ADSES: ANONYMOUS AND DECENTRALIZED SECURE ELECTION SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EŞREF ÖZTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2023

Approval of the thesis:

ADSES: ANONYMOUS AND DECENTRALIZED SECURE ELECTION SYSTEM

submitted by **EŞREF ÖZTÜRK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Ertan Onur
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Pelin ANGIN
Computer Engineering, METU _____

Prof. Dr. Ertan Onur
Computer Engineering, METU _____

Prof. Dr. İbrahim Körpeoğlu
Computer Engineering, Bilkent University _____

_____

_____

Date:08.09.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Eşref Öztürk

Signature :

ABSTRACT

ADSES: ANONYMOUS AND DECENTRALIZED SECURE ELECTION SYSTEM

Öztürk, Eşref
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Ertan Onur

September 2023, 67 pages

This thesis proposes an e-voting scheme, the Anonymous and Decentralized Secure Election System (ADSES), designed to fulfill the essential requirements for secure, accurate, and transparent elections. ADSES employs a multi-stage methodology incorporating distinct user roles, such as Voters and Officials, and utilizes advanced cryptographic algorithms to safeguard the elections integrity and security. Our system thoroughly approaches every aspect of the election process, covering everything from pre-election to post-election procedures. We aim to ensure that the election is verifiable, anonymous, authenticated, accurate, transparent, consistent, dependable, and efficient in terms of time while preventing any intermediary results.

A thorough security analysis of ADSES is provided, showcasing its efficacy in meeting the core requirements for a reliable and trustworthy election process. By empowering voters and observers to verify election results and guaranteeing the protection of individual votes, ADSES fosters trust and confidence in the election system. As a pioneering e-voting protocol, ADSES offers a promising solution to enhance the security and dependability of elections, setting the stage for the widespread adoption of e-voting systems in various democratic processes worldwide.

Keywords: E-voting protocol, Secure election system, Verifiability, Anonymity, Trust

ÖZ

ADSES: ANONİM VE MERKEZSİZ GÜVENLİ SEÇİM SİSTEMİ

Öztürk, Eşref
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ertan Onur

Eylül 2023 , 67 sayfa

Bu tez, güvenli, doğru ve şeffaf seçimler için temel gereksinimleri yerine getiren Anonim ve Merkezi Olmayan Güvenli Seçim Sistemi (ADSES) adlı bir e-oylama sistemi önermektedir. ADSES, seçimin bütünlüğünü ve güvenliğini sağlamak için gelişmiş şifreleme algoritmaları kullanarak, Seçmenler ve Yetkililer gibi farklı kullanıcı rollerini içeren çok aşamalı bir metodolojiye dayanmaktadır. Sistemimiz, seçim öncesinden seçim sonrası prosedürlerine kadar seçim sürecinin her yönüne kapsamlı bir yaklaşım sunmaktadır. Amacımız, seçimin doğrulanabilir, anonim, kimlik doğrulamalı, doğru, şeffaf, tutarlı, güvenilir ve hızlı bir şekilde gerçekleştirilmesini sağlamak ve aynı zamanda herhangi bir ara sonuç oluşumunu önlemektir.

ADSESnin ayrıntılı bir güvenlik analizi sunularak, güvenilir ve güvenilir bir seçim sürecine yönelik temel gereksinimleri karşılamasındaki etkinliği gösterilmektedir. Seçmenlerin ve gözlemcilerin seçim sonuçlarını doğrulama yetkisi vererek ve bireysel oyların korunmasını garanti ederek, ADSES seçim sistemine güven ve güvenilirlik katar. Öncü bir e-oylama protokolü olarak, ADSES, seçimlerin güvenliğini ve güvenilirliğini artırmak için umut vaat eden bir çözüm sunarak, dünya genelinde çeşitli

demokratik süreçlerde e-oylama sistemlerinin yaygın olarak kullanımının önünü açmaktadır.

Anahtar Kelimeler: Elektronik oy verme protokolü, Güvenli seçim sistemi, Doğrulanabilirlik, Anonimlik, Güven

To my family

ACKNOWLEDGMENTS

I am grateful for the support and guidance my advisor, Prof. Dr. Ertan Onur, provided throughout my thesis work. His invaluable advice and patience played a crucial role in helping me reach this point.

Additionally, I sincerely appreciate my friends who offered their unwavering support and insightful feedback. Their challenging questions helped me to refine my ideas and improve the quality of my work.

Finally, I am indebted to my family for their unwavering support and encouragement. Their efforts have been instrumental in my success, and I am grateful for their continued belief in me.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

v	The vote that the voter casts.
r	Random string attached to the vote.
id_i	A random string generated by each Official specifically for the election.
id	IDs of all Officials concatenated in lexicographical order.
m	The message generated by the user, obtained by concatenating v, r, and id.
m_e	The vote after encryption with all Official's encryption keys and concatenation with id.
m_s	The vote after signing with all Official's signing keys.

CHAPTER 1

INTRODUCTION

Elections have been a fundamental part of human decision-making since ancient times, but they have gained particular significance since the 17th century [4]. Large-scale elections, such as national elections, have far-reaching consequences and impact millions. Consequently, malicious actors often target these elections, making election security a critical and extensively researched topic [5] [6] [7].

As technology advances and permeates various aspects of our lives, elections are also increasingly incorporating electronic systems. E-voting has become more prevalent, but it presents numerous security challenges. Computers can perform operations without human intervention, making it easier for malicious actors to manipulate votes without detection.

Transparency is a significant concern in e-voting systems. Since computers operate in the background, people may be unable to observe the entire voting process, leading to suspicions about the accuracy of election results. To address this issue, we have designed ADSES, a system that leverages a distributed peer-to-peer database to reflect all election operations transparently. This approach allows Voters to observe all procedures, ensuring transparency and trust in the system.

A vital component of this thesis is the distributed database system, which ensures that data remains unchanged over time. In the context of this thesis, we use a peer-to-peer distributed database system [8] with an immutable data structure [9], as it is well-suited for such purposes. Readers should note that any reference to a distributed database system in this thesis pertains specifically to this type of system.

In the following sections, we introduce ADSES and explain how it fulfills the require-

ments of a secure election system. We also compare ADSES to other related works in the field to demonstrate the innovations it brings to the state of the art.

The structure of this thesis is as follows: First, we provide a detailed overview of the ADSES system, including its design and the cryptographic features that ensure secure voting. We then explore the various stages of the election process, describing the roles of different entities and the steps taken to guarantee transparency, privacy, and security. In addition, we discuss possible attack vectors and explain how ADSES mitigates these risks.

Next, we delve into the cryptographic features that satisfy essential security properties such as forward secrecy, backward secrecy, and unlinkability. We also address the issue of ensuring unique votes and how the system handles this challenge.

Finally, we compare ADSES to existing e-voting systems and highlight the advantages and innovations it brings to the field. We conclude by discussing the potential for future improvements and enhancements to the ADSES system and its potential impact on electronic elections' overall security and transparency.

In summary, ADSES presents a novel approach to secure and transparent e-voting, leveraging the strengths of distributed databases to address the challenges inherent in electronic voting systems. By offering a comprehensive solution that satisfies crucial security requirements and ensures transparency throughout the voting process, ADSES has the potential to contribute significantly to the advancement of e-voting systems and promote greater trust in the democratic process.

CHAPTER 2

RELATED WORK

The field of electronic voting systems has received significant attention from researchers and practitioners, aiming to address the security and transparency challenges associated with elections. This section provides an overview of existing related works in the domain of e-voting systems.

As mentioned in ADSES Requirement analysis section 3.3, an e-election system must fulfill multiple vital requirements to ensure a trustworthy, secure, and efficient election process. These include verifiability, with both public and individual aspects; protecting voter anonymity; implementing robust authentication mechanisms; ensuring accuracy through prevention of duplication, tampering, and impersonation; maintaining transparency while preserving voter privacy, guaranteeing consistency among all stakeholders, ensuring dependability against attacks, withholding intermediary results, promptly revealing final results, optimizing time efficiency, and designing for scalability. By addressing these requirements, an e-election system can establish a transparent, secure, and efficient process that instills voter confidence, safeguards the integrity of each vote, and accommodates a growing voter population.

Li et al. [10] propose a blockchain-based self-tallying voting protocol in decentralized IoT. Their primary focus is on fairness, satisfying the requirement of no intermediary results. However, their study does not explicitly address anonymity, and they mention that the blockchain used in the model can be a private blockchain or a consortium blockchain, which may have implications for anonymity. Public blockchains are generally considered more anonymous, while private or consortium blockchains may have varying levels of transparency or restrictions. On the other hand, it is essential to note that ADSES ensures anonymity by separating signing operation from

placing votes in the database operations.

Chaudhar et al. [11] propose a blockchain-based voting system that utilizes 5G networks and smart contracts [12], which are implemented using solidity language [13]. Their objective is to display the total votes for each candidate throughout the voting process, creating a transparent environment for participants. However, their system fails to meet the requirement for intermediary results. In contrast, ADSES guarantees that no intermediate results are made public until the post-election phase. This is achieved by ensuring voters encrypt their votes using Officials' encryption keys.

Rathee et al. [14] propose a blockchain-based voting system that utilizes IoT-Oriented Smart Cities. They use polling booths for voters to cast their votes. However, their system lacks anonymity, accuracy, and transparency because the Authorities provide the machines in polling booths. In contrast, ADSES focuses on enabling voters to transparently perform operations on their devices while verifying all other functions that Officials must carry out without exposing their identities. This is achieved through the use of blind signatures and encryption.

Hjalmarsson et al. [15] propose a blockchain-based electronic voting system using an Ethereum private blockchain. Still, their approach lacks anonymity as voters cast their votes in a supervised environment using custom software installed on computers in voting districts. In contrast, ADSES ensures voter anonymity by allowing voters to use their machines with an open-source app and providing Officials with only blinded votes. Furthermore, while Hjalmarsson et al. [15] allow immediate visibility of all votes to Officials, ADSES ensures the requirement of withholding intermediary results by having voters encrypt their votes with Officials' encryption keys, ensuring that election results remain confidential until the election concludes.

Shahzad and Crowcroft [16] also propose a unique approach to electronic voting systems by leveraging blockchain technology. However, their solution falls short of meeting the requirement of individual verifiability. In their system, voters have to use central voting machines and do not receive feedback on their votes, limiting their ability to verify the accuracy of their votes independently. While blockchain technology offers benefits in decentralization and immutability, its approach does not adequately address the essential requirement of individual verifiability for a secure

and trustworthy e-election process. In contrast, ADSES ensures individual verifiability by allowing voters to personally encrypt and sign their votes. These encrypted votes are then stored in a peer-to-peer distributed database, enhancing transparency and instilling confidence in the integrity of the voting process.

McCorry et al. [17] and Lai et al. [18] propose a smart contract-based approach to the Ethereum blockchain for electronic voting systems. Still, its scalability is limited due to the Ethereum network's constraints and the high gas fees associated with maintaining the order of votes. In comparison, ADSES addresses scalability concerns by employing a distributed database architecture. ADSES does not rely on maintaining the order of votes, instead focusing on individually verifying each vote against the signatures of all officials. This approach allows ADSES to accommodate a growing voter population more efficiently and cost-effectively, making it a more scalable solution when compared to McCorry et al.'s Ethereum-based approach.

Gao et al. [19] propose an anti-quantum electronic voting protocol with a trade-off between efficiency and security. However, their approach has a computation overhead time that grows proportionally to the number of Voters. Moreover, they focus on achieving real-time attention to the election results by making vote contents readable on the blockchain, which violates the requirement to withhold intermediary results and leaves the election open to manipulation. In contrast, in ADSES, all votes are encrypted with each of the Official's encryption keys and remain confidential until the post-election phase, adhering to the requirement of withholding intermediary results and ensuring the prevention of manipulation.

Wang et al. [20] propose a self-tallying voting system that allows individual voters to tally votes without third-party control, providing transparency. However, their system's scalability is limited as it requires each voter to store a state matrix with all voter information, making it impractical for large-scale elections. Additionally, their protocol lacks a clear voter selection and authentication method, relying on random voter assignment and secret distribution, compromising security. In contrast, ADSES effectively addresses scalability and authentication concerns by employing a distributed database architecture and requiring independent authentication by each Official. These design choices make ADSES a more secure and scalable solution for

large-scale elections.

In their work, Li et al. [21] propose a quantum voting protocol that relies on single-particle states. However, their protocol includes the ID information of voters in the vote, which raises concerns about preserving voter anonymity. This would create a potential risk if voter information were to be disclosed, such as through a voter center. ADSES achieves this by implementing a secure encryption mechanism and a blinded vote approach. Voters use their devices with a trusted open-source application, encrypting and signing their votes individually. The encrypted votes are stored in a distributed database, while election officials only receive blinded votes without any link to a voter's identity. This strict separation of voter identification from the vote in ADSES ensures that even in the event of voter information leakage, it would be impossible to associate specific votes with individual voters, effectively preserving anonymity throughout the e-election process.

Many recent studies have explored the use of blockchain or smart contracts in various contexts [22] [23] [24] [25] [26]. However, these studies also encounter similar challenges, as mentioned earlier. Some survey studies also analyze blockchain e-voting systems and highlight their limitations [27] [28] [29] [30] [31] [32]. While blockchain enhances elections' transparency, it introduces obstacles that hinder fulfilling specific requirements, such as intermediary results, anonymity, and scalability. To address some of these issues, additional mechanisms like encryption or blind signatures can be employed. Nonetheless, it is essential to note that adopting blockchain also entails overhead due to its focus on maintaining a chain structure with ordered data. Since order is not crucial for vote data, ADSES utilizes a peer-to-peer data structure that ensures immutability.

Some studies have also explored using blind signatures [33] [34]. They proposed a system that fulfills the requirement of anonymity. However, their contribution does not encompass all aspects of the election process, including the pre and post-phases with all their details. Additionally, they fail to satisfy the requirement of having no intermediary results. In contrast, ADSES utilizes blind signatures on encrypted data and incorporates keys from all Officials, addressing the missing parts identified in these previous studies.

Lastly, it is essential to note that the concept of e-voting is not a new one, as numerous studies have been conducted on this topic since 1998 [35] [36] [37] [38] [39]. However, significant advancements have been made in this area since then. These advancements include the increased popularity of studies involving distributed databases following the rise of blockchain technology and the availability of better cloud infrastructures for implementation and experimentation. In this study, we comprehensively implement the ADSES system, providing a complete implementation and a detailed cost and time analysis.

In Table 2.2, we compare the essential studies mentioned above concerning the requirements mentioned in the ADSES requirement analysis section (Section 3.3). Each column represents a specific requirement, and each row pertains to a study. The last row represents ADSES, which receives a perfect score (S) for all requirements. Each cell indicates whether the study satisfies (S), fails (F), or is ambiguous (A) for the corresponding requirement, as shown in Table 2.1.

Table 2.1: Degrees of satisfaction used to evaluate how well different studies meet the requirements.

Symbol	**Explanation**
S	Satisfies
F	Fails
A	Ambiguous

Table 2.2: Comparison of ADSES with the state-of-the-art related work.

Study	1a Public Verifiability	1b Individual Verifiability	2 Anonymity	3 Authentication	4a One Vote per Person	4b No Duplication	4c No Tampering	4d No Impersonation	5 Transparency	6 Consistency	7 Dependability	8 No Intermediary Results	9 Time Efficiency	10 Scalability
Li et al. [10]	S	S	F	S	S	S	S	S	S	S	A	S	A	F
Chaudhar et al. [11]	S	S	F	S	S	S	S	S	S	S	S	F	S	A
Rathee et al. [14]	S	S	F	F	A	A	A	A	F	S	S	S	S	A
Hjalmarsson et al. [15]	F	S	F	S	S	A	A	A	F	S	S	F	S	S
Shahzad and Crowcroft [16]	S	F	A	S	S	S	S	S	F	S	S	A	S	S
McCorry et al. [17]	F	S	F	S	S	S	S	S	F	S	S	S	S	F
Lai et al. [18]	S	S	A	S	S	S	S	S	S	S	S	S	S	F
Gao et al. [19]	S	S	S	S	S	S	S	S	S	S	S	F	F	F
Wang et al. [20]	S	S	S	F	S	S	S	S	S	S	S	S	S	F
Li et al. [21]	S	S	F	S	S	S	S	S	S	S	S	S	F	F
ADSES	S	S	S	S	S	S	S	S	S	S	S	S	S	S

CHAPTER 3

ANONYMOUS AND DECENTRALIZED SECURE ELECTION SYSTEM (ADSES)

In this chapter, we provide an explanation and detailed information about ADSES. We present a general overview of how ADSES operates during the election phase. Following that, we define the system model by comprehensively explaining the components involved. Subsequently, we analyze the requirements for a secure election system.

Moving forward, we delve into the specifics of each ADSES phase, including the pre-election and post-election processes. Moreover, we explore various analysis sections, such as Requirement analysis, Attack vectors and countermeasures, and finally, cryptographic features and guarantees.

3.1 The overall Idea

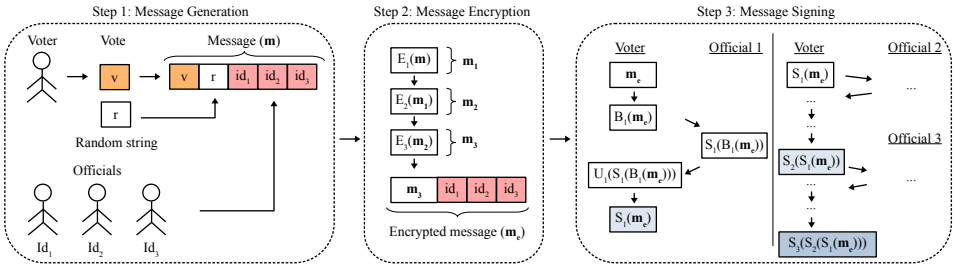

Figure 3.1: Comprehensive representation of the ADSES concept.

To better understand ADSES, let us consider a scenario where Alice participates as a Voter in an election utilizing the ADSES system. A central Authority organizes

the election, and Inspectors from each party or candidate are called Officials within the ADSES framework. This decentralized approach reduces reliance on a single Authority. The roles and responsibilities of these entities are further explained in the system model section.

To cast her vote, Alice begins by selecting the candidate she wishes to vote for. She then appends a random string to her vote to ensure the uniqueness of each vote. Additionally, she attaches an identification string provided by each Official to ensure that her vote is specifically generated for this particular election. The motivations behind these steps are elaborated in the design section of this thesis.

Next, Alice proceeds to encrypt her vote, sequentially using the public encryption key of each Official. This ensures that her vote remains confidential and unreadable until the election concludes, preventing premature disclosure of the results. Subsequently, she blinds her vote for each Official, sending them for signature. The Officials blind sign her vote with their private signing keys. Once she receives the signed votes back, she proceeds to unblind them. Alice also attaches the identification string provided by each Official to her vote before the signing operations, ensuring that the signature of her vote can be validated. This guarantees that while anyone can confirm the validity of the signature, the vote cannot be traced back to the Voter.

To verify eligible voters, Officials can establish a one-time authentication system similar to Turkey's government-issued password system [40], applicable to all future elections.

Lastly, Alice places her vote in a peer-to-peer distributed database with an immutable data structure to ensure that all votes are transparently visible. She can submit her signed vote to the distributed database at her discretion, preventing Officials from establishing connections based on submission time. Additionally, she can employ varying public IPs to enhance anonymity when interacting with the Officials and the distributed database.

After the election concludes, Officials release the private keys of both the Encryption and Decryption key pairs, allowing anyone to decrypt and read the votes in the peer-to-peer distributed database.

In the upcoming sections of this chapter, each phase of the ADSES system will be explained in detail, including the pre-election and post-election processes. Moreover, there will be comprehensive analysis sections covering the requirements, potential attack vectors, countermeasures, and cryptographic features and guarantees offered by ADSES.

3.2 System Model

The system model of the Anonymous and Decentralized Secure Election System (ADSES) comprises several components and processes that ensure a secure and transparent election. This section provides an overview of the critical elements and their operations.

3.2.1 Official

As previously mentioned, ADSES involves Officials who authenticate voters. Typically, a single entity called the Authority is responsible for authenticating voters. However, when the Authority has sole control, it possesses excessive power over the entire election system. For example, it can easily view intermediary results because it holds the private key used in encryption. More entities with the same power level are needed to resolve this issue. As a result, ADSES includes Inspectors who also serve as Officials, similar to the Authority. Inspectors can be selected from the participating candidates in the election.

3.2.1.1 Authority

The Authority plays a critical role in the ADSES system. It is responsible for organizing and overseeing the election process. The Authority ensures the necessary infrastructure, including establishing secure communication channels and generating key pairs for encryption and decryption.

In addition to managing the technical aspects of the election, the Authority is also

responsible for verifying the eligibility of voters. It authenticates the identity of voters and ensures that they are eligible to participate in the election. This verification process helps maintain the election's integrity and prevent fraudulent voting.

To avoid the concentration of power in a single entity, the Authority collaborates with the Inspectors. As mentioned earlier, the Inspectors serve as Officials and possess the same level of control as the Authority. This distributed approach helps to ensure transparency and fairness throughout the election.

3.2.1.2 Inspector

Inspectors, as mentioned earlier, serve as Officials in the ADSES system. They are typically selected from the participating candidates or political parties in the election. The role of Inspectors is to authenticate voters and ensure the proper handling of votes.

Like the Authority, Inspectors are responsible for verifying the eligibility of voters. They play a crucial role in ensuring that only eligible voters can cast their votes and that each vote is treated with confidentiality and integrity.

Inspectors collaborate with the Authority entity to establish a robust and secure election process. Their active participation helps to distribute power and prevent any single entity from having undue influence over the election outcome. Including Inspectors as Officials in the ADSES system aims to create a decentralized and transparent election process that inspires confidence among the participants.

It is important to note that the specific roles of Officials, Authority, and Inspectors can vary depending on the election context. For example, in a presidential election, the Authority could be a government organization responsible for administering the election. At the same time, Inspectors could be representatives from the political parties participating in the election. The exact configuration of ADSES will need to be adapted to the specific election scenario to ensure the appropriate roles and responsibilities are assigned.

By establishing a robust system model with multiple Officials, ADSES aims to dis-

tribute power and reduce the risk of any single entity manipulating the election process.

3.2.2 Encryption

Encryption plays a crucial role in ADSES to ensure the confidentiality and integrity of votes. During the pre-election phase, Officials generate key pairs consisting of public and private keys. Only the public keys are published.

When voters cast their votes, they use the public encryption keys of Officials to encrypt their votes. This ensures that the votes remain hidden and unreadable until the election concludes. The use of encryption maintains the secrecy of each vote and prevents the exposure of intermediary results.

After the election, Officials reveal their private keys for decryption purposes. These private keys are used to decrypt and read the votes. This step allows for the verification and tallying of the votes.

By employing encryption techniques, ADSES provides essential security measures that enhance the overall trustworthiness and integrity of the election process.

3.2.3 Blind Signature

In ADSES, blind signatures [41] serves as a crucial mechanism to enhance the privacy and unlinkability of votes. The blind signature process ensures that the Official cannot establish any connection between the blinded and original votes or between the blinded and original signatures. Here's an overview of the steps involved in how blind signatures work in ADSES:

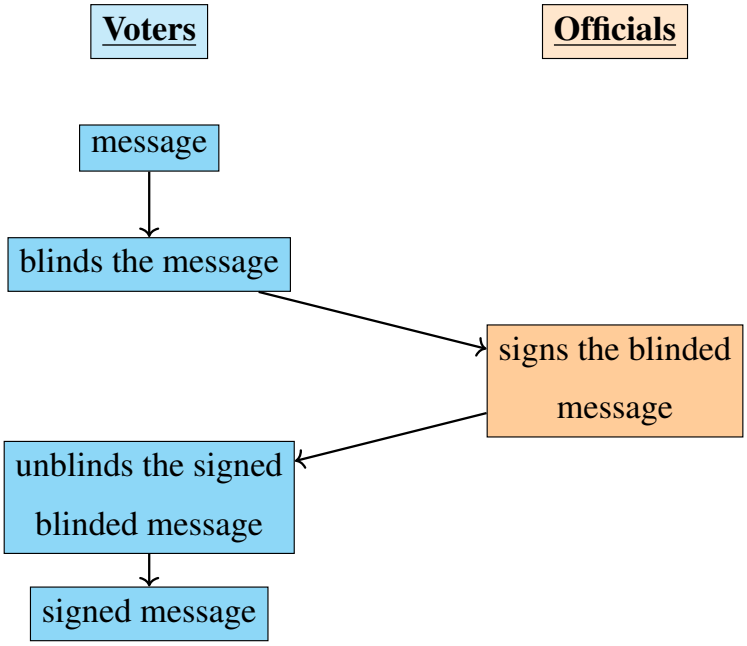

Figure 3.2: Depicts the process flow of the blind signature mechanism.

3.2.3.1 Blinding

Before sending the vote to the Official, the Voter applies cryptographic operations to blind the vote. This blinding process obscures the content of the vote, making it impossible for the Official to determine its actual value or establish any association between the blinded vote and the original vote.

3.2.3.2 Signing

The blinded vote, which conceals its content from the Official, is then presented to the Official for signature. The Official signs the blinded vote, unaware of its actual content. This means that the Official cannot intentionally modify the vote based on the knowledge of specific votes or manipulate the election outcome.

3.2.3.3 Unblinding

Using the inverse cryptographic operations employed for blinding, the Voter can "unblind" the signature. This process retrieves the exact signature for their original vote without compromising its privacy or integrity.

3.2.3.4 Unsigning

Anyone can unsign the vote using the Officials' public signing key pair and verify if the Officials authenticate the vote. For a vote to be saved in a peer-to-peer distributed database, it needs to be confirmed that the vote can unsign with all Official public signing keys, and the resulting value should end with the IDs of all Officials, as explained in the Design section.

Please note that to enable the blinding and reversibility of the signing operation, no padding or hashing mechanism is used during these operations. Votes are directly blinded, signed, unblinded, and unsigned.

By employing blind signatures, ADSES ensures the votes remain private and untraceable. It prevents the Official from linking a vote to a specific voter or being able to distinguish between different votes based on the associated signatures. The blind signature mechanism strengthens the anonymity and integrity of the election process in ADSES, enabling voters to participate confidently in a secure and protected environment.

3.2.4 Peer-to-peer distributed database

A peer-to-peer distributed database with an immutable data structure stores all votes data in ADSES. The requirements for storing votes in the database are as follows:

- Only authenticated votes, meaning votes consecutively signed by all Officials, can be accepted into the database.
- The database with the maximum number of authenticated votes is considered

Table 3.1: Definitions of operations used during the election phase.

Symbol	Definition
$E_i(m)$	Encrypts the message m with the i-th Official's encryption key.
$S_i(m)$	Signs the message m with the i-th Official's signing key.
$B_i(m)$	Blinds the message m with the blinding factor generated for the i-th Official's signing key.
$UB_i(m)$	Unblinds the message m with the blinding factor generated for the i-th Official's signing key.
$D_i(m)$	Decrypts the message m with the i-th Official's decryption key.
$US_i(m)$	Unsings the message m with the i-th Official's signing key.

the most accurate.

These requirements serve the purpose of ensuring that only authenticated votes are included in the database. Furthermore, they guarantee the accuracy of the votes data stored in the database while maintaining consistency among all copies.

The process to add a new vote to the database is as follows:

- Voter sends the signed vote to a peer of the distributed database.
- Peer verifies whether all Officials have signed the vote. If not, it is discarded.
- If the vote is valid, it is saved into the database.

Unfortunately, there is no widely-accepted implementation of a peer-to-peer distributed database with an immutable data structure for storing votes in ADSES. Although OrbitDB [42], which uses IPFS [43] as its data storage, may be considered the closest approximation, even if it is incomplete. This highlights the ongoing challenges and limitations in achieving a fully functional solution for the desired database system. Future development and research are required to address these limitations and refine the implementation of such a database for secure and accurate storage of voting data.

The system model outlined above describes the key components and processes employed in ADSES to ensure a secure and transparent election process. By utilizing

signatures, signing, blind signatures, encryption, and the roles of Authority and Inspector, ADSES meets the requirements for a trustworthy and reliable election system. Table 3.1 shows the operations and their definitions used in the following Design section.

3.3 Requirement Analysis

In this section, we present the basic requirements of an e-election system.

1. **Verifiability**:
   (a) **Public Verifiability** : Allowing everyone to validate the election outcome ensures transparency. Upon election completion, participants can independently verify that the announced results are accurate. Public verifiability is crucial for elections, as voters need confidence in the accuracy of the reported results. This requirement involves making the voting process and results publicly accessible for scrutiny and analysis.
   (b) **Individual Verifiability**: Every voter can confirm that their vote has been accurately included in the final tally. While public verifiability allows for overall result verification, individual verifiability ensures that each voter's choice is accurately reflected. Individual verifiability is vital for elections since voters need assurance that their vote is counted correctly. This requirement involves providing each voter with a means to verify if their vote has been accurately recorded.
2. **Anonymity**: Voter anonymity must be protected, ensuring no one, including officials, can trace a vote back to a specific individual. Anonymity is crucial for elections as it enables voters to cast their votes without pressure or fear of reprisal. This requirement involves implementing mechanisms that prevent the association of votes with voters' identities.
3. **Authentication**: Voting is limited to authorized individuals. Officials prepare eligible voter lists, and only those individuals can cast their votes. After the election, no additional votes should be present in the results. Authentication

is vital for elections to ensure fraudulent votes cannot be added to the system. This requirement involves verifying the identity and eligibility of voters before allowing them to cast their votes.

4. **Accuracy**:
   (a) **One Vote per Person**: Each person can vote only once, and the system prevents multiple voting. Limiting voting to one vote per person is essential for elections as it preserves trust in the electoral process. This requirement involves validating that each voter can only cast a single vote.
   (b) **No Duplication**: Vote duplication is not permitted, even if the content of the vote is unknown. Preventing duplication is crucial for elections to reassure voters that no group can manipulate the results. This requirement involves implementing measures to detect and prevent the inclusion of duplicate votes in the final tally.
   (c) **No Tampering**: Votes cannot be altered without detection, and the system ensures that once cast, votes remain unchanged until the results are announced. Preventing tampering is critical for elections to maintain voter trust in the system. This requirement involves implementing secure storage and cryptographic techniques to ensure the integrity and immutability of the votes.
   (d) **No Impersonation**: Voting on behalf of others is not allowed, and the system verifies authentication credentials to prevent impersonation. Preventing impersonation is essential for elections to protect the integrity of each voter's choice. This requirement involves robust verification mechanisms to ensure that only eligible voters can cast their votes.
5. **Transparency**: The entire election process is open to observation, with all steps adhering to established standards. Transparency is vital for elections as it enables Voters to trust the system through direct observation. This requirement involves providing public access to the various stages of the election process, including vote encryption, decryption, and tallying while maintaining voter privacy.
6. **Consistency**: All election participants maintain identical voting records and ac-

cept the same election outcome. Consistency is crucial for elections to prevent disputes and maintain trust in the system. This requirement involves ensuring that all entities involved in the election process have consistent views of the votes and that discrepancies are detected and resolved.

7. **Dependability**: Cryptographic algorithms guarantee protection against dishonest behaviors and attacks throughout the voting process. Dependability is vital for elections as voters need confidence in the system's security and integrity. This requirement involves selecting and implementing robust cryptographic algorithms and protocols to withstand various threats and attacks.

8. **No Intermediary Results**: Election results are withheld until voting is complete, preventing potential manipulation based on preliminary outcomes. Prohibiting intermediary results is essential for elections to protect voters from undue influence or manipulation. This requirement involves ensuring that the election results are kept confidential until the voting period ends and announcing the results only after all votes have been cast.

9. **Time Efficiency**: Results are revealed promptly after the election concludes. Time efficiency is crucial for elections as voters expect timely results, and rapid announcements reduce opportunities for system tampering. This requirement involves optimizing the voting and tabulation processes to provide quick and accurate results within a reasonable timeframe.

10. **Scalability**: The system must handle many voters and adapt to increasing demand without compromising security, efficiency, or performance. Scalability is essential for elections as it ensures the system remains robust and functional even as the voter population grows while maintaining the integrity and performance expected in the election process. This requirement involves designing the system architecture and infrastructure to handle many simultaneous voters, ensuring efficient and secure communication, and scaling resources according to the demand.

In summary, an election system must ensure verifiability, anonymity, authentication, accuracy, transparency, consistency, dependability, no intermediary results, time efficiency, and scalability. These requirements collectively aim to create a trustworthy,

secure, and efficient electoral process that guarantees the validity of each vote, protects voter privacy, and maintains public confidence in the integrity and fairness of the elections. Implementing these requirements involves combining cryptographic techniques, robust authentication mechanisms, secure storage and transmission of votes, adherence to transparency standards, and adopting scalable infrastructure to accommodate a growing voter population.

3.4 Design

ADSES is designed as a multi-stage e-voting protocol with distinct user roles divided into two groups: Voters and Officials. Voters participate in the election using their local devices, eliminating the need for dedicated voting machines. Officials are responsible for Voter and vote authentication, including the Authority, who organizes the election, and Inspectors from each candidate.

The operations of ADSES can be categorized into three distinct phases: pre-election, election, and post-election. While ADSES primarily focuses on the election process, it also includes preliminary steps to initiate the election and ensures the availability of complete results by tallying votes after the election. The following subsections provide a detailed explanation of ADSES operations during each phase.

3.4.1 Pre-Election Phase

The pre-election phase involves completing operations before the election begins. Some tasks are required before every election, while others can only be performed once.

Key pair and identifier generation operations must be done before each election during the pre-election phase. However, secure channel establishment can be done one time and reused for many elections. It can even be used for processes other than elections by the Authority.

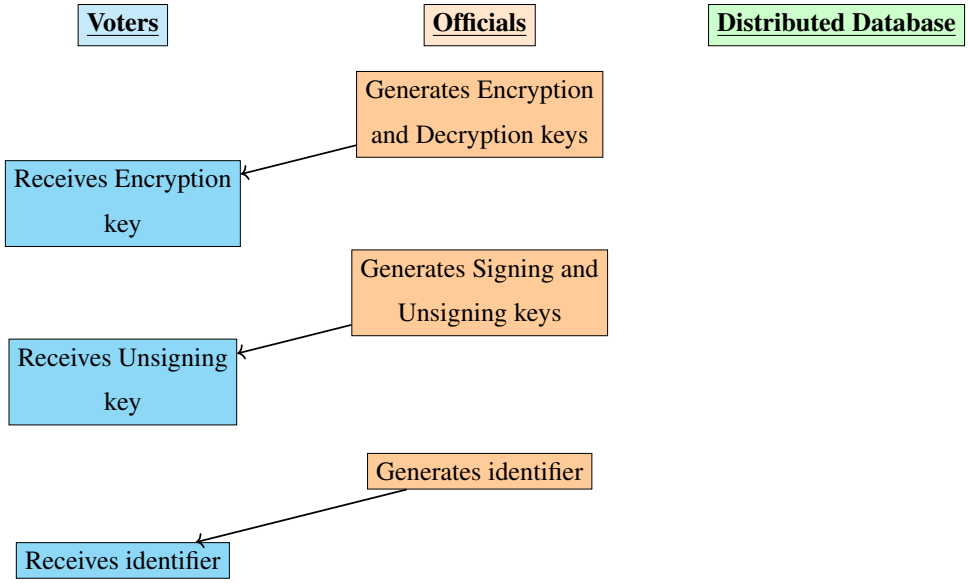

Figure 3.3: Illustration outlining the sequence of actions in the Pre-election phase.

3.4.1.1 Key Pair and Identifier Generation

Each Official generates unique information used by both Voters and Officials during the election and post-election phases. They create two asymmetrical key pairs: Encryption (public) and Decryption (private) keys, and Signing (private) and Unsigning (public) keys. Each Official also generates an identifier for themselves.

Officials only publish the public key parts during the pre-election phase: the Encryption and Unsigning keys. The Decryption and Signing keys are also posted after the election phase is finalized and the post-election phase starts.

Encryption keys conceal results until the election phase is finalized and the post-election phase starts. Each Voter encrypts their votes with each Official's encryption key one by one, ensuring that even officials cannot view the content unless they collaborate. However, collaboration is unlikely since they represent competing interests. When the post-election phase starts, everyone can see the Decryption keys, decrypt all votes, and view the results.

Unsigning keys are used to verify the authentication of the votes placed in the distributed database, ensuring that only authenticated Voters can submit their votes to

the distributed database. Each Voter sends their blinded vote to each Official individually and receives their blinded vote signed by the Official's Signing key. They then unblind it and obtain their signed vote, which cannot be related to the blinded votes sent to Officials in any way. Since Unsigning keys are publicly available, anyone can verify and only accept authenticated votes in the distributed database.

The identifier of each Official is also attached to each vote before encryption by the Voter, allowing verification that the vote is generated specifically for the election. Additionally, the identifier is added after encryption to ensure that the unsigned vote has a specific suffix and is valid.

Key pair and identifier generation is a crucial step in the pre-election phase of the ADSES e-voting protocol. Each Official in the election generates unique asymmetrical key pairs consisting of Encryption/Decryption and Signing/Unsigning keys. These keys play a vital role in securing the voting process. Officials publish the public key parts, enabling Voters to encrypt their votes and verify their authenticity. Additionally, each Official generates an identifier attached to each vote to ensure it is specific to the election and to facilitate the verification of unsigned votes. This process enhances the security and integrity of the ADSES e-voting protocol.

3.4.1.2 Secure Channel Establishment

During the ADSES protocol's election phase, each Official must authenticate every Voter and sign their blinded votes. Establishing a secure channel or method for user authentication is critical to this process. However, the ADSES protocol does not specify a particular way for secure channel establishment, as it depends on each election's specific requirements and existing authentication procedures.

The choice of secure channel establishment must be made for each election and election group individually, ensuring compatibility with the Authority's authentication system or the specific needs of the election organizers. This flexibility allows for integrating the ADSES protocol with the existing infrastructure, enabling a seamless and secure authentication process.

For example, in an election for a Chair position in a Computer Engineering depart-

ment at a university, a suitable method could involve using email as the secure channel. Each Voter could send a "Vote Authentication" email containing their blinded vote to the designated Officials. The Officials would then respond to the email, providing their signature to authenticate the vote. This method relies on the trustworthiness of email ownership, ensuring that only legitimate Voters can participate in the election process.

Similarly, in the context of a national election, the ADSES protocol could leverage an established system, such as an online platform like Turkey's E-Government website. Citizens could log in to their E-Government accounts securely and utilize the existing authentication mechanism to verify their eligibility as Voters.

By allowing flexibility in selecting the secure channel establishment, the ADSES protocol accommodates diverse election scenarios and aligns with the specific needs of different authorities or election organizers. This approach ensures the authentication process is integrated seamlessly, guaranteeing a secure and credible voting environment.

3.4.2 Election Phase

The Election phase is a critical component of the ADSES, where Voters generate their votes, perform necessary operations, and store them in the peer-to-peer distributed database. The overall process involves various steps, depicted in Figure 3.4, illustrating the flow of operations.

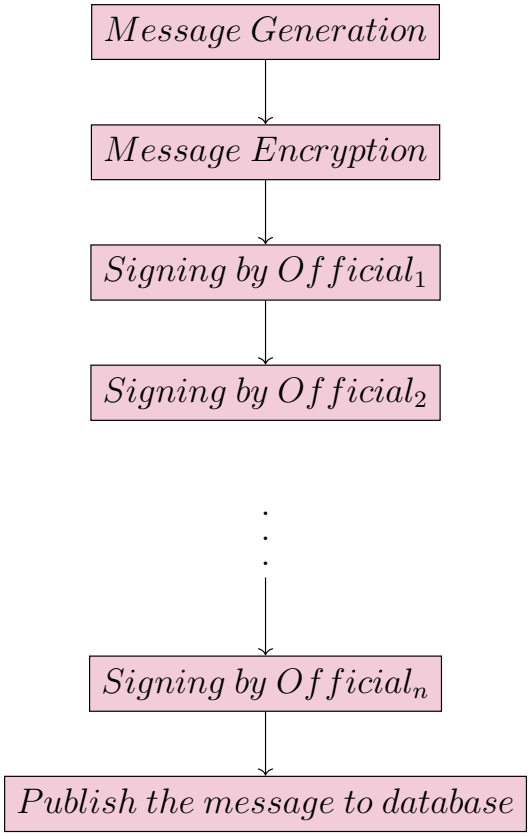

Figure 3.4: Diagram illustrating the sequential steps involved during the Election phase.

The first step is message generation, where the Voter creates their vote. This is followed by message encryption, ensuring the confidentiality of the vote. The following essential step is the signing process, which involves each Official sequentially signing the blinded vote. This ensures the authenticity and integrity of the message. The last step in this phase is publishing the signed message to the distributed database.

Figure 3.5 provides a high-level overview of the Election phase to illustrate the interactions further. The Voter generates, encrypts, and blinds the vote. The blinded vote is then forwarded to the Officials, who individually sign the blinded vote. After receiving the signed vote, the Voter unblinds it to reveal the original message without compromising privacy. Finally, the Voter adds the signed vote to the distributed database.

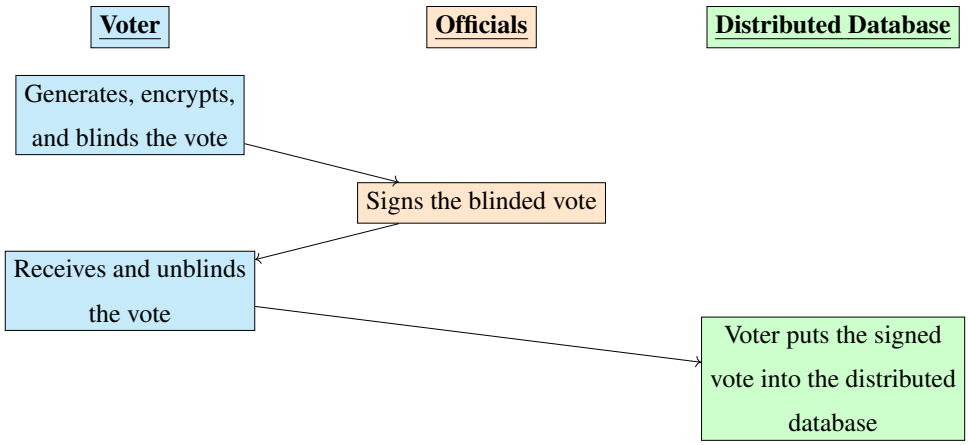

Figure 3.5: A diagram showing the sequence of operations carried out during the Election phase.

By following these steps, the Election phase guarantees the secure handling of votes, maintaining their privacy, authenticity, and integrity throughout the process.

3.4.2.1 Message Generation

In the Message Generation step, Voters decide on their votes and transform them into unique and verifiable messages. These messages will be encrypted, signed, and stored in the peer-to-peer distributed database.

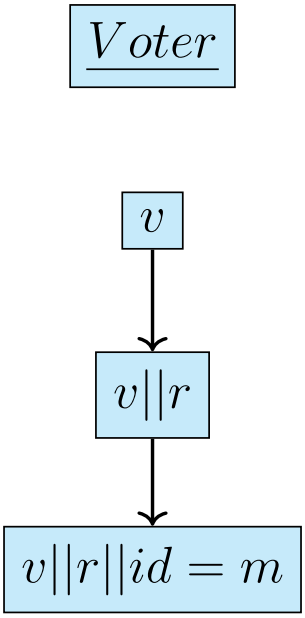

Figure 3.6: Diagram displays the message generation process during the Election phase.

Every Voter decides on their vote when the election starts, as shown in Figure 3.6.

Let us represent the vote of an individual as v. For example, let's assume we have two candidates in the election: Alice and Bob. The Authority can represent Their vote characters as A and B. For example, as a Voter, Mallory chooses A as her vote.

To ensure uniqueness, voters concatenate a random string r to their votes. This prevents malicious entities from quickly duplicating votes in the peer-to-peer distributed database.

$$v||r$$

For example, Mallory concatenates the random string KLD to her vote and generates $AKLD$.

Next, the Voter concatenates the IDs of all officials, sorted lexicographically by their names. These IDs will be used to validate votes at the end of the election.

$$v||r||id = m$$

Let's assume the Authority generates their ID as $XYYZ$, Alice generates their ID as $HTRJ$, and Bob generates their ID as $KQXM$. Then, Mallory's message becomes $AKLDXYYZHTRJKQXM$

As a result, the Voter generates the message by concatenating their vote, the random string, and the IDs of officials. This ensures that each message is unique and can be verified later.

3.4.2.2 Message Encryption

In the Message Encryption step, Voter encrypts their message using the encryption keys of all the officials involved in the election. This ensures no one can read the votes until the election is complete.

The Voter encrypts their message, denoted as m, using the encryption key of each Official. The encryption is performed consecutively, starting from the first Official's key, as shown in Figure 3.7.

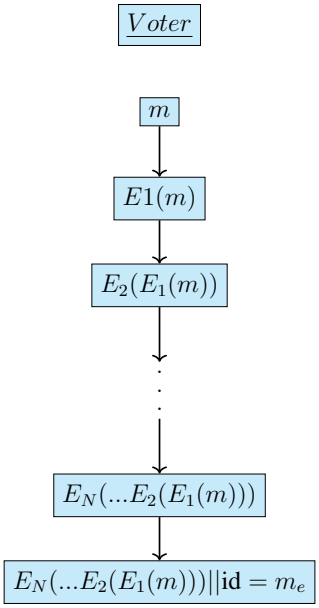

Figure 3.7: The diagram depicts the sequence of message encryption during the Election phase.

The resulting encrypted message is represented as

$$E_n(..E_2(E_1(m)))$$

Finally, the Voter appends the IDs of all officials to the encrypted message to create the final encrypted message, denoted as

$$E_n(..E_2(E_1(m)))||id = m_e$$

This ensures that each encrypted message is unique and can be verified and decrypted later in the post-election phase.

3.4.2.3 Message Signing

Message Signing process ensures the integrity and authenticity of the votes in the peer-to-peer distributed database. This process involves the Voter blind-signing their message and sequentially sending it to each Official for signature. The officials sign the blinded message, which the Voter unblinds to obtain the signed message without

exposing its content to the officials. The signed message is then stored in the peer-to-peer distributed database, where its validity can be verified.

It is essential to prevent officials from seeing the content of the message and making any connection between the message stored in the peer-to-peer distributed database and the messages sent to them.

To obtain the signature from the first Official, the Voter generates a blinding factor for the Official's signing key and blinds their message with this factor:

$$B_1(m_e)$$

The Voter sends the blinded message to the first Official, as shown in Figure 3.8. The Official signs the blinded message using their Signing key:

$$S_1(B_1(m_e))$$

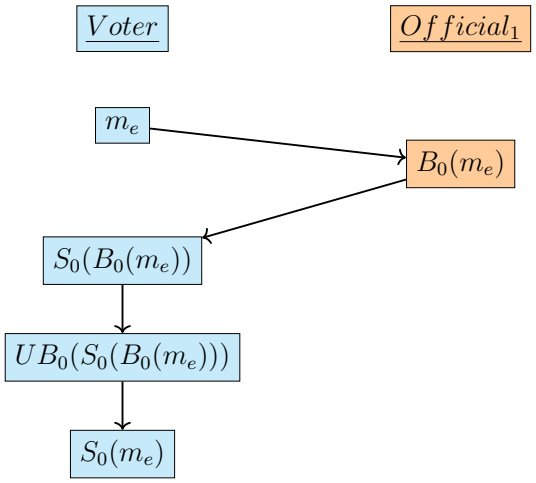

Figure 3.8: This illustrates the process of signing by $Official_1$ during the Election phase.

Then, the Official sends it back to the Voter. To verify the signature's validity, the Voter unsigns it and compares it to the blinded message:

$$US_1(S_1(B_1(m_e)))? = B_1(m_e)$$

Next, the Voter unblinds the message using the blinding factor, resulting in obtaining the signed message without exposing it to the Official:

$$UB_1(S_1(B_1(m_e))) = S_1((m_e))$$

Then, the Voter performs the same operations to obtain a signature from the second Official, as depicted in Figure 3.9.

$$S_2(S_1((m_e)))$$

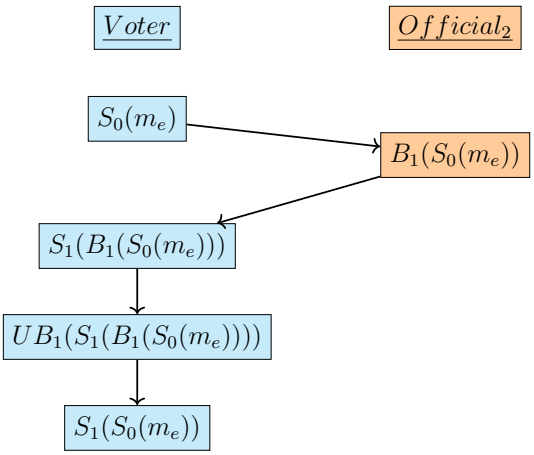

Figure 3.9: Diagram of the election process emphasizing $Official_2$'s participation in signing.

The Voter continues to obtain the signature from each Official one by one until the nth Official, as illustrated in Figure 3.10.

$$S_n(..S_2(S_1((m_e)))) = m_s$$

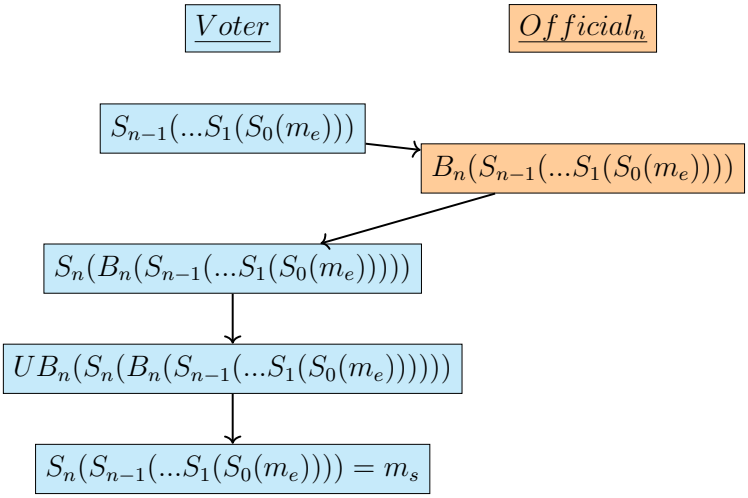

Figure 3.10: The election phase diagram illustrating $Official_n$'s involvement in signing.

3.4.2.4 Message Publishing

In the Message Publishing process, the Voter attempts to store the signed message in the peer-to-peer distributed database. Other nodes (public) verify the validity of the votes by performing the unsigning operation with the unsign key of each Official one by one in reverse order:

$$US_1(US_2(..US_n(m_s)))$$

Following that, they check if the resulting value ends with the IDs of all the Officials. If the value does not match the expected format, indicating that it does not end with the IDs of all officials, the signed message is removed from the peer-to-peer distributed database.

In conclusion, the Election phase ensures the integrity and authenticity of votes within a peer-to-peer distributed database. Through message signing, the Voter securely blinds and obtains signatures from Officials without revealing the content of the vote. The signed messages are then stored in the distributed database, where other nodes verify their validity. Any discrepancies or tampering attempts can be detected and

flagged by performing unsigning operations and checking for the presence of all Officials' IDs. This robust mechanism fortifies the integrity of the voting system, instilling confidence in the accuracy and trustworthiness of the collected votes. With a well-designed and implemented Election phase, the overall voting process can be robust and resilient, safeguarding democratic principles and ensuring a fair representation of the Voters' choices.

3.4.3 Post-Election

After the completion of the election phase, the Post-Election phase begins, which includes two main steps: Decryption Key Broadcasting and Calculating Results.

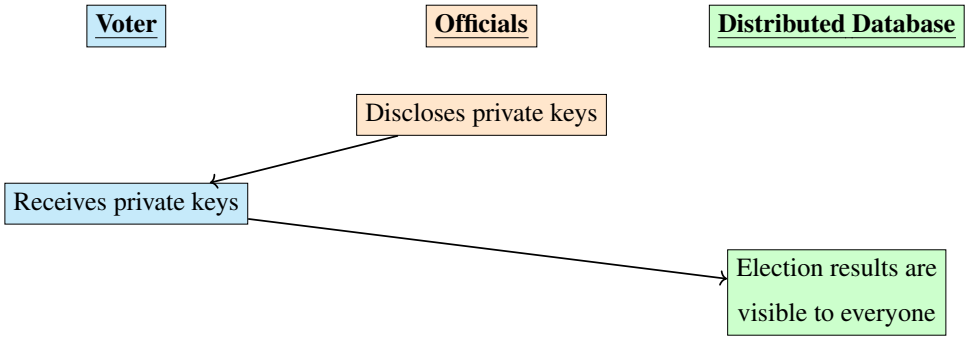

Figure 3.11: Post-election phase overview showing key transfer and results disclosure.

3.4.3.1 Decryption Key Broadcasting

In this step, Officials broadcast their Decryption keys to allow every node in the distributed database to decrypt the messages and reveal the election results. The broadcasting ensures that the decryption keys are readily available to all authorized entities.

3.4.3.2 Tallying

Once the decryption keys are broadcasted, anyone from the public can calculate the election results. The following steps are performed for each vote:

- The signed vote is unsigned by using each official's Unsign key one by one in reverse order.
- Starting from the end of the message, the IDs of all officials are discarded, resulting in the encrypted vote.
- The encrypted vote is decrypted using each Official's Decryption key one by one in reverse order.
- It is then checked if the message ends with the IDs of all officials. If not, the message is discarded.
- The IDs of all officials and the random string at the end of the message are discarded.
- Valid votes are collected, and the overall results are calculated based on these valid votes.

By employing a distributed database, the ADSES voting system ensures the security and privacy of votes while enabling transparency and verifiability. The Post-Election phase validates only legitimate votes and allows any public member to calculate the election results accurately. This process maintains the integrity of the voting system and instills confidence in the election's outcome.

3.5 Security Analysis

This section provides a security analysis of ADSES, focusing on how the system addresses the requirements outlined earlier in the methodology. We discuss each requirement and explain how ADSES ensures the election process's security, accuracy, and transparency.

3.5.1 Verifiability

Public Verifiability: In ADSES, public verifiability is achieved by allowing anyone to verify the election outcome. After the election concludes, any participant can independently confirm that the results announced by the Officials are accurate. This

feature is crucial for establishing trust in the election process and ensuring the reported results are legitimate.

Individual Verifiability: ADSES ensures individual verifiability by allowing voters to confirm that their votes have been accurately counted in the final tabulation. The system's design enables voters to verify the overall election results and the inclusion of their votes in calculating the outcome. Voters can confirm whether their vote, which they put in the peer-to-peer distributed database during the election phase, is still included and counted in the final tabulation. This feature enhances voter confidence in the election process, ensuring their votes have been accurately recorded and counted.

3.5.2 Anonymity

ADSES guarantees Voter anonymity by employing a combination of encryption, blind signatures, and secure communication channels. This approach ensures that no one, including officials, can link a vote to a specific voter. Anonymity is a critical requirement for a secure election process, as it allows voters to cast their votes without fear of retribution or coercion.

3.5.3 Authentication

The system ensures voter authentication by establishing a secure channel between Voters and Officials, wherein only authorized Voters can participate in the election. Officials define the eligible voter list before the election, and the system ensures that only those on the list can vote. This requirement is essential for maintaining the integrity of the election, as it prevents unauthorized votes from being cast and included in the final results.

3.5.4 Accuracy

Voting Once: ADSES ensures that no one can vote more than once by implementing unique authentication credentials for each voter and verifying their identity during

voting. This feature is critical for maintaining trust in the election, as voters can be confident that their vote carries equal weight as everyone else's.

No Duplication: The system prevents vote duplication by adding a random string to each vote, making them unique. This feature ensures that no one can duplicate someone else's vote, even if they cannot see the content of the vote. For example, it prevents scenarios where someone obtains an already signed vote from the distributed database and attempts to send it back as a new vote. Each vote in the distributed database is expected to be unique, further enhancing security. Preventing duplication is crucial for maintaining the integrity of the election, as it prevents malicious actors from manipulating the results.

No Change: ADSES employs cryptographic algorithms to ensure no one can change a vote without being discovered. Once a vote is placed in the system, it remains unchanged until the election results are announced. All votes are encrypted and signed, which means that the content of a vote cannot be altered while preserving its validity. A mutable data structure further ensures that votes cannot be changed once stored. This feature is critical for fostering trust in the election process, as it assures voters that their vote will not be tampered with after submission.

No Imitation: The system prevents vote imitation by verifying the authentication credentials provided by each voter and ensuring that they belong to the intended individual. Additionally, the system does not allow anyone to bypass the authentication step and cast a vote on behalf of someone else. This requirement is crucial for maintaining the integrity of the election, as it prevents unauthorized individuals from voting in place of legitimate voters.

3.5.5 Transparency

ADSES promotes transparency by allowing everyone to observe the entire election procedure. The system ensures that all steps of the election adhere to established standards, allowing everyone to view signed votes on the system during the election. After the election, everyone can see the unsigned and unencrypted votes without any voter relation and validate the results. Transparency is crucial for a secure election,

enabling voters to observe the process and trust the system.

3.5.6 Consistency

The system ensures consistency by requiring all participants involved in the election to maintain the same record of the voting procedure and accept the same outcome. Consistency is vital for a secure election, as it prevents disputes over the results and fosters trust in the system.

3.5.7 Dependability

ADSES guarantees dependability by utilizing cryptographic algorithms to protect the voting procedure against dishonest behaviors and attacks. This feature is critical for a secure election, as it assures voters that their votes are processed securely and that the system is resilient against potential threats.

3.5.8 No Intermediary Results

The system prevents the release of intermediary results by encrypting votes before officials sign them. This approach ensures that election results can only be known after the voting process has concluded, protecting voters from potential manipulation and ensuring that early results do not influence their choices.

3.5.9 Time Efficiency

ADSES achieves time efficiency by quickly revealing the results after the election concludes. This feature is essential for a secure election, as voters are eager to know the outcome as soon as possible. Additionally, announcing results promptly minimizes the window of opportunity for potential attackers to tamper with the system. Time efficiency helps build trust in the election process and increases voter satisfaction.

In summary, the security analysis of ADSES demonstrates that the system effectively addresses the critical requirements for a secure, accurate, and transparent election process. By ensuring verifiability, anonymity, authentication, accuracy, transparency, consistency, dependability, protection against intermediary results, and time efficiency, ADSES provides a robust and reliable e-voting protocol that fosters trust and confidence in the election process.

3.6 Attack Vectors and Countermeasures in ADSES

In this section, we discuss the possible attack vectors that ADSES may face and the countermeasures taken by the system to mitigate these risks.

3.6.1 Vote Tampering

Attack: An attacker may attempt to modify the votes recorded on the distributed database to influence the election results.

Countermeasure: ADSES employs a combination of encryption, signing, and storing in a peer-to-peer distributed database to ensure the integrity of votes. Each vote is encrypted using the encryption keys of all officials and signed individually. The distributed database, coupled with a consensus mechanism, verifies the authenticity of votes and prevents any attempts to tamper with them.

3.6.2 Fake Votes

Attack: An attacker may attempt to introduce fake votes into the system to skew the election results.

Countermeasure: The ADSES system requires that each vote be signed by all Officials using their unique Signing keys. This ensures that the system accepts only valid votes cast by authenticated Voters. Any fake votes will not have valid signatures and will be rejected by the nodes in the distributed database.

3.6.3 Voter Privacy Violation

Attack: An attacker may attempt to compromise voter privacy by associating votes with an individual or a group of voters.

Countermeasure: ADSES utilizes blind signatures and encryption to protect voter privacy. Each voter's vote is encrypted using the encryption keys of all Officials and then blinded before being individually signed by each Official. This process ensures that officials cannot associate the vote with the specific voter. Furthermore, including a random string attached to each vote enhances privacy by guaranteeing the uniqueness of each vote, making it challenging to link multiple votes to any specific group of voters.

3.6.4 Denial of Service [1]

Attack: An attacker may attempt to flood the system with invalid votes or transactions, causing a denial of service and preventing legitimate votes from being processed.

Countermeasure: The distributed database's consensus mechanism and the requirement for valid signatures from all Officials help mitigate the risk of denial of service attacks. Additionally, the peer-to-peer distributed nature of the database makes it more resilient against such attacks. The network automatically rejects invalid votes or transactions, ensuring the system can process legitimate votes without interruptions.

3.6.5 Collusion Among Officials

Attack: A group of Officials may conspire to manipulate the election results or compromise voter privacy.

Countermeasure: In ADSES, Officials are chosen from each candidate, such as different political parties or public officers, in case of a national election. This makes collaboration among Officials less likely. Furthermore, the use of blind signatures

and encryption ensures that even if a group of Officials was to collude, they would not be able to compromise voter privacy or alter the election results without being detected by the consensus mechanism among nodes in the distributed database.

3.6.6 Sybil Attack [2]

Attack: In a Sybil attack, a malicious actor creates multiple fake identities to gain control or influence over the network. In the context of ADSES, this could involve an attacker creating multiple fake voter identities to cast illegitimate votes.

Countermeasure: ADSES requires voters to be registered and authenticated by officials, ensuring that each voter has a unique and valid identity. This process helps prevent Sybil attacks by ensuring that only legitimate voters can participate in the election.

3.6.7 Man-in-the-Middle (MITM) Attack [3]

Attack: A man-in-the-middle (MITM) attack occurs when a malicious actor intercepts communication between two parties and can read, modify, or inject new messages. In the context of ADSES, a MITM attack could target the communication between voters and officials or nodes in the distributed database.

Countermeasure: ADSES employs end-to-end encryption and digital signatures to secure the communication between voters and officials. This ensures that an attacker cannot read or modify any intercepted messages. Furthermore, securing the communication between nodes in the distributed database can help prevent MITM attacks targeting the database.

3.6.8 Vote-Buying and Coercion

Attack: Vote-buying and coercion involve a malicious actor attempting to influence a voter's choice by offering incentives or applying pressure. In the context of ADSES, this could involve an attacker trying to determine a voter's choice based on the

encrypted vote data.

Countermeasure: ADSES ensures vote privacy using encryption and blind signatures. This makes it practically impossible for an attacker to determine the content of a vote without access to the decryption keys of all officials. Furthermore, the random string added to each vote during the message generation step ensures that even if an attacker could decrypt the vote, they could not link it back to a specific voter, thereby protecting against vote-buying and coercion.

3.7 Cryptographic Features and Guarantees in ADSES

In this section, we discuss the cryptographic features [44] and guarantees provided by the ADSES system to ensure the security, privacy, and integrity of the electronic voting process.

3.7.1 Forward Secrecy

Definition: Forward secrecy is a property of cryptographic systems that ensures the compromise of a long-term key does not compromise past session keys or the data encrypted with those session keys.

ADSES Implementation: In ADSES, each voter generates a random string (r) and appends it to their vote. This randomization ensures that even if an Official's encryption key is compromised in the future, the attacker cannot determine the original vote by analyzing the encrypted data. Moreover, since the vote is encrypted using the encryption keys of all Officials, the attacker would need to compromise all Officials' keys to reveal the original vote, further enhancing forward secrecy.

3.7.2 Backward Secrecy

Definition: Backward secrecy, also known as future secrecy, is a property of cryptographic systems that ensures the compromise of a long-term key does not compromise the security of future session keys or the data encrypted with those session keys.

ADSES Implementation: ADSES achieves backward secrecy by using unique encryption and signing key pairs for each election. If an Official's key is compromised after an election, the attacker cannot use it to manipulate future elections. Additionally, selecting Officials from various independent entities reduces the risk of a single point of failure and further strengthens backward secrecy.

3.7.3 Key Authenticity

Definition: Key authenticity is the property of cryptographic systems that ensures the keys used for encryption and signing are genuine and belong to their respective owners.

ADSES Implementation: In ADSES, key authenticity is achieved by having each Official generate their encryption, signing key pairs, and broadcast their public keys before the election. This allows voters and other nodes in the distributed database to verify the authenticity of the keys used to encrypt and sign votes.

3.7.4 Non-repudiation

Definition: Non-repudiation is a property of cryptographic systems that ensures a party cannot deny the authenticity of their signature on a document or the transmission of a message.

ADSES Implementation: Non-repudiation is achieved in ADSES through digital signatures. When an Official signs a vote, they use their private signing key to generate a unique signature. Anyone can verify this signature using the Official's public Unsigning key. Since the private Signing key is known only to the Official, they cannot deny signing the vote.

3.7.5 Unforgeability

Definition: Unforgeability is a property of cryptographic systems that ensures it is computationally infeasible for an attacker to create a valid signature or encryption

without possessing the appropriate private key.

ADSES Implementation: ADSES ensures unforgeability through public-key cryptography for encryption and digital signatures. Since the private keys are known only to their respective owners (Officials), it is computationally infeasible for an attacker to create a valid signature or encryption without access to the private key.

3.7.6 Data Integrity

Definition: Data integrity is a property of cryptographic systems that ensures the accuracy and consistency of data over its lifecycle. In the context of ADSES, data integrity ensures that votes are not tampered with or altered during the election process.

ADSES Implementation: ADSES achieves data integrity through digital signatures and the distributed database. When a voter casts their vote, the vote is signed by the voter and all officials involved. Any attempt to tamper with the vote will invalidate the signatures, making it evident that the data has been altered. Additionally, the distributed nature of the database ensures that multiple copies of the data exist, further safeguarding against tampering.

3.7.7 Authentication

Definition: Authentication is a property of cryptographic systems that ensures the identity of the entities involved in the process. In the context of ADSES, authentication ensures that only legitimate voters can participate in the election and that the officials are genuine.

ADSES Implementation: ADSES achieves authentication by requiring Voters to be registered and authenticated by Officials before participating in the election. This ensures that each voter has a unique and valid identity. Furthermore, Officials are authenticated using their digital signatures and public keys, guaranteeing they are genuine and authorized to participate in the election process.

CHAPTER 4

IMPLEMENTATION

This chapter provides a detailed overview of the implementation of ADSES [45] and the conducted experiments. It begins with the practical implementation of ADSES, including the technologies utilized, the testing environment's deployment, and the validation presentation with experimental results.

4.1 Practical Implementation of ADSES

ADSES was implemented by developing two web applications, the Official and Voter Web Apps. These applications were designed to ensure practical applicability and have been made available as open-source projects [46]. By being open-source, the public is provided with the opportunity to examine and verify the functionality and security features of the applications.

4.1.1 Official Web App

The Official Web App serves as a platform for election authorities and party inspectors to oversee the election process. It empowers these individuals to generate encryption and decryption keys and distribute them as needed throughout the various stages of the election.

To ensure transparency and trust, the Official Web App is designed to be easily deployed and operated on the systems of each authority and inspector. This enables multiple parties to independently verify and validate the election process, promoting a robust and reliable electoral system.

4.1.1.1 API Endpoints

Key Pairs: The Voter Web App utilizes this endpoint to obtain the encryption and signing keys of the Official. Initially, only the public parts of the keys are accessible. After the election concludes, the private parts become available as well.

Sign a vote: This endpoint is used by the Voter Web App to request a blind signature for a submitted vote.

Votes: The Votes endpoint facilitates the submission of encrypted and signed votes from the Voter Web App to the Official Web App. It is important to note that the original ADSES design incorporates a peer-to-peer distributed database to securely and transparently store all votes. However, due to the unavailability of a widely accessible peer-to-peer distributed database, we could not implement it as intended. Therefore, to demonstrate the complete implementation of ADSES, we utilized an alternative API endpoint and Relational Database to store votes.

Result: Once the election concludes and decryption keys are distributed, this endpoint provides access to the final election results.

4.1.2 Voter Web App

The Voter Web App is designed to ensure a smooth and secure voting experience for eligible voters. It guides voters through submitting their votes and allows them to verify their votes. The web application incorporates an intuitive user interface to simplify voting and encourage voter participation.

With the Voter Web App, voters can confidently cast their votes, knowing that their anonymity is preserved and their votes are securely recorded in the database.

4.1.2.1 Web pages

Vote: This page enables voters to submit their votes securely. The process involves selecting a candidate, appending a random string to the vote, blinding it, forwarding it to the authorities for authentication and signature, and finally submitting it to the

database.

4.2 Used Technologies

This section explores the various technologies utilized in the development of our solution. We begin with an overview of Python 3, highlighting its high-level capabilities and extensive library ecosystem. Next, we discuss the Python Cryptography Toolkit (PyCryptodome) and its role in seamlessly handling cryptographic operations. We then delve into implementing RSA, a widely-used public-key cryptography system known for its security and versatility. Additionally, we examine the Django Rest Framework, which played a vital role in developing a flexible and extensible API. Lastly, we showcase the utilization of Amazon Web Services (AWS), specifically CloudFormation, EC2, and Route53, for deploying and managing our solution in the cloud.

4.2.1 Python

Python3 [47] was chosen as the programming language for its high-level capabilities, enabling us to focus on core concepts without getting bogged down in low-level details. Additionally, Python3's extensive library ecosystem greatly facilitated our implementation of cryptographic operations.

4.2.2 Python Cryptography Toolkit

We used the Python Cryptography Toolkit, PyCryptodome [48], to seamlessly handle cryptographic operations. This toolkit allowed us to focus on the design and functionality of our solution without the need for intricate knowledge of cryptographic implementations.

4.2.3 RSA

RSA [49], a widely-used public-key cryptography system known for its security and versatility, was employed in our implementation for signing and encryption operations. Leveraging RSA ensured a solid and reliable cryptographic foundation for our system.

4.2.4 Django Rest Framework

The Django REST Framework [50] was instrumental in developing a flexible and extensible API for our solution. By utilizing this framework, we could easily create RESTful web services. The Django REST Framework empowered us to implement a modular, scalable, and maintainable API.

4.2.5 AWS

We utilized Amazon Web Services (AWS) [51] as our cloud provider for deploying our solution. AWS offers a comprehensive suite of services that played a vital role in implementing and deploying our system. Specifically, the following AWS services were employed:

4.2.5.1 Cloudformation

AWS CloudFormation [52] enabled the definition and provisioning of the required infrastructure resources for our solution. Leveraging CloudFormation templates allowed us to automate the creation and management of our infrastructure, ensuring a consistent and reproducible environment.

4.2.5.2 EC2

Amazon Elastic Compute Cloud (EC2) [53] provided the scalable compute resources necessary for running our application.

4.2.5.3 Route53

Amazon Route 53 [54] was used for managing the DNS records of our application. This service provided a reliable and scalable domain name system (DNS) solution, seamlessly integrating our custom domain name with our application. Route 53 enhanced the discoverability and usability of our solution.

4.3 Testing Environment Deployment

The deployment process for the testing environment of our ADSES implementation involves several steps to ensure a smooth and error-free setup.

Creation of Artifacts: We create two zip artifacts, one for the Official Web App and another for the Voter Web App. These artifacts contain all the files and configurations required to run the applications.

Publishing on AWS S3: The artifacts, along with the associated cloud templates, are published on a public file storage service provided by AWS called S3. This step simplifies the process for officials and authorities to download the required files and initiate deployment.

Official Web App Deployment: Officials can download and run the Official Web App artifact on their local systems. This ensures that the Official Web App is deployed correctly and ready for use.

Voter Web App Deployment: In a similar manner, authorities can download and run the Voter Web App artifact on their respective systems. This ensures the Voter Web App is correctly deployed and ready for use.

By following this deployment process, we ensure that the testing environment is set up correctly and that the applications are running as intended. This approach helps identify and resolve potential issues before election day, ensuring a smooth and error-free voting experience for all eligible voters.

4.4 Validations with Experiments

Two critical stages that may create a bottleneck in the election process are vote signing and result announcement. Analysis of the requirements for these stages is crucial. Additionally, the efficiency and promptness of these stages are vital considerations.

For our experiments, we selected AWS EC2 c6i instances known for their efficient compute optimization. The table 4.1 details the pricing and CPU counts for the various c6i instance types we assessed during our experimentation process [55]. Please note that the prices and CPU counts provided are valid as of May 2023.

Considering these specifications when determining the appropriate EC2 instance type is essential based on the use case's compute requirements and budget constraints.

Table 4.1: CPU counts and pricing for different EC2 instance types (as of May 2023).

Instance Type	**CPU Count**	**Price per Hour**
c6i.large	2	$0.085
c6i.xlarge	4	$0.170
c6i.2xlarge	8	$0.340
c6i.4xlarge	16	$0.680
c6i.8xlarge	32	$1.360
c6i.12xlarge	48	$2.040
c6i.16xlarge	64	$2.720

4.4.1 Validation of Result Announcement Time Change with Voter Count

The first experiment aimed to determine the time required to announce results based on the number of votes counted. To achieve this, a Python script was developed with multi-processing capabilities, utilizing all CPUs of the machines. The experiment was conducted on AWS EC2 c6i instances known for their compute optimization. Seven different instance types were used to investigate the impact of CPU counts on result announcement time. Five vote counts were tested: 1, 10, 100, 1000, and 10000. Each case was tested 100 times to ensure reliability, resulting in 35 cases.

The experiment's results indicated that the time required to count votes and announce results varied significantly based on the vote count and CPU count. For example, on a c6i.large machine with 2 CPUs, it took approximately 84 seconds to count 10000 votes, while on a c6i.16xlarge machine with 64 CPUs, it took around 11 seconds for the exact vote count.

The pricing of c6i machines is directly proportional to their CPU counts. For example, the c6i.large machine, which has 2 CPUs, is priced at 0.085 USD per hour, while the cgi.16xlarge machine, which has 32 times as many CPUs (64 CPUs), is priced at precisely 32 times the rate of the c6i.large machine, which is 2.72 USD per hour.

These findings show that results can be obtained quickly and at minimal costs, even for large-scale elections involving 100 million people. The accessibility of all votes in the database allows for their efficient distribution across multiple machines, enabling parallel evaluation.

Considering the performance of the c6i.16xlarge machine, which can evaluate 10,000 votes in 11 seconds, we can estimate that it can determine approximately 3,272,727 votes in an hour (3600/11 * 10000). Utilizing 31 machines simultaneously allows evaluating 100 million votes to be completed within an hour. Remarkably, the cost for this process is a mere $84.32, which is remarkably low in comparison to the scale of the election. Consequently, this approach can be employed by individuals or organizations to validate the election's integrity.

In addition, by running ten c6i.large machines simultaneously, each counting 1000 votes, a total count of 10000 votes can be achieved in 8.4 seconds, which is faster than the 11 seconds required by a single c6i.16xlarge machine counting the same number of votes. However, it should be noted that although most of the process can be multi-processed, some tasks still rely on the main thread. Further experimentation with different cases is needed to optimize the process thoroughly.

The experiment results are visualized in Figure 4.1 and Figure 4.2. Figure 4.1 demonstrates the evaluation time for 10000 votes based on the number of CPUs used. As depicted, increasing the CPU count results in a reduction in evaluation time. The graph shows a downward trend in evaluation time as the CPU count rises. The eval-

uation time decreased from 84.09 seconds using 2 CPUs to 11.23 seconds when 64 CPUs were used. This observation indicates that utilizing more CPUs can significantly reduce the time required to evaluate many votes.

Figure 4.2 shows the time for counting votes for different numbers of total votes. The number of votes is displayed on a log axis due to the exponentially increasing number of votes chosen for the experiment.

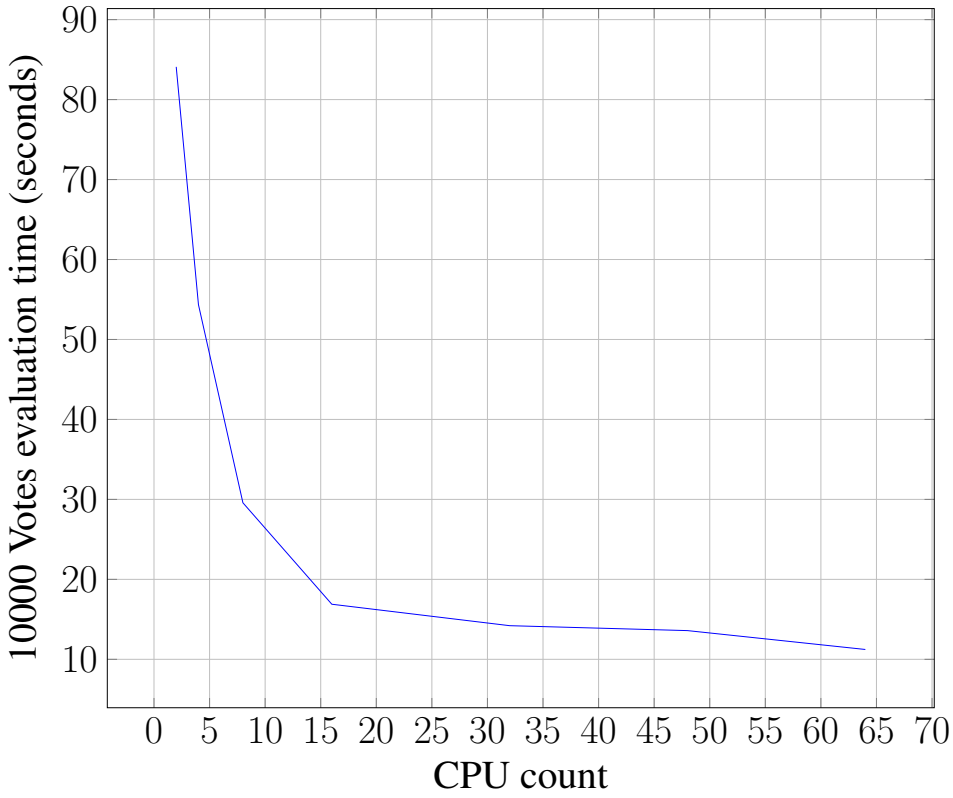

Figure 4.1: Correlation between CPU count and evaluation time for 10,000 votes.

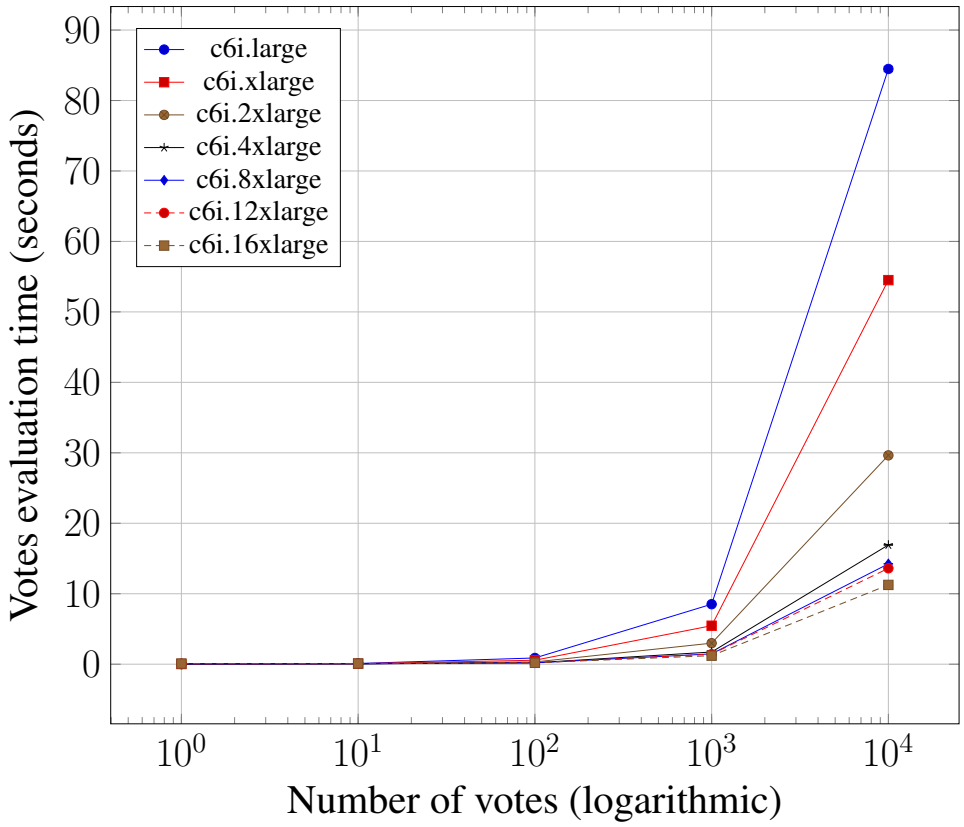

Figure 4.2: Evaluation time of votes with different instance types.

Experiment 1 evaluated the performance of different c6i instance types regarding signing votes per second. The results showed that all instance types performed similarly in our experiments. Although there was a linear trend in the graph until 100 Virtual Users (VUs), we could not reach higher signed votes per second beyond that point.

The experiments indicated a capacity limit of around 100 VUs, beyond which the signed votes per second did not increase significantly. To measure higher capacities, further experimentation with varying machine setups is necessary. This could involve exploring different instance types, sizes, or configurations to optimize the system's performance and achieve higher throughput.

4.4.2 Validation of Total Vote Signing Time Change with Voter Count

The second experiment explored the relationship between the total time required for all voters to sign their votes and the number of voters. The k6 load testing tool [56] [57] was utilized to execute multiple signing requests to the sign endpoint and measure response times. AWS EC2 c6i instances, known for their compute optimization, were used for the experiment to examine the impact of CPU counts on the total vote signing time. Seven different instance types were used to assess the effect of CPU counts on the whole vote signing time. Five different voter counts were tested: 1, 10, 100, 1000, and 10000, resulting in 35 tests lasting 30 seconds each to ensure reliability.

The k6 load testing tool proved invaluable for this experiment as it allowed us to simulate thousands of virtual users and measure the application's response time. By setting the virtual user count and time duration, we could replicate the behavior of actual users sending requests to sign their votes.

The k6 script was set up for each experiment with a specific virtual user count [58], representing the desired number of simultaneous requests to the sign endpoint. For voter counts of 1, 10, and 100, the experiments yielded very similar numbers of signed votes per second. For example, with 100 virtual users using the c6i.large instance, we achieved a rate of signing 97.2 votes per second. However, for voter counts of 1000 and 10000, the experiments did not reach the target vote counts per second. Nevertheless, the results obtained using 100 virtual users were sufficient to demonstrate that signing 100 million votes within a short time frame is feasible.

Using the c6i.16xlarge instance, we could sign 97.33 votes simultaneously with an average response time of 7.78 ms per vote. This implies that within one hour, approximately 45037018 votes can be signed (calculated as 60601000/7.78 * 97.33). Consequently, signing 100 million votes would take around 2.22 hours (calculated as 100000000/45037018). The cost for this process would be approximately 6 USD (calculated as 2.22 * 2.72), which is relatively low compared to the scale of the election involving 100 million voters.

The following table 4.2 presents the outcomes of 35 experiments, where seven differ-

ent instance types and five virtual user counts were employed. In each experiment, the k6 script generated virtual users, and each virtual user sent vote-signing requests every second. The table displays the average number of votes sent in each experiment and their corresponding response time. The table's header shows the number of virtual users used in each experiment.

Table 4.2: Experiment results showing vote signing capacity across various instance types.

Instance Type	**1 Virtual User**	**10 Virtual Users**	**100 Virtual Users**	**1000 Virtual Users**	**10000 Virtual Users**
c6i.large	0.99 votes in 7.71 ms	9.9 votes in 16.53 ms	97.2 votes in 22.86 ms	167.66 votes in 4680 ms	156.15 votes in 5420 ms
c6i.xlarge	0.99 votes in 7.24 ms	9.9 votes in 11.52 ms	97.34 votes in 13.82 ms	170.98 votes in 4540 ms	186.32 votes in 5340 ms
c6i.2xlarge	0.99 votes in 7.35 ms	9.9 votes in 13.56 ms	97.11 votes in 13.35 ms	156.72 votes in 4950 ms	167.68 votes in 5650 ms
c6i.4xlarge	0.99 votes in 7.36 ms	9.9 votes in 10.57 ms	97.28 votes in 13.63 ms	170.72 votes in 4650 ms	166.99 votes in 5390 ms
c6i.8xlarge	0.99 votes in 7.29 ms	9.92 votes in 8.29 ms	97.24 votes in 23.99 ms	162.53 votes in 4710 ms	179.01 votes in 5280 ms
c6i.12xlarge	0.99 votes in 7.12 ms	9.9 votes in 11.01 ms	97.5 votes in 14.02 ms	182.59 votes in 4290 ms	196.72 votes in 5080 ms
c6i.16xlarge	0.99 votes in 6.99 ms	9.9 votes in 8.22 ms	97.33 votes in 7.78 ms	169.81 votes in 4420 ms	179.92 votes in 5360 ms

Our experiments with 100 Virtual Users showed that the ADSES signing phase is a feasible solution for real-life scenarios. The table 4.3 provides detailed response times statistics for each sign operation, including the Average, Minimum, Median, Maximum, 90th percentile, and 95th percentile. All instance types tested were capable of concurrently signing approximately 97 votes. The table displays a variation in response times for each instance type; however, there seems to be an overall trend of decreasing response times as the CPU count increases across all instances. However, the difference in response times is inconsistent, as seen in the table. Further experimentation with extended duration times is required to identify the precise relationship between CPU count and response times. Despite this, it is evident that even the worst p95 results obtained are still feasible.

Table 4.3: Vote signing times across various instances for 100 virtual users.

Instance Type	Average (ms)	Minimum (ms)	Median (ms)	Maximum (ms)	P90 (ms)	P95 (ms)
c6i.large	17.63	5.61	6.1	607	9.29	22.86
c6i.xlarge	16.68	5.36	6.13	567.58	8.05	13.82
c6i.2xlarge	17.93	5.41	6.54	621.14	7.87	13.35
c6i.4xlarge	17.04	5.44	6.37	576.44	8.92	13.63
c6i.8xlarge	17.18	5.28	6.57	554.05	14.29	23.99
c6i.12xlarge	15.76	5.32	5.88	545.58	7.06	14.02
c6i.16xlarge	16.47	5.78	6.43	589.12	7.29	7.78

Figure 4.3 shows the signed votes per second for seven c6i instance types at varying Virtual User counts. Both x and y coordinates are logarithmic as we expect exponential increases. Results show that all instance types performed similarly in our experiments. The graph follows a linear trend until 100 Virtual Users, beyond which we could not reach higher signed votes per second. Further experimentation with varying machine setups is necessary to measure higher capacities.

Figure 4.4 depicts the relationship between p95 response times and the number of Virtual Users (VUs) in our experiments. Both axes are logarithmic due to the exponential increase of VUs and p95 response times. The graph demonstrates that the current experiments had a capacity limit of around 100 VUs, beyond which the p95 response times increase exponentially. Consequently, further experimentation with varying machine setups is essential to measure higher capacities. The legend indicates the different machine setups used in the experiment, namely c6i.large, c6i.xlarge, c6i.2xlarge, c6i.4clarge, c6i.8xlarge, c6i.12xlarge, and c6i.16xlarge.

Experiment 2 focused on analyzing the response times for each sign operation, specifically looking at the p95 response times for different c6i instance types and VU counts. The experiment results demonstrated an exponential increase in p95 response times as the number of VUs increased.

The findings revealed that the experiments had a capacity limitation of around 100 VUs, beyond which the p95 response times increased significantly. These results signify the need for further experimentation with varying machine setups to accurately

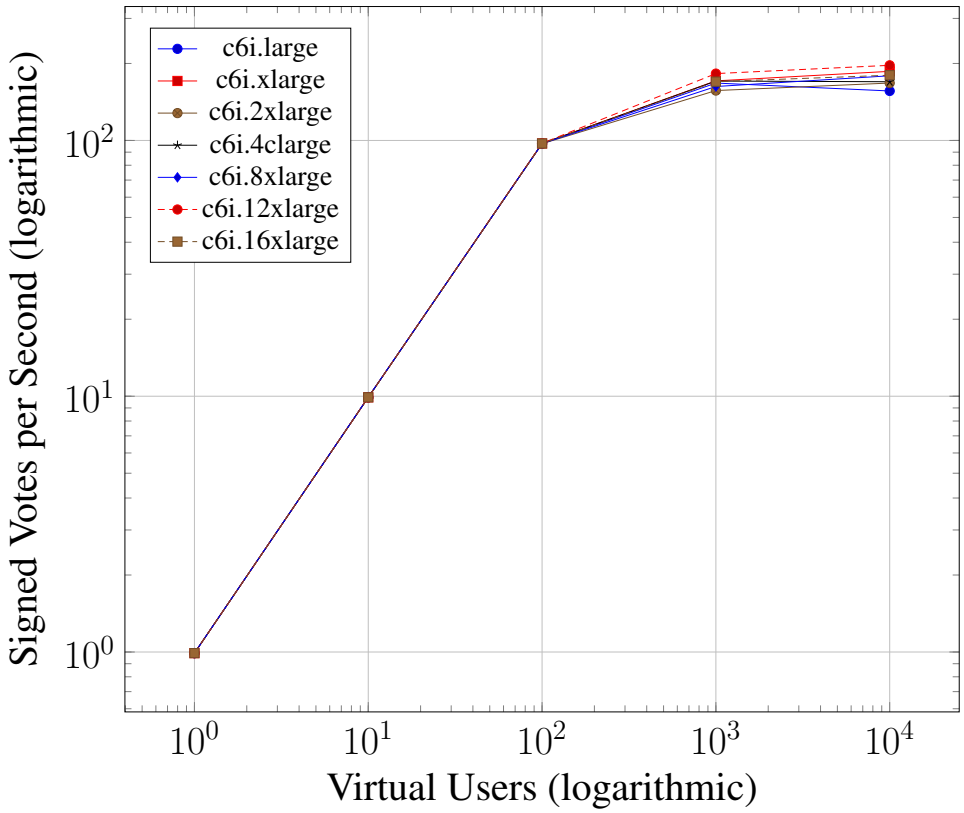

Figure 4.3: Logarithmic comparison of signed votes per second against virtual user counts.

measure higher capacities and optimize response times in scenarios with larger VU counts.

CPU count also played a role in the experiment, as increasing the CPU count generally led to decreased response times. However, the differences in response times were inconsistent across all instance types, emphasizing the necessity for further investigation into the relationship between CPU count and response times.

4.5 Conclusion

In conclusion, our experiments focused on evaluating the feasibility and performance of the ADSES signing phase in real-life scenarios. The experiments involved testing different c6i instance types and varying Virtual User (VU) counts.

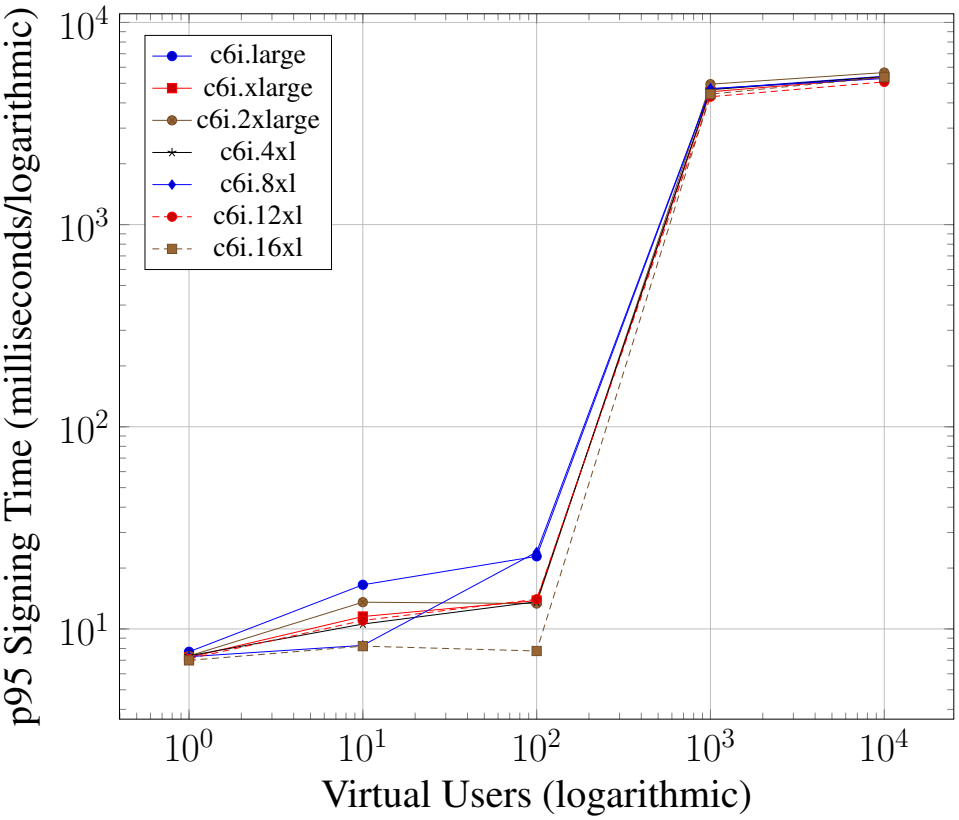

Figure 4.4: p95 Signing Time vs. Virtual User Counts

The results indicated that the ADSES signing phase is a viable solution for real-life scenarios, as all instance types performed similarly regarding signed votes per second. However, we observed a capacity limitation of around 100 VUs, beyond which the p95 response times increased exponentially.

Despite the increase in response times, even at the worst p95 results obtained, the ADSES signing phase remained feasible. However, further experimentation with varying machine setups is necessary to measure higher capacities and optimize the response times for larger VU counts.

The experiments also highlighted the importance of considering CPU count. While there was a trend of decreasing response times as the CPU count increased across all instance types, the difference in response times was inconsistent. Therefore, additional experimentation with extended duration times is required to identify the precise relationship between CPU count and response times.

The findings suggest that the ADSES signing phase can deliver efficient performance in real-life scenarios. However, further investigation and optimization measures are needed to achieve higher capacities and optimize response times, including experimenting with different machine setups and potentially exploring parallelization techniques.

CHAPTER 5

CONCLUSION

5.1 Conclusions

In this thesis, we presented ADSES, a decentralized and secure election system that leverages the power of distributed databases to enhance the security, transparency, and efficiency of electronic voting. ADSES is designed to be resilient against attacks aimed at manipulating votes and ensures that voters can observe the entire election process.

We outlined the requirements for a secure election and demonstrated how ADSES addresses each challenge. By utilizing distributed databases, ADSES offers a robust and secure solution that fulfills these requirements, ensuring the integrity and reliability of the voting process.

As demonstrated in this study, distributed databases hold great potential to improve various aspects of our lives. Further research is needed to explore the integration of distributed databases into diverse fields, thereby unlocking their potential benefits.

5.2 Future Work

Looking forward, there are several avenues for future research and development of ADSES. One potential direction is the exploration of a fully decentralized election system. Although ADSES is decentralized, the power is currently divided among a limited number of participants rather than distributed among millions of voters. With the continued advancement of distributed databases and other emerging technologies,

a fully decentralized election system could become a reality, further enhancing security, transparency, and trust in the democratic process.

In addition to pursuing a fully decentralized election system, other areas of future work may include refining the cryptographic features of ADSES to improve its security and privacy and exploring methods for scalability to accommodate larger populations and more complex voting scenarios. Moreover, it would be worthwhile to investigate further the integration of ADSES with other technologies, such as secure multiparty computation [59] and zero-knowledge proofs [60], to enhance the privacy and verifiability of the voting process.

Another area of future work is to develop user-friendly interfaces and tools that can facilitate the adoption of ADSES by election authorities and voters. This includes creating secure voting applications, providing educational resources, and offering training for election officials and voters to ensure the smooth and successful implementation of ADSES in real-world elections.

Additionally, further research could be conducted to analyze the legal and regulatory aspects of implementing ADSES in various jurisdictions. Understanding the legal requirements and potential barriers to adoption will be crucial for successfully deploying the system in different countries and regions.

Lastly, conducting pilot studies and real-world trials of ADSES would be beneficial to evaluate its performance, usability, and overall effectiveness in elections. This would provide valuable insights and feedback that can be used to refine and improve the system, ultimately contributing to the development of a more secure, transparent, and trustworthy democratic process for all.

REFERENCES

[1] S. "Yu, "An Overview of DDoS Attacks". New York, NY: Springer New York, 2014. [Online]. Available: https://doi.org/10.1007/978-1-4614-9491-1_1

[2] J. R. Douceur, "The sybil attack," in Peer-to-Peer Systems, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[3] "Manipulator-in-the-middle attack | OWASP Foundation — owasp.org," https://owasp.org/www-community/attacks/Manipulator-in-the-middle_attack, [Accessed 07-08-2023].

[4] B. Manin, The Principles of Representative Government, ser. Themes in the Social Sciences. Cambridge University Press, 1997.

[5] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach, "Analysis of an electronic voting system," in IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004, 2004, pp. 27–40.

[6] G. Schryen and E. Rich, "Security in large-scale internet elections: A retrospective analysis of elections in estonia, the netherlands, and switzerland," IEEE Transactions on Information Forensics and Security, vol. 4, no. 4, pp. 729–744, 2009.

[7] J. Bannet, D. Price, A. Rudys, J. Singer, and D. Wallach, "Hack-a-vote: Security issues with electronic voting systems," IEEE Security & Privacy, vol. 2, no. 1, pp. 32–37, 2004.

[8] A. Bonifati, P. K. Chrysanthis, A. M. Ouksel, and K.-U. Sattler, "Distributed databases and peer-to-peer databases: Past and present," SIGMOD Rec., vol. 37, no. 1, p. 5–11, mar 2008. [Online]. Available: https://doi.org/10.1145/1374780.1374781

[9] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan, "Making data structures persistent," in Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, ser. STOC '86. New York, NY, USA: Association for Computing Machinery, 1986, p. 109–121. [Online]. Available: https://doi.org/10.1145/12130.12142

[10] 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, no. 1, pp. 119–130, 2022.

[11] S. Chaudhary, S. Shah, R. Kakkar, R. Gupta, A. Alabdulatif, S. Tanwar, G. Sharma, and P. N. Bokoro, "Blockchain-based secure voting mechanism underlying 5g network: A smart contract approach," IEEE Access, vol. 11, pp. 76 537–76 550, 2023.

[12] V. Buterin, "A next-generation smart contract and decentralized application platform," January 2014.

[13] "Solidity 2014; Solidity 0.8.22 documentation — docs.soliditylang.org," https://docs.soliditylang.org/en/develop/, [Accessed 09-08-2023].

[14] G. Rathee, R. Iqbal, O. Waqar, and A. K. Bashir, "On the design and implementation of a blockchain enabled e-voting application within iot-oriented smart cities," IEEE Access, vol. 9, pp. 34 165–34 176, 2021.

[15] F. P. Hjalmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjalmtysson, "Blockchain-based e-voting system," in Proc. of the IEEE 11th International Conference on Cloud Computing (CLOUD), vol. 00, Jul 2018, pp. 983–986. [Online]. Available: doi.ieeecomputersociety.org/10.1109/CLOUD.2018.00151

[16] B. Shahzad and J. Crowcroft, "Trustworthy electronic voting using adjusted blockchain technology," IEEE Access, vol. 7, pp. 24 477–24 488, 2019.

[17] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in International conference on financial cryptography and data security. Springer, 2017, pp. 357–375.

[18] W.-J. Lai, Y.-C. Hsieh, C.-W. Hsueh, and J.-L. Wu, "Date: A decentralized, anonymous, and transparent e-voting system," in 2018 1st IEEE International

Conference on Hot Information-Centric Networking (HotICN). IEEE, 2018, pp. 24–29.

[19] S. Gao, D. Zheng, R. Guo, C. Jing, and C. Hu, "An anti-quantum e-voting protocol in blockchain with audit function," IEEE Access, vol. 7, pp. 115 304–115 316, 2019.

[20] Q. Wang, C. Yu, F. Gao, H. Qi, and Q. Wen, "Self-tallying quantum anonymous voting," Physical Review A, vol. 94, no. 2, p. 022333, 2016.

[21] Y.-R. Li, D.-H. Jiang, Y.-H. Zhang, and X.-Q. Liang, "A quantum voting protocol using single-particle states," Quantum Information Processing, vol. 20, no. 3, pp. 1–17, 2021.

[22] H. Li, Y. Li, Y. Yu, B. Wang, and K. Chen, "A blockchain-based traceable self-tallying e-voting protocol in ai era," IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 1019–1032, 2021.

[23] E. Zaghloul, T. Li, and J. Ren, "d-bame: Distributed blockchain-based anonymous mobile electronic voting," IEEE Internet of Things Journal, vol. 8, no. 22, pp. 16 585–16 597, 2021.

[24] 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), 2020, pp. 176–180.

[25] P. P. Mukherjee, A. A. Boshra, M. M. Ashraf, and M. Biswas, "A hyper-ledger fabric framework as a service for improved quality e-voting system," in 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 394–397.

[26] T. Roopak and R. Sumathi, "Electronic voting based on virtual id of aadhar using blockchain technology," in 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020, pp. 71–75.

[27] S. Al-Maaitah, M. Qatawneh, and A. Quzmar, "E-voting system based on blockchain technology: A survey," in 2021 International Conference on Information Technology (ICIT), 2021, pp. 200–205.

[28] S. Vivek, R. Yashank, Y. Prashanth, N. Yashas, and M. Namratha, "E-voting systems using blockchain: An exploratory literature survey," in 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 890–895.

[29] V. Neziri, R. Dervishi, and B. Rexha, "Survey on using blockchain technologies in electronic voting systems," in 2021 25th International Conference on Circuits, Systems, Communications and Computers (CSCC), 2021, pp. 61–65.

[30] K. L. Ohammah, S. Thomas, A. Obadiah, S. Mohammed, and Y. S. Lolo, "A survey on electronic voting on blockchain," in 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON), 2022, pp. 1–4.

[31] M.-V. Vladucu, Z. Dong, J. Medina, and R. Rojas-Cessa, "E-voting meets blockchain: A survey," IEEE Access, vol. 11, pp. 23 293–23 308, 2023.

[32] J. Huang, D. He, M. S. Obaidat, P. Vijayakumar, M. Luo, and K.-K. R. Choo, "The application of the blockchain technology in voting systems: A review," ACM Comput. Surv., vol. 54, no. 3, apr 2021. [Online]. Available: https://doi.org/10.1145/3439725

[33] L. Zhang, Y. Hu, X. Tian, and Y. Yang, "Novel identity-based blind signature for electronic voting system," in 2010 Second International Workshop on Education Technology and Computer Science, vol. 2, 2010, pp. 122–125.

[34] S. Ibrahim, M. Kamat, M. Salleh, and S. Aziz, "Secure e-voting with blind signature," in 4th National Conference of Telecommunication Technology, 2003. NCTT 2003 Proceedings., 2003, pp. 193–197.

[35] Y. Mu and V. Varadharajan, "Anonymous secure e-voting over a network," in Proceedings 14th Annual Computer Security Applications Conference (Cat. No.98EX217), 1998, pp. 293–299.

[36] J.-K. Jan, Y.-Y. Chen, and Y. Lin, "The design of protocol for e-voting on the internet," in Proceedings IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No.01CH37186), 2001, pp. 180–189.

[37] A. Stone, “E-voting: should we pull the lever?” IEEE Software, vol. 20, no. 6, pp. 12–14, 2003.

[38] C. Lambrinoudakis, D. Gritzalis, and S. Katsikas, “Building a reliable e-voting system: functional requirements and legal constraints,” in Proceedings. 13th International Workshop on Database and Expert Systems Applications, 2002, pp. 435–.

[39] G. Salomonsen, “Protection against hackers on client computers for e-voting systems,” in Proceedings. 13th International Workshop on Database and Expert Systems Applications, 2002, pp. 436–.

[40] “e-Devlet Kapısı Devletin Kısayolu | www.türkiye.gov.tr — turkiye.gov.tr,” https://www.turkiye.gov.tr/, [Accessed 08-08-2023].

[41] D. Chaum, “Blind signatures for untraceable payments,” in Advances in Cryptology, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Boston, MA: Springer US, 1983, pp. 199–203.

[42] “GitHub - orbitdb/orbitdb: Peer-to-Peer Databases for the Decentralized Web — github.com,” https://github.com/orbitdb/orbitdb, [Accessed 01-08-2023].

[43] “IPFS Powers the Distributed Web — ipfs.tech,” https://ipfs.tech/, [Accessed 09-08-2023].

[44] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, 2001. [Online]. Available: http://www.cacr.math.uwaterloo.ca/hac/

[45] E. Ozturk, “ADSES: Anonymous and Decentralized Secure Election System,” Aug. 2023. [Online]. Available: https://github.com/esrefozturk/adsesv3

[46] “The Open Source Definition — opensource.org,” https://opensource.org/osd/, [Accessed 10-08-2023].

[47] “Welcome to Python.org — python.org,” https://www.python.org/, [Accessed 08-08-2023].

[48] “Welcome to PyCryptodome&x2019;s documentation &x2014; PyCryptodome 3.190b1 documentation — pycryptodome.org,” https://www.pycryptodome.org/, [Accessed 08-08-2023].

[49] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, p. 120–126, feb 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[50] T. Christie, “Home - Django REST framework — django-rest-framework.org,” https://www.django-rest-framework.org/, [Accessed 05-08-2023].

[51] “Cloud Computing Services - Amazon Web Services (AWS) — aws.amazon.com,” https://aws.amazon.com/, [Accessed 05-08-2023].

[52] “Provision Infrastructure as Code - AWS CloudFormation - AWS — aws.amazon.com,” https://aws.amazon.com/cloudformation/, [Accessed 08-08-2023].

[53] “Secure and resizable cloud compute – Amazon EC2 – Amazon Web Services — aws.amazon.com,” https://aws.amazon.com/ec2/, [Accessed 08-08-2023].

[54] “Amazon Route 53 | DNS Service | AWS — aws.amazon.com,” https://aws.amazon.com/route53/, [Accessed 08-08-2023].

[55] “AWS On-Demand Instances – Amazon Web Services (AWS) — aws.amazon.com,” https://aws.amazon.com/de/ec2/pricing/on-demand/, [Accessed 09-08-2023].

[56] “Load testing for engineering teams | Grafana k6 — k6.io,” https://k6.io/, [Accessed 09-08-2023].

[57] “What is Load Testing? How to create a Load Test in k6 — k6.io,” https://k6.io/docs/test-types/load-testing/, [Accessed 09-08-2023].

[58] “Constant VUs — k6.io,” https://k6.io/docs/using-k6/scenarios/executors/constant-vus/, [Accessed 09-08-2023].

[59] R. Canetti, U. Feige, O. Goldreich, and M. Naor, “Adaptively secure multi-party computation,” in Proceedings of the Twenty-Eighth Annual ACM

Symposium on Theory of Computing, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 639–648. [Online]. Available: https://doi.org/10.1145/237814.238015

[60] U. Fiege, A. Fiat, and A. Shamir, "Zero knowledge proofs of identity," in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, ser. STOC '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 210–217. [Online]. Available: https://doi.org/10.1145/28395.28419