their voting data. Conversely, when the code does match, they can be a lot more certain about the legitimacy of the site.

6.3 Non-repudiation of origin

In Belgium, it is mandatory to participate in the voting process. This is one of the reasons that a person should be able to prove that he/she cast a vote for someone, independent of what the database says. This can be achieved through a digital signature of the confirmation. The system generates a hash (preferably SHA512) from the confirmation, encrypts it using its private key and sends it to the user. The user decrypts the digital signature using the public key and generates its own hash of the confirmation. If the hashes match, we know we are talking about exactly the same confirmation. Since the private key is required to produce a valid digital signature, it serves as proof of the sender's identity and intent.

The other way around works as well, but this would mean that every single person in Belgium has a public and private key. As most people don't even know what a public/private key-pair is, it imposes a big threat on availability.

For optimal security, we could use a HSM (Hardware Security Module) box to store the private key. Deriving multiple public keys from the private key mitigates the risk associated with a single point of failure.

6.4 Data Transmission

Secure data transmission is essential to safeguarding the integrity and confidentiality of online voting. A ballot must be kept confidential and unaltered. The most common method for securing communications over the internet is through Transport Layer Security (TLS). Think of TLS as creating a secure tunnel between the voter's device and the election website. TLS also verifies the identity of the website, making sure it's really who it claims to be.

TLS achieves confidentiality, authentication, and integrity in order to prevent man-in-the-middle

(MITM) attacks, as explained further. During the TLS handshake, a secret session key is established, which is used to encrypt and decrypt the data, ensuring confidentiality. The election server presents a digital certificate issued by a trusted Certificate Authority (CA). The voter device verifies the certificate's authenticity by checking its validity period and if it's signed by a trusted CA. This ensures communication with the legitimate website and provides authentication. Lastly, to ensure integrity, TLS uses hash verification. Before sending the data, the voter's device creates a unique mathematical fingerprint (hash) of the data. Upon receiving the data, the server calculates its own hash of the received data. If both hashes match, it confirms the data hasn't been altered during transmission. Any changes to the data would result in a different hash, indicating tampering.

We will leverage TLS through HTTPS. One critical aspect of implementing TLS is the selection of a reliable Certificate Authority (CA) to issue digital certificates. For our proof of concept, we opt to use self-signed certificates. While typically not recommended due to the lack of organization validation, self-signed certificates are sufficient in this controlled setting and allow us to demonstrate secure data transmission. These self-signed certificates use Domain Validation (DV) which only verifies the control of the domain name, not the organization itself, posing a security issue. Outside of this controlled setting, we recommend contacting a trusted CA organization.

Unfortunately, TLS can be vulnerable to side-channel attacks. For example, it is essential to ensure that the length of each packet is consistent. Otherwise, an attacker could correlate packet lengths with specific actions or votes, compromising confidentiality. Further, attackers could exploit the compression ratio of a TLS request and potentially decrypt requests. To mitigate these and other attacks, our proposed system must utilize the latest and safest versions of TLS (1.3) (2024) [11]. One must remember that security is an ongoing process, and staying informed about the latest developments is crucial.

6.5 Data Storage

We will rely on PostgreSQL for our data storage for several reasons:

1. ACID properties

   The ACID properties are a set of four characteristics that ensure reliability and consistency in database transactions. They are essential for maintaining the integrity of the voting data and ensuring that database transactions are executed reliably.

   - Atomicity: Transactions are treated as indivisible units of work. It ensures that either all operations within a transaction are successfully completed and committed to the database, or none are. If any part fails, the entire transaction is rolled back, keeping the database's original state.
   - Consistency: It ensures that the database remains in a valid state before and after each transaction. It enforces integrity constraints, such as foreign key relationships and data validation rules.
   - Isolation: Each transaction is executed in isolation from other transactions, preventing interference.
   - Durability: Once a transaction is committed, its changes are permanent and survive system failures. Durability guarantees that committed data remain intact and accessible,

even in the event of crashes or power outages, by ensuring that changes are safely written to durable storage.

2. Open source + big community

   PostgreSQL's source code is open for anyone to inspect, modify, and enhance. This transparency allows a large community of developers to review the code continuously, identifying and fixing security vulnerabilities promptly. In contrast, in some apps where the source code is closed, only a limited amount of people have access to it, thus also limiting the number of people who can review it for security flaws.

3. Scalable

   PostgreSQL is scalable for multiple reasons: every connection is handled by a separate process, sharding, various replication mechanisms for load balancing and more. To ensure reliable availability for the 11 million people in Belgium at practically the same time, this is absolutely a must.

4. Security

   PostgreSQL offers a robust authentication and authorization system. It supports various authentication methods, including password-based authentication, certificate authentication, LDAP authentication, and more. Additionally, PostgreSQL allows fine-grained access control through roles and privileges, enabling granular permissions for users and roles. This granularity is essential in a system of governance as complex as ours.

   It also provides built-in mechanisms for encrypting data stored in the database using Transparent Data Encryption (TDE) techniques. Additionally, PostgreSQL supports Secure Sockets Layer (SSL) encryption for encrypting data transmitted between the client and the server, ensuring data confidentiality and integrity during transmission.

   Finally, PostgreSQL also provides comprehensive auditing and logging capabilities, allowing you to monitor and track database activity. It supports the logging of various events, including database connections, authentication attempts, data modifications, and more.

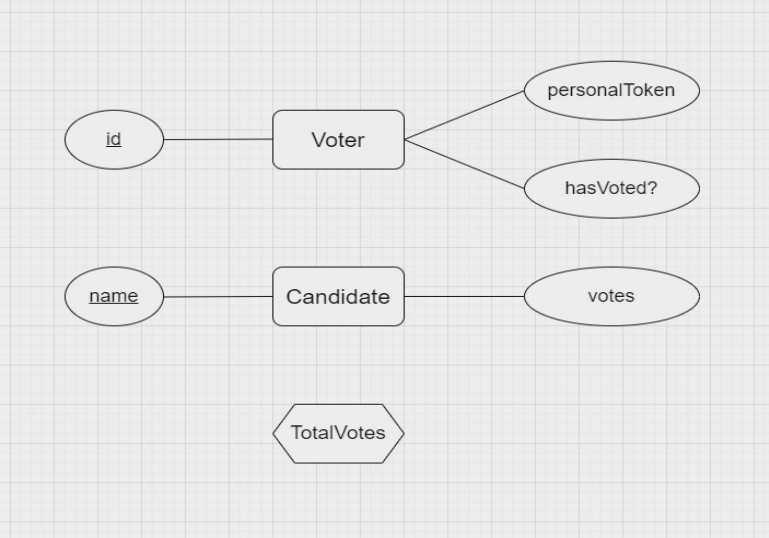
6.6 What data will be stored and how does the system work?

Using the National Register of Natural Persons (a database managed by the Belgian federal government, containing information about Belgian citizens and residents), one can pull the names of everyone that may vote in a table. Then, for every person, a Boolean indicating whether or not the person has voted yet, is initialized to false. These persons will be referred to as 'voters' in the future.

The system also needs information about the people one can vote for: the 'candidates'. Each candidate has a counter, indicating the amount of votes they received.

As is the case in the actual voting system, keeping track of the total number of votes might be a good idea. This serves as an extra check for the validity of the system. Namely, if the sum of the votes for each candidate is different from the separated 'total-vote' counter, we know something went wrong. The database is now ready to be used. Once the backend receives a voting request. The Boolean indicating if that person has voted yet is set to true. The counter for the selected candidate goes up by one, as well as the total amount of votes.

Figure 3: EER diagram



6.7 Why is the system secure?

Let's assume PostgreSQL crashed or a hacker messed up the data. Do we have a backup? For that, there is the Write-Ahead Logging mechanism. It ensures that modifications to the database (writes) are logged before they are applied to the actual data files. Now, by replaying the WAL log from the last checkpoint, the database gets brought back to a consistent state. WAL allows for precise recovery to any point in time, so every vote is accounted for.

What in case of a silent breach of the database where a hacker spies on the data without adjusting it? First, one should notice the lack of information in the database about who voted for who. Just as in the real voting process, this is an (almost) sure-fire way to maintain the anonymity of the vote. However, there is a vulnerability. Setting the has-voted property to true and raising the counter of the candidate by one, is an atomic operation. This is positive because it ensures data integrity, but also negative because a hacker can simply look at what values change at the same time and use that relation to detect who votes for who the moment the vote happens.

A simple countermeasure would be to encrypt the names of the voters. Itsme provides this service already. Upon requesting a voter's personal information, we get back a subject-id as a hash. By using this instead of the actual name to index the has-voted table, we ensure total anonymity of the vote. Another countermeasure could be to add a random delay between setting has-voted to true and increasing the counter. However, this comes with its own set of problems, making encryption the easier way out.

7 Conclusion

In conclusion, the system we presented here offers a lot of security measures that together should provide a voting platform in which voters can trust the platform, and the platform can trust the voters. Data integrity is ensured through the usage of a secure database, achieved through both

technological measures and physical on-site protection against potential attacks.

However, the proposed solutions also have some disadvantages, as would any other techniques. Overall, an entirely internet-based voting system might be discriminating for older people who are not used to using computers on a day-to-day basis. However, even traditional voting systems are discriminating against some groups of people, i.e. there is no perfect solution. Additionally, restricting the authentication service to one provided by a third-party might be sub-optimal, but also ensures a proven and tested solution.

Note: chatGPT was used for the purposes of paraphrasing and grammar correction throughout this paper. [12]

References

[1] Inter-Parliamentary Union. Declaration on criteria for free and fair elections. `https://www.ipu.org/impact/democracy-and-strong-parliaments/ipu-standards/declaration-criteria-free-and-fair-elections`, n.d.

[2] A. Rodríguez-Pérez. Five common attacks against online voting. *Medium*, December 15 2021.

[3] Mohamed Rua, Thiyab, Musab, A. Ali, Farooq Abdulqader, and Abdulqader. The impact of sql injection attacks on the security of databases. 2017.

[4] Dimitris Mitropoulos and Diomidis Spinellis. Securing e-voting against mitm attacks. In *PCI 2009: Proceedings of 13th Panhellenic Conference on Informatics*, 2009.

[5] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman. Security analysis of the estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[6] Cybersecurity and Infrastructure Security Agency (CISA). *No Downtime in Elections: A Guide to Mitigating Risks of Denial-of-Service*, September 2023.

[7] F. Roelofs. Analysis and comparison of identification and authentication systems under the eidas regulation. 2019.

[8] itsme OIDC Documentation. Authentication api. `https://belgianmobileid.github.io/doc/authentication/`, n.d.

[9] B. Saltaformaggio and M. Konte. Understanding the role of extended validation certificates in internet ... June 28 2019.

[10] Wikimedia Foundation. Dns spoofing. `https://en.wikipedia.org/wiki/DNS_spoofing`, March 29 2024.

[11] Eric Rescorla. Rfc 8446: The transport layer security (tls) protocol version 1.3. `https://www.rfc-editor.org/rfc/rfc8446`, August 2018. (Accessed on 05/09/2024).

[12] OpenAI. Chatgpt, 2021.