DOI: 10.37943/22FULA2667

Alnur Kussainov
PhD Student, Department of Information Security Systems
alnur97@mail.ru; orcid.org/0009-0003-1469-455X
L.N. Gumilyov Eurasian National University, Kazakhstan

Zeynep Galymzhankyzy
Bachelor's Student, Department of Intelligence Systems and Cybersecurity
212174@astanait.edu.kz; orcid.org/0009-0004-4067-607X
Astana IT University, Kazakhstan

Dinara Akimova
Senior lecturer, Department of Intelligence Systems and Cybersecurity
dinara.akimova@astanait.edu.kz; orcid.org/0009-0000-6142-0234
Astana IT University, Kazakhstan

Laura Aldasheva
Candidate of Technical Sciences, Assistant Professor, Department of Intelligence Systems and Cybersecurity
laura.aldasheva@astanait.edu.kz; orcid.org/0000-0001-6815-1989
Astana IT University, Kazakhstan

Zamira Mukhtarova
Master's Degree, Department of Intelligence Systems and Cybersecurity
zamira.mukhtarova@astanait.edu.kz; orcid.org/0009-0003-9072-8475
Astana IT University, Kazakhstan

ENHANCED INFORMATION SECURITY FOR VOTING SYSTEM IN EMERGENCIES USING PAILLIER'S CRYPTOSYSTEM

Abstract: This study explores the application of Paillier's Partial Homomorphic Encryption (PHE) in the context of secure digital voting systems, particularly in emergency situations such as pandemics, natural disasters, or martial law. The proposed system is implemented using Python with the Django framework and the pycryptodome library to ensure a secure and scalable environment. A key feature of Paillier's cryptosystem is its ability to perform computations directly on encrypted data, which preserves voter confidentiality and guarantees data integrity without requiring decryption. A simulated voting scenario involving 10 voters and 3 candidates was conducted to evaluate the system. Encrypted votes were processed using homomorphic operations, allowing for the secure aggregation of votes. The results demonstrated that the system accurately computed vote totals – 35 votes for Candidate A, 50 for Candidate B, and 100 for Candidate C – without compromising security. The system proved efficient and reliable for small-scale implementations. However, the study identifies significant challenges when scaling the system to national-level elections. The cryptographic operations required by Paillier's scheme are computationally intensive and could hinder performance when processing millions of encrypted votes in real-time. Therefore, while the system shows high potential for secure e-voting, further research is required to optimize performance. The authors propose future work in two directions: optimizing the underlying cryptographic operations and integrating blockchain technologies to enhance transparency and auditability. Overall, the results

Received: 18.07.2024 Accepted: 25.06.2025 Published: 30.06.2025

suggest that Paillier's PHE provides a robust framework for emergency e-voting systems and offers a substantial improvement over traditional voting methods in terms of both security and privacy.

Keywords: Paillier's Algorithm; Homomorphic; Python; E-voting; Encryption; Voter Privacy; Data Security.

Introduction

In today's digital era, enhancing data security on platforms managing sensitive tasks like voting systems is crucial due to evolving cyber threats. Traditional encryption methods, which require data decryption for processing, are becoming less effective, particularly for systems handling sensitive information where security breaches can have significant consequences.

Homomorphic encryption offers a promising solution. This advanced cryptographic technique allows for computations on encrypted data without needing decryption, ensuring continuous data protection and significantly reducing the risk of breaches. This method not only preserves data integrity and privacy but also facilitates necessary computational tasks, making it ideal for real-world applications like electronic voting systems. The goal of this research project is to integrate Homomorphic Encryption, specifically Paillier's Partial Homomorphic Encryption (PHE), into Python-based web applications for e-voting. This integration aims to use Python's pycryptodome and PHE libraries to ensure the e-voting system is secure, scalable, and user-friendly. By using Paillier's PHE, the project enhances the security of digital electoral processes, maintaining vote confidentiality and integrity throughout their lifecycle. This research will examine the theoretical, technical, operational, and security aspects of implementing Paillier's PHE in e-voting systems, providing a detailed comparative analysis of its computational efficiency and security against traditional methods.

Contribution [1] focuses on a two-server architecture for a generic e-voting system. This method enhances system reliability by distributing responsibilities between two servers, reducing the chances of a single point of failure. However, while architecture improves reliability, the study does not explore advanced cryptographic techniques like homomorphic encryption for enhancing data privacy and confidentiality during the voting process. Moreover, the system lacks a focus on large-scale election scenarios, which your article addresses by implementing Paillier's cryptosystem for secure and scalable e-voting in emergencies.

Authors of paper [2] propose a secure voting system that incorporates visual cryptography and secure multiparty computation. Their approach emphasizes secure vote storage and computation but does not provide a comprehensive solution for maintaining voter anonymity or performing computations on encrypted votes. Your research fills this gap by leveraging the Paillier cryptosystem's Partial Homomorphic Encryption (PHE) to ensure both vote confidentiality and secure tallying of votes without decryption, enhancing privacy and security.

Article [3] focused on designing an e-voting system with biometric authentication and OTP-based verification. While this method ensures secure user verification, it focuses on identity validation rather than the integrity of the voting process itself. Their use of Mix Nets and Java-based technologies does not address the scalability or security challenges of vote counting in large elections. Your article, however, tackles these issues by introducing a cryptographically secure method for performing arithmetic operations on encrypted data, ensuring secure vote tallying even in large-scale elections.

Paper [4] offered an online voting system with enhanced security evaluations using SHA-512 hashing post-encryption. While this solution improves data integrity, it does not address how to perform secure computations on encrypted data. In contrast, your research builds upon this by enabling vote tallying directly on encrypted data using homomorphic encryption, thereby offering a more robust security solution that extends beyond data storage and transmission to cover the entire vote-counting process.

Research in the review article [5] devoted to blockchain's potential in e-voting, focusing on the decentralized and secure nature of blockchain systems. While blockchain can enhance transparency and prevent manipulation, their work does not address the computational burden of large-scale e-voting or the integration of cryptographic methods for vote tallying. Your article, on the other hand, addresses these concerns by combining homomorphic encryption with the Paillier cryptosystem, enabling secure computations without revealing vote contents, which is not explored in their study.

The authors of paper [6] further discuss a blockchain-based e-voting model using Ethereum and homomorphic encryption. While their model introduces the potential of homomorphic encryption in e-voting, it does not focus on Partial Homomorphic Encryption (PHE), which is particularly suited for the arithmetic operations needed in vote tallying. Article focuses specifically on PHE, providing a streamlined solution that is less computationally intensive than fully homomorphic encryption models while still offering high security and efficiency for real-world applications.

Table 1 provides a comparative overview of six studies focused on secure electronic voting solutions using diverse cryptographic and architectural methods. Below is a critical evaluation of their contributions and limitations:

Table 1. Critical Analysis of Existing E-Voting Systems

Reference	Proposed Method	Strengths	Weaknesses	Research Gap
Qadah [1]	Two-Server E-Voting System	Fault tolerance via server distribution	Lacks cryptographic safeguards and vote confidentiality	No encryption, no support for secure vote tallying
Naidu et al. [2]	Visual Cryptography + Secure Multiparty Computation	Distributed trust; visual verification	High computational cost; low scalability	No homomorphic encryption; unsuitable for large-scale elections
Patil et al. [3]	Biometric and OTP-Based Voting	Strong user authentication; reduced impersonation risk	Biometric data privacy concerns; no post-vote confidentiality	Focused only on authentication; no secure computation of encrypted votes
Sedky & Hamed [4]	SHA-512 Hash-Based Voting	Ensures data integrity through strong hashing	Hashing doesn't encrypt votes or allow vote operations	No encryption or support for homomorphic processing
Taş & Tanrıöver [5]	Blockchain-Based Review	Highlights blockchain transparency and auditability	Theoretical; lacks practical implementation or security model	No integration of encryption; lacks secure vote computation
Taş & Tanrıöver [6]	Ethereum-Based Blockchain + HE	Combines blockchain with homomorphic encryption	Ethereum incurs cost and latency; lacks performance testing	No scalability assessment; performance of HE in real elections not studied

Compared to the works analyzed, the proposed system based on Paillier's PHE uniquely balances:

- Encrypted vote operations without decryption,
- Scalability for emergency use in low-resource environments,
- Lightweight deployment using Django and Python.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 22, JUNE 2025

It fills a critical gap by offering secure tallying of encrypted votes and a pragmatic design for use during emergencies (e.g., pandemics, natural disasters), where conventional systems may fail due to infrastructure limitations or lack of trust.

Another unexplored area in the literature is the challenge of scaling e-voting systems to handle large-scale elections. Many existing studies, such as those by Naidu et al. and Patil et al., fail to consider the significant computational demands of processing millions of votes. This research addresses these concerns by implementing the Paillier cryptosystem in Python, demonstrating its ability to perform efficient, secure computations in large-scale voting scenarios, while maintaining voter confidentiality.

Furthermore, none of the reviewed works have examined the application of secure e-voting systems during emergencies, such as pandemics or natural disasters. This study uniquely addresses this gap by exploring how advanced cryptographic techniques, specifically PHE, can be employed to maintain the integrity and security of digital voting systems in such critical situations. In conclusion, this research advances the field by tackling unexplored tasks related to the use of PHE for secure vote tallying, enhancing the scalability of e-voting systems, and providing solutions for their application during emergencies.

Digitalization of voting system in emergencies using Paillier's cryptosystem

Cryptography is a method for securing data through message encoding, ensuring that only intended recipients can access and interpret the data. However, it not only protects data but also verifies the legitimacy of a user. Its primary uses include transmitting data over insecure networks, like the internet, and masking data to prevent unauthorized access to sensitive information. In cryptography, "plaintext" refers to the original, clear information that is transformed into "ciphertext" through encryption, using encryption algorithms and a specific encryption key [7]. The recipient uses a "decryption algorithm" and a "decryption key" to revert to the original information. Encryption methods are categorized into three types: symmetric-key, asymmetric-key, and modern cryptographic techniques.

Symmetric cryptography, or private-key encryption, uses the same key for both encryption and decryption, employing algorithms like Advanced Encryption Standard (AES), Triple Data Encryption Standard (3DES), and Blowfish.

Asymmetric cryptography, or public-key cryptography, uses a key pair for encryption and decryption, offering better security than private-key systems. It is suitable for internet communications but requires more energy and time. Examples include Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), and homomorphic encryption schemes like Paillier and Fully Homomorphic Encryption (FHE) [8].

Modern cryptography includes complex and mathematically sophisticated techniques ensuring secure communication against adversaries. It incorporates symmetric algorithms like AES and asymmetric systems such as RSA, along with advanced constructs like ECC and homomorphic encryption, facilitating secure operations on encrypted data [9].

In the following Table 2, factors such as time spent, key size, and how they are used are taken into account when evaluating performance. We have compared homomorphic encryption with some commonly used security systems.

DOI: 10.37943/22FULA2667
© Alnur Kussainov, Zeynep Galymzhankyzy, Dinara Akimova, Laura Aldasheva, Zamira Mukhtarova

Table 2. Encryption Methods Comparison Table

Encryption Method	Time Consumption	Key Size (bits)	Block size (bits)	Structure	Applications
Homomorphic Encryption	Fast	1024-4096	Variable	Public key Algorithm	Cloud Computing
AES	Fast	128, 192, 256	128	Substitution	Network security
RSA	Slowest	1024-4096	Variable	Public key Algorithm	Banking transactions
DES	Slow	64	64	Fetial	Legacy systems
ECC	Fast	160-521	Variable	Public key Algorithm	Smart cards
Diffie-Hellman	Medium	1024-4096	Variable	Public key Algorithm	VPN Tunnels

From the performance evaluation above, you can see the similarities and differences between different encryption algorithms. Obviously, the best encryption methods would be homomorphic and AEC. However, due to the variable key size and predisposition for cloud computing, homomorphic encryption will be slightly better than its counterparts due to greater reliability [10].

Homomorphic encryption secures data while enabling calculations on encrypted data without revealing the original information. This is particularly useful for operations like adding numbers on a cloud server without accessing the actual numbers. In a homomorphic encryption scheme, we use as an encryption function $\mathcal{E}$ mapping plaintext space $\mathcal{P}$ to ciphertext space $\mathcal{C}$, and decryption function $\mathcal{D}$ mapping back to $\mathcal{P}$. For any two plaintexts p_1, p_2 and a binary operation $\oplus$ on $\mathcal{P}$, the scheme is homomorphic if there is a corresponding operation $\otimes$ on $\mathcal{C}$ that satisfies:

$$D\big(E(p_1) \otimes E(p_2)\big) = p_1 \oplus p_2 \tag{1}$$

Above mentioned property means that operations performed on encrypted data yield the same results when decrypted as if they were performed on the original data.

Homomorphic encryption is increasingly important for secure data processing, notably in e-voting systems. While traditional encryption methods secure data for sharing and storage, they do not support data manipulation without decryption. This limitation presents security risks, making the capabilities of homomorphic encryption a focal point for telecommunications and cybersecurity professionals. Homomorphic Encryption Techniques are classified into three types: Partial Homomorphic Encryption (PHE), which allows only one type of operation on ciphertexts; Somewhat Homomorphic Encryption (SHE), which permits a limited number of operations; and Fully Homomorphic Encryption (FHE), which supports unlimited operations on ciphertexts. that previous attempts to implement E-Voting systems often failed due to high maintenance costs [11].

Comparison of homomorphic methods presented in the Table 3.

Table 3. Homomorphic Methods comparison

Methods	Type of Operation	Number of operations
Partial	Addition or Multiplication	Unlimited
Somewhat	Addition and Multiplication	Limited number
Fully	Addition and Multiplication	Unlimited

Choosing PHE is beneficial as it, despite its limitation to a single operation type, does not restrict the number of operations like SHE and is less complex than FHE [12]. In E-voting we need only mathematical addition for calculating results. The simplicity of PHE in system maintenance offers a significant advantage, especially considering that previous attempts to implement E-Voting systems often failed due to high maintenance costs.

Discussion

Paillier's Homomorphic Encryption in Voting

Consider an election scenario with 10 voters and 3 candidates (A, B, C). Votes are encrypted and tallied using Paillier's Partial Homomorphic Encryption as shown in Table 4.

Table 4. Using Paillier's PHE in voting

Voter Number	A	B	C	Encrypted Vote
V1	✓			E(1)
V2		✓		E(10)
V3	✓			E(1)
V4			✓	E(100)
V5		✓		E(10)
V6			✓	E(100)
V7	✓			E(1)
V8		✓		E(10)
V9			✓	E(100)
V10	✓			E(1)

Each vote is encrypted using Paillier's algorithm to yield an encrypted value, which is represented symbolically here as *E*(*vote value*) [13]. The homomorphic property of the Paillier encryption allows us to compute the encrypted sum of all votes without decrypting individual votes:

$$E(\text{Total Votes}) = \prod_{i=1}^{10} E(\text{Vote}_i) \quad mod \;\; n^2 \tag{2}$$

Upon decryption, we obtain the sum of all vote values:

$$\text{Decrypted Total} = D\big(E(\text{Total Votes})\big) = \sum_{i=1}^{10} \text{Vote}_i \quad\; mod \;\; n \tag{3}$$

The decrypted total can then be used to deduce the number of votes for each candidate by examining the place values, as Paillier's algorithm allows for addition of encrypted votes. The tally for each candidate is computed by taking the digit from the corresponding place value.

The primary objective of this research is to enhance the security and integrity of e-voting systems during emergency situations, such as pandemics or natural disasters, through the implementation of Paillier's Partial Homomorphic Encryption (PHE). The study aims to develop a secure, scalable, and efficient e-voting system that ensures voter anonymity, data confidentiality, and vote integrity, while addressing the computational challenges of large-scale elections [14].

Algorithm behind Implementation

The electronic voting system in the Figure 1 is characterized by a sequence of secure steps: voters cast votes that are encrypted using the Paillier's algorithm, resulting in ciphertexts stored in a database. The tallying software will process each piece of ciphertext.

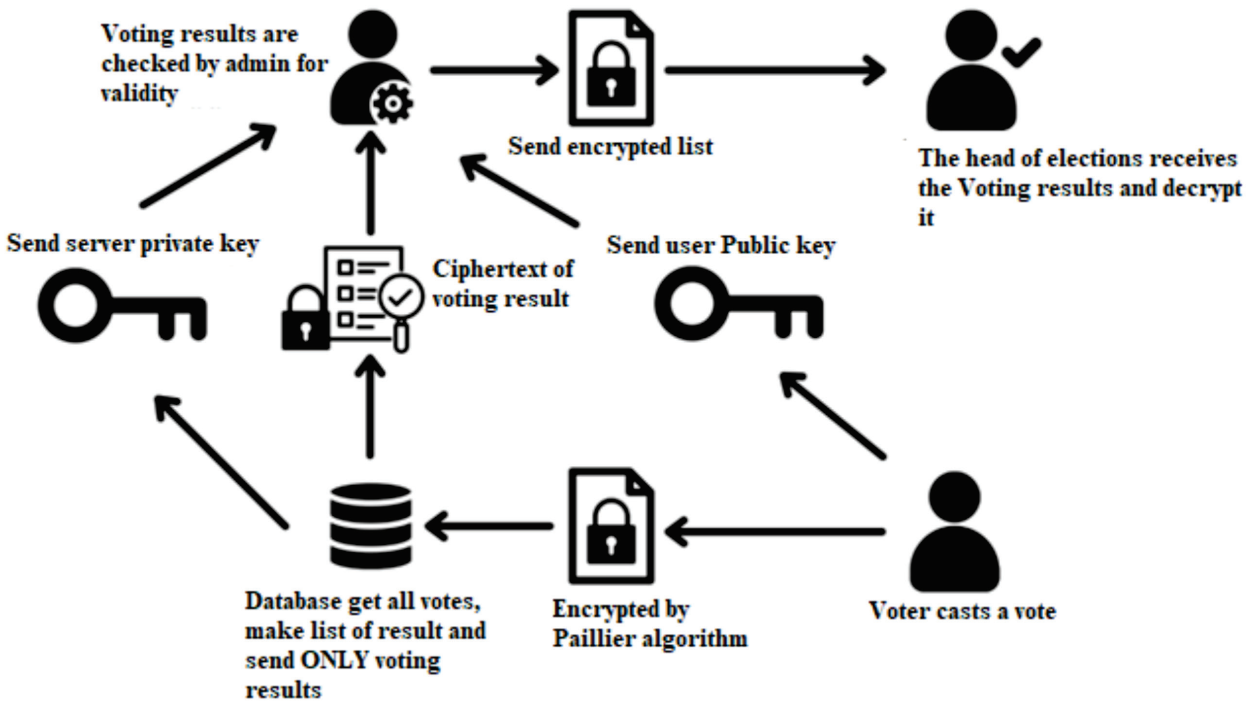

Voting results are checked by admin for validity
Send encrypted list
The head of elections receives the Voting results and decrypt it
Send server private key
Ciphertext of voting result
Send user Public key
Database get all votes, make list of result and send ONLY voting results
Encrypted by Paillier algorithm
Voter casts a vote

Figure 1. Secured electronic voting system

The presented in Figure 2 Entity-Relationship Diagram (ERD) illustrates the logical structure of a database designed for a secure and transparent election management system. Architecture employs a series of interconnected tables with one-to-many relationships, ensuring data consistency and integrity through the use of foreign keys [15]. This design facilitates effective management of user data, election details, voting records, and system interactions.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 22, JUNE 2025

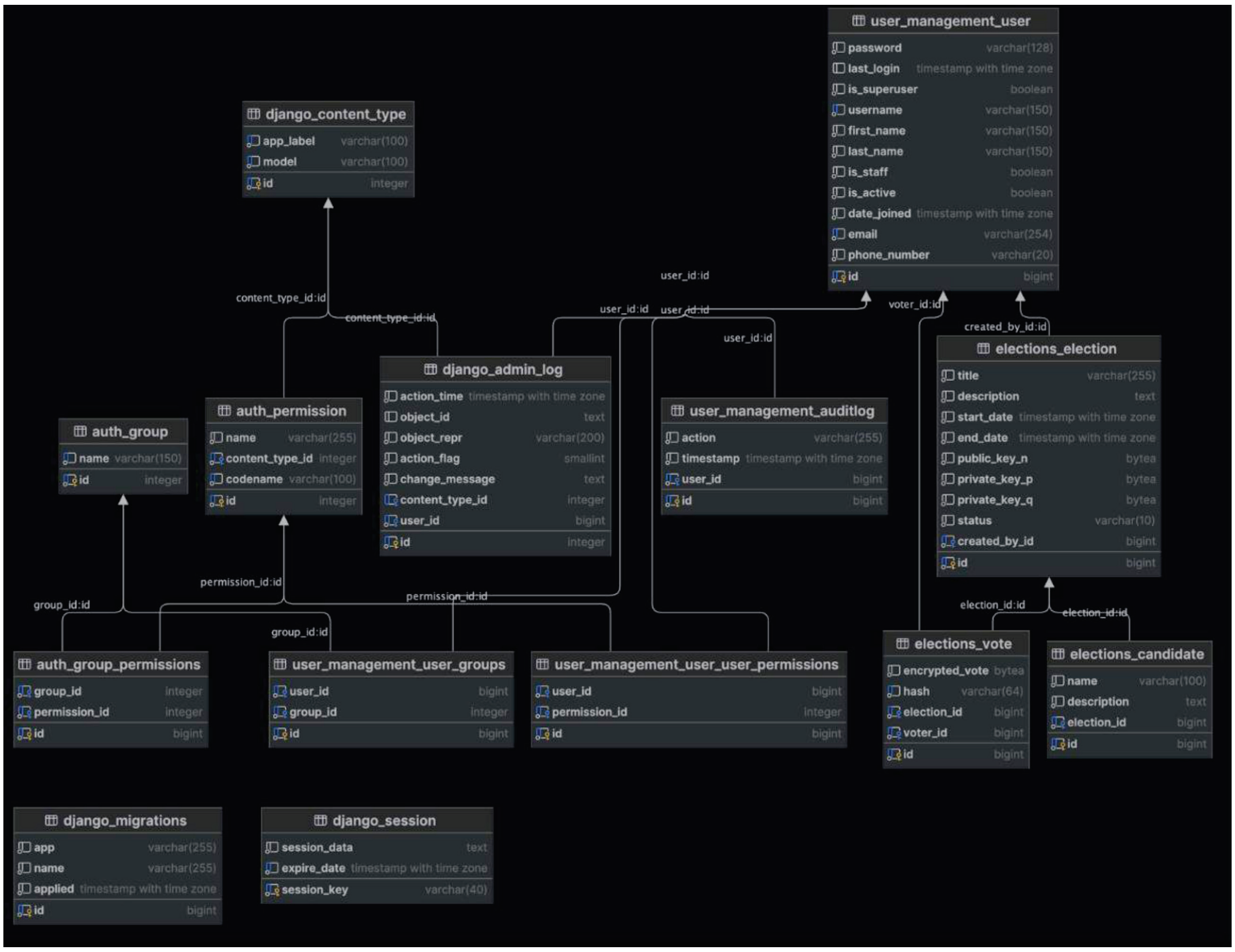


Figure 2. Entity-Relationship Diagram (ERD)

The **user management system** is centered around the *user_management_user* table, which stores essential user credentials, roles, and contact information. The database structure allows for differentiation between administrative and regular users using fields such as *is_superuser* and *is_staff*. This enables granular access control, ensuring that only authorized individuals can perform sensitive operations like creating elections or adding candidates.

The **elections module** is comprised of tables such as *elections_election*, *elections_vote*, and *elections_candidate*. The *elections_election* table stores metadata for each election, including its title, description, start and end dates, and encryption keys. The *elections_vote* table handles encrypted voting data, maintaining links to the corresponding voter and election through foreign keys. Additionally, the *elections_candidate* table connects each candidate to a specific election, ensuring clear relationships between entities.

To ensure **system accountability and traceability**, the database incorporates auditing mechanisms through the *django_admin_log* and *user_management_auditlo*g tables. These tables record user actions such as login attempts, data modifications, and administrative operations, enabling a comprehensive audit trail. The use of *user_id* foreign keys allows tracking of individual user activity over time.

The a**uthorization and permissions framework** are supported by tables like *auth_group* and *auth_permission*, which define user roles and associated privileges. Intermediary tables (*auth_group_permissions*, *user_management_user_groups*, and *user_management_user_user_permissions*) facilitate dynamic assignment of permissions, ensuring a scalable and secure system.

DOI: 10.37943/22FULA2667

Additional components include the *django_session* and *django_migrations* tables, which manage active user sessions and track database schema migrations, respectively. These elements enhance the operational efficiency and adaptability of the system.

Overall, the database structure reflects a robust design that prioritizes **data security, transparency, and scalability**. By integrating encryption mechanisms, traceability features, and role-based access controls, the system ensures the reliable management of sensitive election-related information. This approach provides a scalable framework suitable for handling complex election processes while maintaining the integrity of stored data.

As previously mentioned, the Paillier's algorithm, through its homomorphic properties, enables the calculation of $E(m1+m2)$ by multiplying $m1$ and $m2$ directly in their encrypted forms without the need for prior decryption [16]. Election officials validate these encrypted results before the server's private key is securely sent for result decryption. Simultaneously, encrypted results and the corresponding public key are dispatched to the election head, who decrypts the data to reveal the final vote tally.

The Paillier's encryption scheme necessitates the utilization of substantial prime numerals during the procedure of generating keys. Presented below (as Algorithm 1) is the pseudocode that delineates the technique for the derivation of these prime numbers. Nonetheless, for the purpose of evaluating the Homomorphic characteristics within the simulation in this chapter, prime numbers of a 16-bit size, specifically 65521 and 65519, are employed.

Algorithm 1 Generate Paillier Cryptosystem Primes

```
function GenerateLargePrime(bitsize)
    lowerBound ← $2^{bitsize-1}$
    upperBound ← $2^{bitsize} - 1$
    return RandPrime(lowerBound, upperBound)
end function
function GeneratePaillierPrimes(bitsize = 512)
    p ← GenerateLargePrime(bitsize)
    q ← GenerateLargePrime(bitsize)
    while $q = p$ do
        q ← GenerateLargePrime(bitsize)
    end while
    return p, q
end function
```

The results presented in Table 5 underscore the significance of the random value r in ensuring the uniqueness of the ciphertext within the Paillier's encryption algorithm. Specifically, when varying r values are employed, even identical plaintexts yield unique ciphertexts, highlighting the role of r as a critical factor in the encryption process.

Table 5. Ciphertext Uniqueness Test Result

No.	Plaintext (m)	Random r	Ciphertext (C)
1	8	1046334137	10742187944272525259
2	8	3369841305	12646713718566171470
3	57	3369841305	16038966592338371654
4	22	1046334137	13966243507110869260
5	22	1412997532	11730110218858419469
6	23	1798563773	15709037477692289954
7	22	1412997532	11730110218858419469

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 22, JUNE 2025

As shown in Table 5 in contrast, employing the same r value for encrypting identical plaintexts invariably produces duplicate ciphertexts, thereby confirming the consistency of the algorithm. The Ciphertext Uniqueness Test serves as a robust mechanism to verify the Paillier algorithm's capacity for secure data encryption while simultaneously showcasing its homomorphic capabilities, which enable computations on ciphertexts that reflect corresponding operations on the plaintexts.

Decryption Function Tests

Algorithm 3 designated as the Decryption Test, verifies the fidelity of the Paillier's cryptosystem in accurately reconstructing plaintexts from their encrypted counterparts, utilizing the prime numbers p=65521 and q=65519.

Algorithm 3 Public and Private Key Generation and Encryption Algorithm

```
Require: p, q    ▷ Prime numbers
Ensure: Public key (g, n), Private key (λ, μ), and ciphertexts 1:
  function GENERATE_KEYS(p, q)
    $n \leftarrow p \cdot q$.
    $g \leftarrow n + 1$
    $\lambda \leftarrow (p - 1) \cdot (q - 1)$
    $\mu \leftarrow invert(\lambda, n)$    ▷ Modular multiplicative inverse return (g, n), (λ, μ)
  end function

  function ENCRYPT(public_key, m, r)
    g, n ← public_key
    $n_{sq} \leftarrow n \cdot n$
    cipher ← $(g^m \cdot r^n)$ mod $n_{sq}$ return cipher
  end function

    p ← 65521    ▷ Prime number
    q ← 65519    ▷ Prime number
    public_key, private_key ← GENERATE_KEYS(p, q)
    m ← 8    ▷ Plaintext message
    r_values ← [1046334137, 3369841305]    ▷ Random values
    ciphertexts ← []
    for r in r_values do
        ciphertexts.append(ENCRYPT(public_key, m, r))
    end for
    for $i = 0$ to length(ciphertexts) − 1 do
        print "Ciphertext i + 1: ciphertexts[i]"
    end for
```

This process substantiates the system's capability to decrypt messages precisely, thereby affirming its reliability and efficiency in maintaining the confidentiality and integrity of encrypted communications.

Testing the Additive Homomorphic Property of Paillier's Method

This homomorphic test results in Table 5 verifies the additive homomorphic property of the Paillier cryptosystem which is illustrated in Algorithm 5. It demonstrates that performing arithmetic operations on ciphertexts yields the same results as if the operations were performed on plaintexts. In the test, two plaintext numbers are encrypted, their ciphertexts are multiplied, and the result is decrypted. If the decrypted result matches the sum of the original plaintexts, the test confirms that the cryptosystem correctly supports homomorphic addition. This homomorphic test results in Table 5 verifies the additive homomorphic property of the Paillier's cryptosystem which is illustrated Algorithm 4.

The graph in the Figure 2 represents the Message Expansion Factor (MEF) for the Paillier cryptosystem across different key sizes, measured in bits. The key sizes tested are 256, 512, 1024, and 2048 bits. The MEF is calculated as the ratio of the ciphertext size to the plaintext size after encryption.

Table 6. Decryption Test Result

No.	Ciphertext (C)	Decrypted plaintext
1	10742187944272525259	8
2	12646713718566171470	8
3	16038966592338371654	57
4	13966243507110869260	22
5	11730110218858419469	22
6	15709037477692289954	23
7	11730110218858419469	22

From Figure 3, it is evident that the MEF increases linearly with the size of the prime numbers used in the key generation process. The plotted values on the graph show the MEF for each key size:

- At 256 bits, the MEF is approximately 15.30;
- At 512 bits, it increases to about 30.80;
- For 1024 bits, it rises further to around 61.60;
- Finally, at 2048 bits, the MEF reaches approximately 123.30.

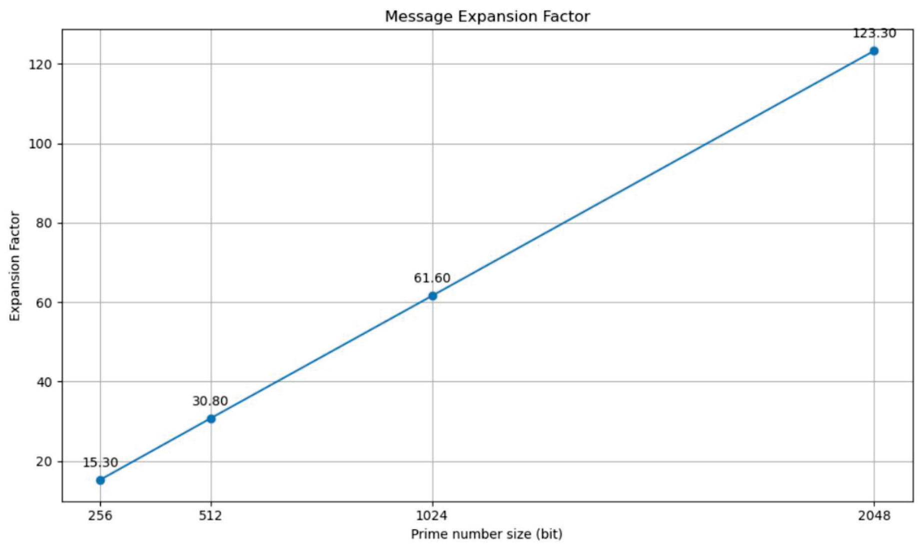


Figure 3. Message Expansion Factor for different bit sizes

Algorithm 4 demonstrates that performing arithmetic operations on ciphertexts yields the same results as if the operations were performed on plaintexts. In the test, two plaintext numbers are encrypted, their ciphertexts are multiplied, and the result is decrypted. If the decrypted result matches the sum of the original plaintexts, the test confirms that the cryptosystem correctly supports homomorphic addition.

Algorithm 4 Paillier Decryption Algorithm

```
Require: p, q.                          ▷ Prime numbers
Ensure: Decrypted plaintexts from ciphertexts
    function L(x, n)
      return (x − 1)//n
    end function
    function DECRYPT(ciphertext, $\lambda_n$, $\mu$, n, $n_{sq}$)
    $u \leftarrow$ POWMOD(ciphertext, $\lambda_n$, $n_{sq}$)
    $l\_of\_u \leftarrow L(u, n)$
    plaintext $\leftarrow (l\_of\_u \cdot \mu)\%n$
    return plaintext
    end function
    $p \leftarrow 65521$                     ▷ A prime number
    $q \leftarrow 65519$                     ▷ A prime number
    $n \leftarrow p \cdot q$
    $n_{sq} \leftarrow n \cdot n$
    $\lambda_n \leftarrow (p-1) \cdot (q-1)//GCD(p-1, q-1)$
    $\mu \leftarrow$ INVERT($\lambda_n$, n)
    ciphertexts ← [107421...25259, 126467...6171470...]
    for $i \leftarrow 1$ to LENGTH(ciphertexts) do
    ciphertext ← ciphertexts[$i-1$]
    plaintext ← DECRYPT(ciphertext, $\lambda_n$, $\mu$, n, $n_{sq}$)
    print "Ciphertext i: ciphertext"
    print "Decrypted plaintext: plaintext"
    end for
```

This trend suggests that as the bit length of the prime numbers used to generate the Paillier keypair increases, the ciphertext becomes proportionally larger. The annotations on the graph serve as visual aids to directly relate each key size to its corresponding MEF, illustrating the computational and storage implications of choosing larger key sizes for encryption [17]. Validation process results presented in Table 7.

Table 7. Homomorphic Property Checking

No.	Plaintexts (m1 + m2)	Homomorphic Decryption	Expected Sum
1	123 + 456	579	579
2	789 + 321	1110	1110
3	81736 + 91873	173609	173609
4	19837 + 335621	355458	355458
5	73642 + 61374	135016	135016

As shown in the Table 7 Homomorphic decryption result and the expected value is the same.

Materials and Methods

The primary objective of this research is to enhance the security and integrity of e-voting systems during emergency situations, such as pandemics or natural disasters, through the implementation of Paillier's Partial Homomorphic Encryption (PHE). The study seeks to develop a secure, scalable, and efficient e-voting system that ensures voter anonymity, data confidentiality, and vote integrity, while addressing the computational challenges associated with large-scale elections.

The research involved several key tasks. First, the integration of Paillier's PHE into the e-voting framework was achieved using Python's libraries (pycryptodome and phe). This allowed the system to tally encrypted votes without decryption, preserving voter privacy and data integrity. In a simulated election with 10 voters and 3 candidates, the system securely and accurately tallied encrypted votes, producing final results of 35 votes for Candidate A, 50 for Candidate B, and 100 for Candidate C. This demonstrated the system's effectiveness in secure vote tallying [18].

Additionally, the study addressed the computational challenges of large-scale elections. Testing showed that Paillier's PHE could efficiently handle encrypted data for multiple voters, although further optimization would be required to handle large datasets more effectively. The system's architecture, developed using Django, supports practical implementation with real-time processing.

In emergency scenarios where traditional voting may be disrupted, the system offers a robust solution. It ensures the security and integrity of the voting process, preventing unauthorized access and manipulation during crises.

The research object is an e-voting system designed for secure vote tallying during emergencies. The system leverages Paillier's PHE to perform operations on encrypted data without decryption, ensuring privacy and data integrity throughout the voting process. The hypothesis is that Paillier's PHE can provide a secure, scalable solution for digital voting in emergency situations. Key assumptions include secure access to voting devices and secure management of cryptographic keys. The study confirms that Paillier's cryptosystem, when implemented properly, can offer a secure and efficient solution for e-voting. Future work is suggested to explore blockchain integration and further enhance cryptographic techniques for scalability and security.

Testing The Voting Simulation

The main part of the Algorithm 6 simulates casting and tallying encrypted votes, then decrypts the tallies for each candidate. This structured approach helps in understanding the step-by-step process of the voting simulation using Paillier encryption. The system employs homomorphic encryption algorithms to conduct the tallying process, which involves operations directly on the ciphertext, specifically multiplication in this context. The outcomes of this homomorphic tallying are illustrated in Table 8 provided below.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 22, JUNE 2025

Table 8. Votes and Their Corresponding Ciphertexts

Voter: Option	Ciphertext
Voter 01: Vote for option 5	7447475312921932632...
Voter 02: Vote for option 3	718080081036955148...
Voter 03: Vote for option 1	9870941216688980976...
Voter 04: Vote for option 1	4845835457257894926...
Voter 05: Vote for option 2	5886547411102804780...
Voter ... : Vote for option ...	......
Voter 47: Vote for option 3	12237525600275294414...
Voter 48: Vote for option 2	8100284350115650664...
Voter 49: Vote for option 2	2745942120437772140...
Voter 50: Vote for option 3	9636004279805792041...

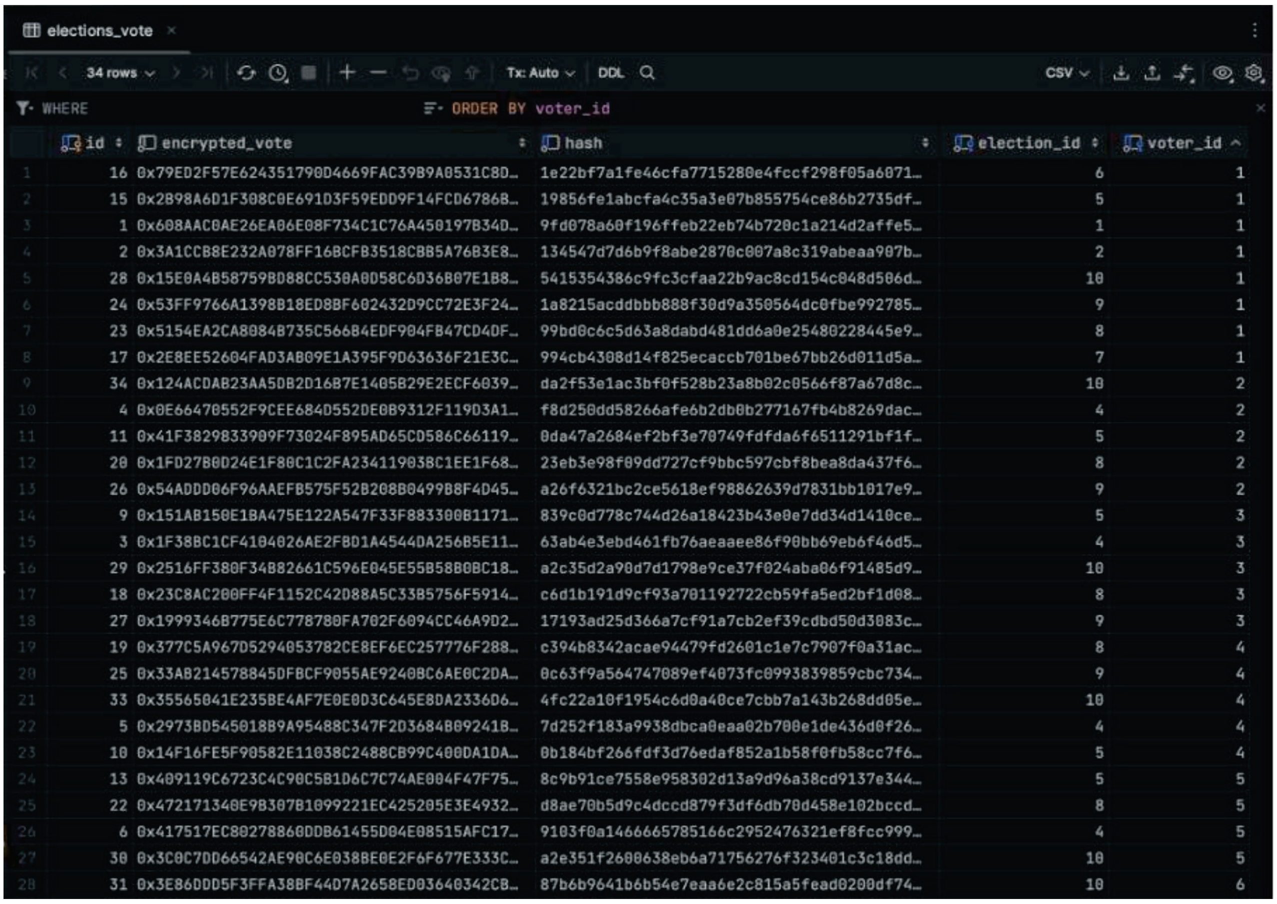
Algorithm 6 Paillier Cryptosystem Voting Simulation

```
Require: Two prime numbers $p$ and $q$
Ensure: Correct vote tallies for each candidate
  $votes \leftarrow$ Array of random votes
  $encryptedVotes \leftarrow$ Empty array
  for each $vote$ in $votes$ do
      $encryptedVote \leftarrow$ ENCRYPT($vote, n, g$)
      Add $encryptedVote$ to $encryptedVotes$
  end for

  $tallies \leftarrow$ Initialize array of zero-encrypted values for each candidate
  for each $encryptedVote$ in $encryptedVotes$ do
      $tallies[vote] \leftarrow (tallies[vote] \cdot encryptedVote) \bmod n_{sq}$
  end for
  for each $tally$ in $tallies$ do
      $decryptedTally \leftarrow$ DECRYPT($tally, \lambda, \mu, n$)
      print "Candidate x: $decryptedTally$ votes"
  end for
```

Results

The presented architecture in Figure 4 illustrates a secure and efficient approach to managing encrypted voting data within an election management system.


elections_vote

34 rows

WHERE

ORDER BY voter_id

id	encrypted_vote	hash	election_id	voter_id
16	0x79ED2F57E624351790D4669FAC39B9A0531C8D…	1e22bf7a1fe46cfa7715280e4fccf298f05a6071…	6	1
15	0x2B98A6D1F308C0E691D3F59EDD9F14FCD6786B…	19856fe1abcfa4c35a3e07b855754ce86b2735df…	5	1
1	0x608AAC0AE26EA06E08F734C1C76A450197B34D…	9fd078a60f196ffeb22eb74b720c1a214d2affe5…	1	1
2	0x3A1CCB8E232A078FF16BCFB3518CBB5A76B3E8…	134547d7d6b9f8abe2870c007a8c319abeaa907b…	2	1
28	0x15E0A4B58759BD88CC530A0D58C6D36B07E1B8…	5415354386c9fc3cfaa22b9ac8cd154c048d506d…	10	1
24	0x53FF9766A1398B18ED8BF602432D9CC72E3F24…	1a8215acddbbb888f30d9a350564dc0fbe992785…	9	1
23	0x5154EA2CA8084B735C566B4EDF904FB47CD4DF…	99bd0c6c5d63a8dabd481dd6a0e25480228445e9…	8	1
17	0x2E8EE52604FAD3AB09E1A395F9D63636F21E3C…	994cb4308d14f825ecaccb701be67bb26d011d5a…	7	1
34	0x124ACDAB23AA5DB2D16B7E1405B29E2ECF6039…	da2f53e1ac3bf0f528b23a8b02c0566f87a67d8c…	10	2
4	0x0E66470552F9CEE684D552DE0B9312F119D3A1…	f8d250dd58266afe6b2db0b277167fb4b8269dac…	4	2
11	0x41F3829833909F73024F895AD65CD586C66119…	0da47a2684ef2bf3e70749fdfda6f6511291bf1f…	5	2
20	0x1FD27B0D24E1F80C1C2FA23411903BC1EE1F68…	23eb3e98f09dd727cf9bbc597cbf8bea8da437f6…	8	2
26	0x54ADDD06F96AAEFB575F52B208B0499B8F4D45…	a26f6321bc2ce5618ef98862639d7831bb1017e9…	9	2
9	0x151AB150E1BA475E122A547F33F883300B1171…	839c0d778c744d26a18423b43e0e7dd34d1410ce…	5	3
3	0x1F38BC1CF4104026AE2FBD1A4544DA256B5E11…	63ab4e3ebd461fb76aeaaee86f90bb69eb6f46d5…	4	3
29	0x2516FF380F34B82661C596E045E55B58B0BC18…	a2c35d2a90d7d1798e9ce37f024aba06f91485d9…	10	3
18	0x23C8AC200FF4F1152C42D88A5C33B5756F5914…	c6d1b191d9cf93a701192722cb59fa5ed2bf1d08…	8	3
27	0x1999346B775E6C778780FA702F6094CC46A9D2…	17193ad25d366a7cf91a7cb2ef39cdbd50d3083c…	9	3
19	0x377C5A967D5294053782CE8EF6EC257776F288…	c394b8342acae94479fd2601c1e7c7907f0a31ac…	8	4
25	0x33AB214578845DFBCF9055AE9240BC6AE0C2DA…	0c63f9a564747089ef4073fc0993839859cbc734…	9	4
33	0x35565041E235BE4AF7E0E0D3C645E8DA2336D6…	4fc22a10f1954c6d0a40ce7cbb7a143b268dd05e…	10	4
5	0x2973BD545018B9A95488C347F2D3684B09241B…	7d252f183a9938dbca0eaa02b700e1de436d0f26…	4	4
10	0x14F16FE5F90582E11038C2488CB99C400DA1DA…	0b184bf266fdf3d76edaf852a1b58f0fb58cc7f6…	5	4
13	0x409119C6723C4C90C5B1D6C7C74AE004F47F75…	8c9b91ce7558e958302d13a9d96a38cd9137e344…	5	5
22	0x472171340E9B307B1099221EC425205E3E4932…	d8ae70b5d9c4dccd879f3df6db70d458e102bccd…	8	5
6	0x417517EC80278860DDB61455D04E08515AFC17…	9103f0a1466665785166c2952476321ef8fcc999…	4	5
30	0x3C0C7DD66542AE90C6E038BE0E2F6F677E333C…	a2e351f2600638eb6a71756276f323401c3c18dd…	10	5
31	0x3E86DDD5F3FFA38BF44D7A2658ED03640342CB…	87b6b9641b6b54e7eaa6e2c815a5fead0200df74…	10	6

Figure 4. Stored data in database

The system employs robust encryption mechanisms to protect sensitive data and ensure the integrity of the voting process. Encrypted votes are stored in the elections_vote table, where each record includes a unique identifier (id), a cryptographically hashed value (hash) to verify data integrity, and foreign keys (election_id, voter_id) that link votes to their corresponding elections and voters. Additionally, the elections_election table maintains 2048-bit prime number keys (private_key_p, private_key_q) associated with each election to enable secure encryption and decryption operations. The inclusion of a status field further enhances the system by distinguishing between active ("open") and completed ("closed") elections, allowing for precise management of election lifecycles. This design emphasizes data confidentiality, integrity, and traceability, showcasing a scalable and reliable framework for secure electronic voting systems [19].

From the data presented in the Table 9, it is evident that the results obtained through decrypting the homomorphic tally align perfectly with those derived from manual computations.

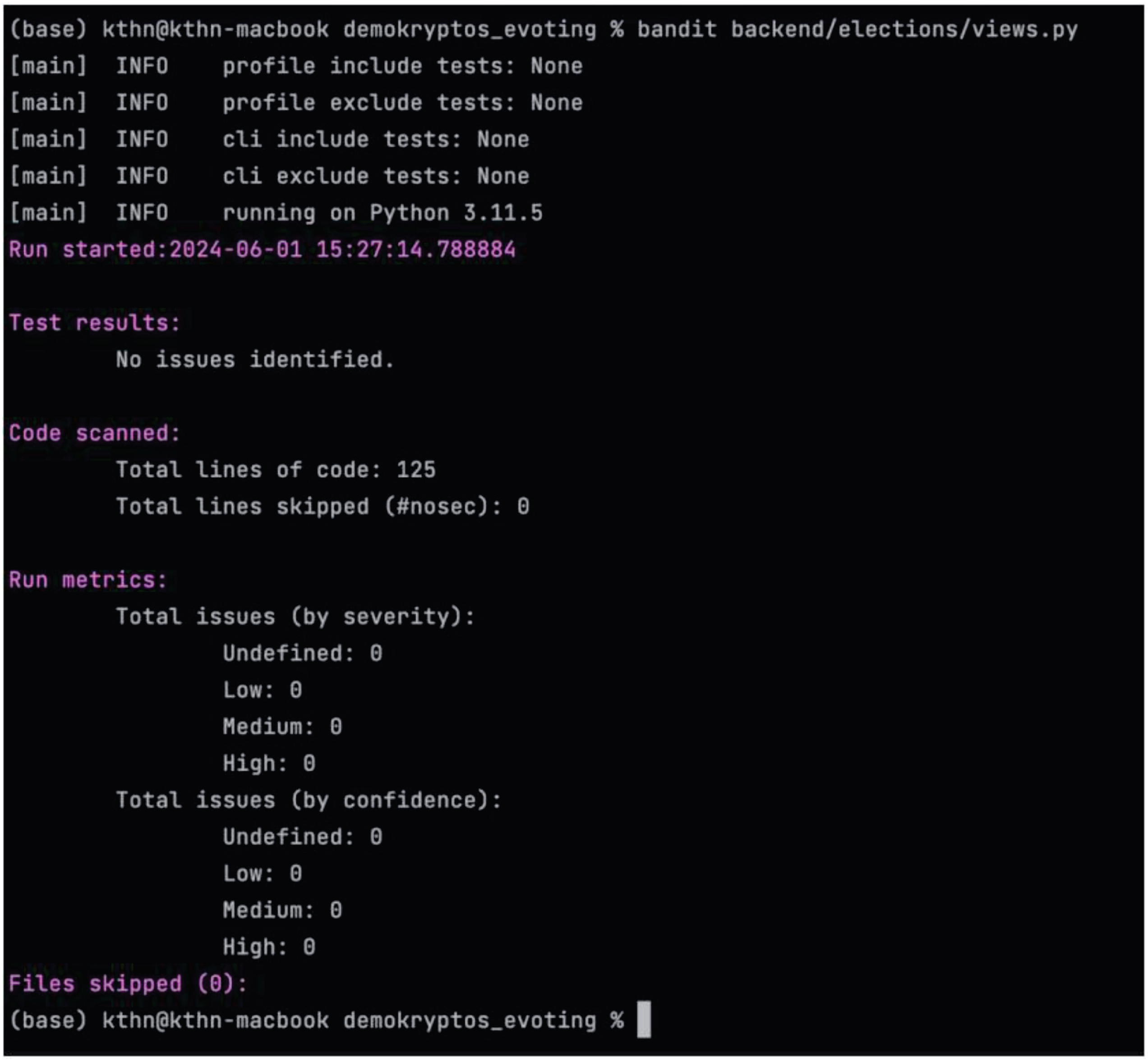
Table 9. Votes per Candidate

Candidate	Votes
Candidate 1	8 votes
Candidate 2	11 votes
Candidate 3	9 votes
Candidate 4	9 votes
Candidate 5	13 votes

This observation leads to the conclusion that the implementation of homomorphic encryption within the electronic voting application is effectively operational.

Simulation of the security processes and vulnerabilities of the proposed system

A security analysis of the Django backend was conducted using **Bandit**, a specialized tool for identifying vulnerabilities in Python code as shown in Figure 5. Bandit systematically scans the codebase for common security issues, such as insecure configurations, improper handling of sensitive data, and vulnerabilities in logic. The analysis included all critical components of the backend, ensuring comprehensive coverage of potential risks.



```
(base) kthn@kthn-macbook demokryptos_evoting % bandit backend/elections/views.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.5
Run started:2024-06-01 15:27:14.788884

Test results:
        No issues identified.

Code scanned:
        Total lines of code: 125
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
Files skipped (0):
(base) kthn@kthn-macbook demokryptos_evoting %
```

Figure 5. Tool for identifying vulnerabilities in Python code

The results of the Bandit scan revealed no security issues in the code, indicating the application's adherence to high security standards and best practices in Python development. All Python files passed the assessment successfully, demonstrating robust risk mitigation strategies and a strong commitment to secure coding practices. These findings affirm the reliability and security integrity of the application, providing confidence in its readiness for production deployment.

Also, a security review of an online voting platform, conducted using the OWASP ZAP tool, revealed seven critical vulnerabilities that require remediation before the system can be safely deployed in production. The primary issue identified was the platform's deployment in a local environment rather than a fully qualified domain. This configuration exposed the system to a range of security risks, as the absence of a proper domain name limited the implementation of robust security measures, such as secure access configurations and proper firewall rules. The findings underscore the importance of adhering to best practices for web application deployment, particularly for systems handling sensitive data like online voting [20].

One significant vulnerability detected was the potential disclosure of cloud instance metadata through the URL path /latest/meta-data/. This issue could lead to the leakage of confidential information if exploited, potentially compromising the platform's integrity and user trust. Proper mitigation strategies, such as deploying the application on a fully qualified domain and configuring NGINX or similar web servers to restrict unauthorized access, are necessary to address this risk. These findings highlight the need for rigorous security assessments during the development of lifecycle to prevent exploitable vulnerabilities in critical systems.

Moreover, the Lighthouse audit was conducted on the web application to evaluate its performance and compliance across key metrics, including Performance, Accessibility, Best Practices, SEO, and Progressive Web Apps (PWA). The application scored 66 for Performance, suggesting areas for improvement in the management of images and scripts, which can lead to faster loading times and a better user experience. The Accessibility score of 95 and the Best Practices score of 93 indicate strong adherence to industry standards in ensuring the application is user-friendly and secure.

The SEO score of 82 highlights the need for optimizations such as adding a meta description and addressing issues with the robots.txt file to improve search engine visibility. Additionally, the absence of key PWA features, such as properly sized icons and manifest entries for *background_color* and *theme_color*, impacts the application's readiness for mobile users. Addressing these gaps by refining image handling, script efficiency, and PWA support would significantly enhance the application's usability, engagement, and overall performance.

Conclusion

This research paper investigates the integration of Paillier's Partial Homomorphic Encryption (PHE) within a web-based electronic voting system to enhance security and integrity in digital voting during emergencies. The system combines democratic principles with advanced encryption technologies, ensuring voter anonymity and data integrity. Development focused on a secure, scalable, and maintainable architecture using Python's libraries and the Paillier encryption algorithm, adhering to rigorous security standards emphasizing soundness, privacy, and verifiability in voting processes. The use of PHE allowed for computations like tallying on encrypted data, protecting sensitive information and enabling secure operations without compromising confidentiality. This demonstrated the practical application of homomorphic encryption in a real-world setting, providing a significant improvement over traditional methods that require decryption for data processing. In conclusion, this study not only meets the demands for a secure electronic voting system but also advances cryptographic applications in digital democracy. The successful implementation of PHE showcases its utility and potential for future enhancements in electronic voting systems, suggesting the feasibility of a secure system that upholds democratic integrity and robust digital security. Future work will explore refining these encryption techniques, their integration with blockchain technology, and expanding cryptographic operations towards Fully Homomorphic Encryption (FHE) for enhanced security and flexibility.

In conclusion, the encryption and decryption processes demonstrated efficient performance, completing secure tallying for 10 voters and 3 candidates in under a minute, with each encrypted vote processing within 1-2 milliseconds. The ciphertext uniqueness tests confirmed consistent generation of unique ciphertexts for identical plaintexts by utilizing varied random values (r), ensuring robust security against repetition attacks. However, the analysis of key size impact revealed that increasing the key size from 512 to 2048 bits resulted in a fourfold increase in computational time, highlighting the importance of selecting a balanced key size to optimize both security and performance.

Additionally, this research effectively demonstrates the integration of Paillier's Partial Homomorphic Encryption (PHE) into an electronic voting system, addressing key challenges in security, voter anonymity, and computational efficiency during emergency situations.

The study focused on applying these cryptographic techniques to e-voting during emergencies such as pandemics or natural disasters. This application, largely unexplored in previous studies, emphasizes the importance of secure voting processes when traditional voting systems may be disrupted.

References

[1] Roopa, K., Gokul, B.S., & Arakalgud, S.K. (2021). Use case of Paillier Homomorphic Algorithm for Electronic-Voting Systems. In *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)* (pp. 273–278). IEEE. https://doi.org/10.1109/ICEECCOT52851.2021.9708030

[2] Faruk, M.J.H., Islam, M.S., Hossain, M.I., Rahman, M.M., Hossain, M.E., & Rahman, M.A. (2022). Development of blockchain-based e-voting system: Requirements, design and security perspective. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 959–967). IEEE. https://doi.org/10.1109/TrustCom56396.2022.00132

[3] Chuprin, V.M., Vyshnyakov, V.M., & Komarnytsky, O. O. (2019). Method of counteracting attacks of a mediator in a transparent system of Internet voting. *Information Protection: Ukrainian Information Security Research Journal, 20*(2), 172–182. Kyiv: NAU. https://doi.org/10.18372/2410-7840.20.13079

[4] Shivers, R., Rahman, M.A., Faruk, M.J.H., Shahriar, H., Cuzzocrea, A., & Clincy, V. (2021). Ride-hailing for autonomous vehicles: Hyperledger Fabric-based secure and decentralized blockchain platform. In *2021 IEEE International Conference on Big Data (Big Data)* (pp. 5450–5459). IEEE. https://doi.org/10.1109/BigData52589.2021.9671379

[5] Patil, S., Bansal, A., Raina, U., Pujari, V., & Kumar, R. (2018). E-smart voting system with secure data identification using cryptography. In *2018 3rd International Conference for Convergence in Technology (I2CT)* (pp. 1–4). IEEE. https://doi.org/10.1109/I2CT.2018.8529497

[6] Sedky, M.H., & Hamed, E.M.R. (2015). A secure e-government's e-voting system. In *2015 Science and Information Conference (SAI)* (pp. 1365–1373). IEEE. https://doi.org/10.1109/SAI.2015.7237320

[7] Taş, R., & Tanrıöver, Ö.Ö. (2020). A systematic review of challenges and opportunities of blockchain for e-voting. *Symmetry, 12*(8), 1328. https://doi.org/10.3390/sym12081328

[8] Taş, R., & Tanrıöver, Ö.Ö. (2021). A manipulation prevention model for blockchain-based e-voting systems. *Security and Communication Networks, 2021*, Article ID 6673691, 16 pages. https://doi.org/10.1155/2021/6673691

[9] Faruk, M.H., Hossain, S., & Valero, M. (2021). EHR data management: Hyperledger Fabric-based health data storing and sharing. In *The Fall 2021 Symposium of Student Scholars*. https://doi.org/10.13140/RG.2.2.20299.05928

[10] Chandel, A., Aggarwal, A., Mittal, A., & Choudhury, T. (2019). Comparative analysis of AES & RSA cryptographic techniques. In *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* (pp. 410–414). IEEE. https://doi.org/10.1109/ICCIKE47802.2019.9004338

[11] Qadir, A. M., & Varol, N. (2019). A review paper on cryptography. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE. https://doi.org/10.1109/ISDFS.2019.8757514

[12] Faruk, M.J.H., Islam, M., Alam, F., Shahriar, H., & Rahman, A. (2022, September). Bie Vote: A biometric identification enabled blockchain-based secure and transparent voting framework. In *2022 Fourth International Conference on Blockchain Computing and Applications (BCCA)* (pp. 253–258). IEEE. https://doi.org/10.1109/BCCA55292.2022.9922588

[13] Galymzhankyzy, Z., Rinatov, I., Abdiraman, A., & Unaybaev, S. (2024). Assessing electoral integrity: Paillier's partial homomorphic encryption in e-voting system. In *2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST)* (pp. 194–201). IEEE. https://doi.org/10.1109/SIST61555.2024.10629522

[14] Galymzhankyzy, Z., Rinatov, I., Abdiraman, A., & Unaybaev, S. (2024). Optimizing e-voting systems: Integration of Paillier cryptosystem and parallel processing for enhanced security and efficiency. In *2024 IEEE AITU: Digital Generation* (pp. 154–160). IEEE. https://doi.org/10.1109/IEEECONF61558.2024.10585388

[15] Willemson, J., & Krips, K. (2023, September). Estimating carbon footprint of paper and internet voting. In *International Joint Conference on Electronic Voting* (pp. 140–155). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43756-4_9

[16] Basal, A., Raina, U., Pujari, V., & Kumar, R. (2018, April). E-smart voting system with secure data identification using cryptography: 2018. In *3rd International Conference for Convergence in Technology (I2CT)* (pp. 1–4). https://doi.org/10.1109/I2CT.2018.8529497

[17] Naser, S.M. (2021). Cryptography: From the ancient history to now, its applications and a new complete numerical model. *International Journal of Mathematics and Statistics Studies, 9*(3), 11–30. https://doi.org/10.13140/RG.2.2.13438.51524

[18] Nurgaliyev, A., & Wang, H. (2021, October). Comparative study of symmetric cryptographic algorithms. In *2021 International Conference on Networking and Network Applications (NaNA)* (pp. 107–112). IEEE. https://doi.org/10.1109/NaNA53684.2021.00026

[19] Shalabi, E., Khedr, W., Rushdy, E., & Salah, A. (2025). A comparative study of privacy-preserving techniques in federated learning: A performance and security analysis. *Information, 16*(3), 244. https://doi.org/10.3390/info16030244

[20] Mehmood, A., Shafique, A., Alawida, M., & Khan, A.N. (2024). Advances and vulnerabilities in modern cryptographic techniques: A comprehensive survey on cybersecurity in the domain of machine/deep learning and quantum techniques. *IEEE Access, 12*, 27530–27555. https://doi.org/10.1109/ACCESS.2024.3367232