

COMPARATIVE ANALYSIS OF PAILLIER AND BFV CRYPTOSYSTEMS FOR BUILDING AN ELECTRONIC VOTING SYSTEM

Jalal Alizada, Ali Huseynli*

Department of Cybersecurity, Azerbaijan Technical University, Baku
Azerbaijan

* Corresponding Author, jalal.alizadeh@aztu.edu.az

Abstract: This study presents a comparative performance evaluation of the Paillier and BFV homomorphic encryption schemes applied to a simplified voting model using one-hot encoded ballots. The experimental setup involved a Python-based implementation of both schemes: a custom implementation of Paillier and the TenSEAL library for BFV. Key performance metrics, including encryption time, homomorphic aggregation time, decryption time, and ciphertext size, were measured across varying voter counts (1,000 to 10,000). Results demonstrate that Paillier has significantly lower aggregation time, while BFV achieves substantially lower encryption and decryption latency, especially under non-batching conditions. However, BFV offers stronger theoretical security guarantees under standard lattice assumptions.

Key words: homomorphic encryption, electronic voting, Paillier, BFV, performance comparison.

1. INTRODUCTION

In the modern world, the demand for secure and scalable electronic voting systems has increased. Although modern systems are becoming increasingly convenient, they still face serious security issues, particularly with regard to confidentiality, integrity, and the reliability of vote counting. Many cryptographic systems exist, but a key feature of the cryptosystem used in electronic voting systems is homomorphism. Among these cryptographic systems are the classical Paillier homomorphic system [1], which is homomorphic with respect to addition, and the Brakerski–Fahn–Vercauteren (BFV) lattice-based homomorphic system, which relies on the ring learning with errors (RLWE) assumption and provides post-quantum security guarantees. Although both systems support homomorphic encryption, their algebraic structures differ significantly.

1.1. Research contributions

The main contributions of this study can be summarized as follows:

1. A unified experimental framework was developed in which both Paillier and BFV were implemented and evaluated under identical hardware and software

conditions. This allows a direct practical comparison rather than a purely theoretical one.

2. Scheme-specific ballot encoding strategies were designed in accordance with the algebraic properties of each cryptosystem. Paillier employs a positional scalar encoding, whereas BFV operates on one-hot vector representations compatible with polynomial ring arithmetic.
3. A quantitative performance analysis was conducted across varying voter populations (1,000–10,000), focusing on encryption time, aggregation cost, decryption time, and ciphertext size.
4. The evaluation deliberately excludes batching techniques in order to reflect the realistic “one vote – one ciphertext” model typical of client-side encryption in practical voting systems.

Unlike many existing works that focus either on theoretical properties or machine learning workloads, this study concentrates specifically on implementation-level trade-offs in a voting-oriented setting.

Thus, the choice of a homomorphic cryptosystem for voting systems depends not only on security assumptions but also on implementation efficiency, scalability, and ciphertext overhead. A balanced assessment must consider both cryptographic security and practical deployment constraints. It is this dual perspective that motivates this comparative study.

2. RELATED WORKS

In early e-voting systems, additive schemes such as Benaloh and Paillier were widely used, as they allow natural aggregation of encrypted votes without decryption until the final stage.

The study presented in [5] focuses on a theoretical comparison of the Benaloh and Paillier schemes, particularly in terms of encoding methods and algebraic efficiency. However, the study is limited to the analytical level and does not include an experimental comparison of implementations under controlled conditions.

More recent research shifts the focus to practical applicability. For example, in [9], e-voting protocols based on partially homomorphic encryption are examined (in particular, Paillier), demonstrating the feasibility of secure vote aggregation without revealing individual ballots.

In [10], Paillier is combined with a block chain infrastructure to increase transparency and verifiability of results. Such solutions aim to enhance trust in the system through immutability of logs and distributed architecture.

Other modern approaches include the integration of homomorphic encryption with zero-knowledge proofs, which simultaneously ensure the correctness of ballots and preserve their confidentiality. Such mechanisms are particularly relevant for large-scale elections, where the requirements for verifiability and transparency increase significantly.

A separate area of research concerns post-quantum solutions. The authors in [7] demonstrated the practical implementability of the BFV lattice scheme using the TenSEAL framework, confirming its applicability in secure computing applications.

However, this and similar studies lack a direct experimental comparison with classical additive schemes under identical implementation conditions.

In contrast to these studies, the present work performs an experimental comparison of Paillier and BFV within a single e-voting model. In this case, batching mechanisms are deliberately disabled to reproduce a realistic client-side encryption scenario in which each vote corresponds to a separate ciphertext—which is typical for practical e-voting systems.

Although existing studies confirm the feasibility of both additive and lattice-based homomorphic schemes in voting applications, direct experimental comparisons under identical implementation constraints remain rare. In particular, many recent works assume batching configurations or focus on secure machine learning workloads rather than voting-specific models. As a result, the practical trade-offs between classical additive cryptosystems and post-quantum lattice constructions in realistic voting environments remain insufficiently quantified.

Recent benchmarking studies comparing major homomorphic encryption libraries such as SEAL, HELib, OpenFHE, and Lattigo [11] provide implementation-level performance insights; however, they do not focus specifically on voting-oriented deployment models.

3. HOMOMORPHIC ENCRYPTION: FUNDAMENTALS

Homomorphic encryption refers to encryption schemes that allow certain algebraic operations to be performed directly on encrypted data, such that the decrypted result matches the outcome of operations performed on the plaintext.

Formally, for an encryption function *Enc*, decryption function *Dec*, and operation $\circ$ on ciphertexts matching operation $*$ on plaintexts:

$$Dec(Enc(m^1) \circ Enc(m^2)) = m^1 * m^2$$

Homomorphism can vary based on the supported operations:

- Partially Homomorphic Encryption (PHE): supports only one operation — e.g., Paillier supports addition only.
- Somewhat Homomorphic Encryption (SHE): supports limited operations of multiple types.
- Fully Homomorphic Encryption (FHE): supports arbitrary sequences of both additions and multiplications.

Early practical schemes like Paillier and BFV later enabled homomorphic encryption to be applied in privacy-sensitive areas like e-voting (e-voting), medical data processing, and cloud computing [3]. Thus, homomorphic encryption empowers secure processing of encrypted data without exposing its content — a key advantage in building next-generation secure information systems.

4. CRYPTOGRAPHIC PARAMETER CONFIGURATION

To ensure a fair and consistent comparison between the Paillier and BFV cryptosystems, both schemes were configured using secure and practically recommended parameters from established sources.

4.1. Paillier Cryptosystem

The plaintext space consists of integers modulo n , and ballots are encoded using a positional plaintext representation to support homomorphic tallying.

The selected configuration provides a classical security level consistent with contemporary recommendations for public-key cryptosystems and is suitable for comparative evaluation with lattice-based homomorphic schemes.

1. Key size: 2048 bits
2. Plaintext space: $\mathbb{Z}_n^2$
3. Ciphertext modulus: $\mathbb{Z}_n^2$
4. Ciphertext size: Approximately 4096 bits
5. Homomorphic property: Additive

4.2. BFV Cryptosystem

The BFV scheme was implemented using the TenSEAL library, a Python wrapper over Microsoft SEAL. Parameters were selected based on the security guidelines.

1. Polynomial modulus degree (n): 8192 (provides ≈ 128 -bit security under the RLWE assumption)
2. Plaintext modulus (t): 1032193 (selected to support one-hot encoded vote vectors and to prevent modular overflow during homomorphic aggregation) (used to encode one-hot vote vectors)
3. Ciphertext modulus (q): ~ 218 -bit modulus chain (default SEAL parameter set)
4. Noise distribution: Discrete Gaussian, standard deviation $\sigma = 3.19$
5. Secret key distribution: Ternary uniform distribution (coefficients $\in \{-1, 0, 1\}$)
6. Relinearization keys: Enabled (to support ciphertext-ciphertext multiplication)
7. Batch encoding: Not used (typical for voting contexts).
8. Bootstrapping: Not used (BFV operates in a levelled mode only)

All operations, including encryption, homomorphic aggregation, and decryption, were executed without ciphertext refreshing. Parameter selection prioritized clarity and reproducibility in the context of secure e-voting, while maintaining compliance with published standards.

5. E-VOTING AND HOMOMORPHIC ENCRYPTION

E-voting allows elections to be conducted digitally, from casting to counting votes. Critical requirements include confidentiality, integrity, and verifiability of votes, while maintaining transparency and public trust.

Homomorphic encryption allows vote aggregation without decryption — each vote is encrypted client-side and transmitted securely. Authorities can then tally votes directly over ciphertexts, decrypting only the final aggregated result.

Benefits of this approach include:

- Confidentiality: Individual votes remain secret, even to election organizers.
- Reduced infrastructure trust: No intermediate decryption minimizes risk.
- Verifiability: Cryptographic proof (e.g., zero-knowledge) ensures correctness.

- Tamper resistance: Any tampering with ciphertexts disrupts data integrity and is detectable.

Homomorphic encryption provides a technically viable mechanism for privacy-preserving tallying in digital voting systems.

6. ALGORITHM IMPLEMENTATION

Python was chosen for implementation due to its ease of use and rapid development capabilities.

A manual implementation of the Paillier cryptosystem was developed in Python to provide transparency and fine-grained control over cryptographic operations. The source code of the implementation is publicly available [8].

BFV was implemented using TenSEAL [4], a Python wrapper for Microsoft SEAL, enabling homomorphic operations via high-level APIs

To ensure consistency, all tests were run on identical hardware configurations (Windows 11, Intel i9, 16GB RAM), and operation times were measured using Python's time module.

6.1. Simulation Model

A simplified e-voting model was used to evaluate the Paillier and BFV homomorphic encryption schemes. The model considers a fixed number of candidates and a variable number of voters. Each voter casts exactly one vote for a single candidate. The simulation covers the three main phases of a homomorphic voting system: ballot encryption, homomorphic aggregation, and decryption of the final tally, which is consistent with standard evaluation methodologies in homomorphic e-voting systems.

For the Paillier cryptosystem, ballots are encoded using a scalar positional representation. Let C denote the number of candidates and V the number of voters. A public base b is selected such that $b > V$. A vote for candidate $i \in \{0, \dots, C - 1\}$ is represented as a plaintext integer $m_i = b^i$.

Due to the additive homomorphic property of the Paillier cryptosystem, ciphertext multiplication corresponds to plaintext addition. After homomorphic aggregation and decryption, the resulting plaintext has the form

$$M = \sum_{i=0}^{C-1} c_i b^i \quad (1)$$

where c_i denotes the number of votes received by candidate i , and b is the positional base selected such that no overflow occurs during aggregation.

For the BFV scheme, ballots are represented as one-hot encoded vectors of length C , where the component corresponding to the selected candidate is set to one and all remaining components are set to zero. Each ballot is encrypted as an independent ciphertext, and homomorphic aggregation is performed via vector addition. This vector-based representation is naturally aligned with the polynomial structure of lattice-based homomorphic encryption schemes such as BFV and is commonly adopted in practical implementations based on SEAL and TenSEAL.

The use of scheme-specific plaintext representations reflects the algebraic characteristics of the underlying cryptosystems and follows established practices in homomorphic encryption literature. This approach enables a consistent evaluation of encryption, aggregation, and decryption costs while preserving the security assumptions of both Paillier and BFV.

Simulations were conducted for varying numbers of voters:
[1000,2500,5000,7500,10000]

For each count, the process included: key generation → encryption → encrypted aggregation → decryption.

6.2. Results of implementation

Results for encryption time are presented in Figure 1 and Table 1. As the number of voters increases from 1,000 to 10,000, both schemes exhibit a linear growth in runtime; however, the magnitude and growth rate differ significantly.

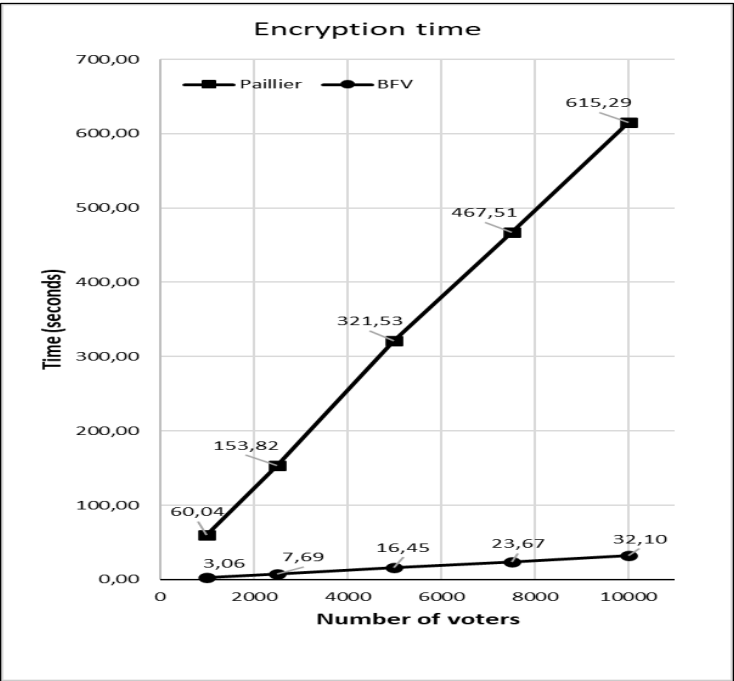


Figure 1. Encryption Time Graph

Table 1. Encryption implementation

# of voters	Paillier Time(s)	BFV Time(s)	Difference (BFV-Paillier)
1000	60.04	3.05	-56.99
2500	153.82	7.69	-146.13
5000	321.53	16.45	-305.08
7500	467.51	23.67	-443.83
10000	615.29	32.10	-583.19

As expected, the encryption time of Paillier increases from approximately 60 s for 1,000 voters to over 600 s for 10,000 voters. This behaviour is primarily attributed to the high computational cost of modular exponentiation in $\mathbb{Z}_n^2$.

In contrast, the BFV scheme demonstrates lower encryption time. With 1,000 voters, the process takes about 3 seconds, and with 10,000 voters, about 32 seconds. This result is in line with expectations, given the optimized implementation of polynomial arithmetic in the SEAL library and the use of a ring of fixed degree. Even when encrypting vector ballots without batching, the computational cost remains significantly lower than Paillier while maintaining a comparable level of cryptographic security.

Figure 2 and Table 2 present the results of homomorphic summation (aggregation) for both encryption schemes. This stage simulates the server-side accumulation of encrypted votes. This is a core operation in e-voting systems based on homomorphic encryption.

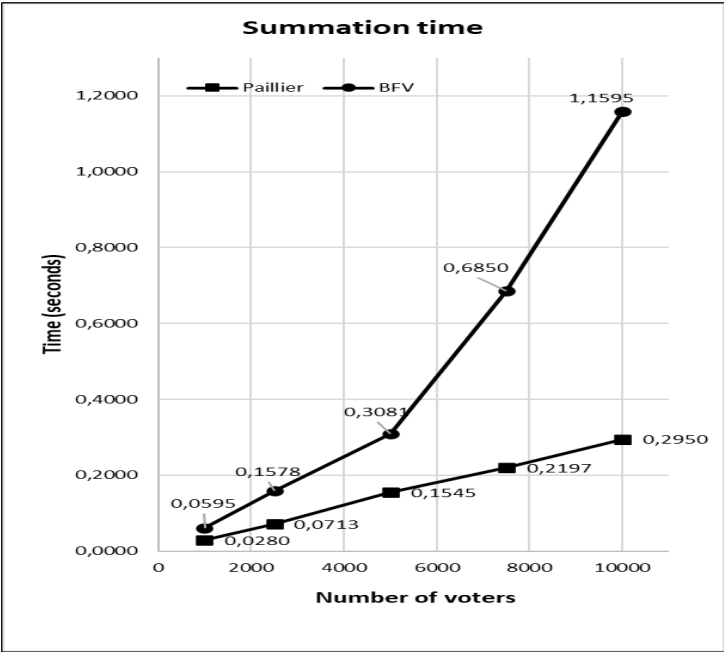


Figure 2. Summation Time Graph

Table 2. Summation implementation

# of voters	Paillier Time(s)	BFV Time(s)	Difference (BFV-Paillier)
1000	0.0288	0.0595	+0.0307
2500	0.0713	0.1518	+0.0805
5000	0.1545	0.3081	+0.1536
7500	0.2197	0.6850	+0.4653
10000	0.2950	1.1595	+0.8645

In the case of the Paillier-based implementation, applying homomorphism aggregation distributes all the encrypted ballots into one ciphertext through multiplication, equivalent to summing up all the plaintext representations using a positional structure as described by (1). As a result, the aggregation cost grows linearly with the number of voters and remains low even for large-scale elections. Specifically, the summation time increases from approximately 0.05 s for 1,000 voters to below 0.3 s for 10,000 voters.

This efficiency is a result of the direct connection between the additive homomorphic property of the Paillier cryptosystem, as the multiplication of the ciphertext is successfully translated to a single modular multiplication over $\mathbb{Z}_n^2$. This is because only one aggregated ciphertext results, hence a minimal overhead in terms of computation on the server.

In contrast, the BFV scheme performs aggregation through repeated vector additions over encrypted one-hot ballots. Although this process also exhibits linear scalability, it involves polynomial arithmetic over large rings and internal Residue Number System (RNS) representations, which introduce additional computational costs. Consequently, BFV aggregation time increases from approximately 0.08 s at 1,000 voters to over 1.0 s at 10,000 voters.

Decryption performance results are summarized in Figure 3 and Table 3. In this experiment, both schemes decrypt a single aggregated ciphertext representing the final vote tally — a typical operation at the end of a homomorphic voting process.

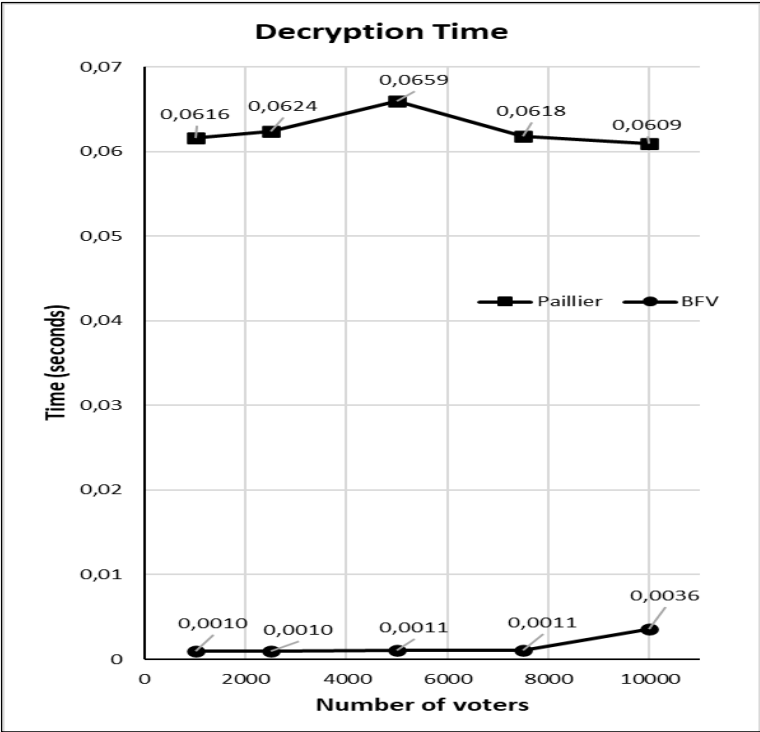


Figure 3. Decryption Time Graph

Table 3. Decryption implementation

# of voters	Paillier Time(s)	BFV Time(s)	Difference (BFV-Paillier)
1000	0.0616	0.0010	-0.0606
2500	0.0624	0.0010	-0.0614
5000	0.0659	0.0011	-0.0648
7500	0.0618	0.0011	-0.0607
10000	0.0609	0.0036	-0.0573

In both cases, only a single aggregated ciphertext is decrypted, making the decryption cost effectively independent of the number of voters. For Paillier, decryption time remains nearly constant at approximately 0.05–0.06 s across all configurations, reflecting the fixed cost of modular exponentiation and application of the L-function.

BFV decryption exhibits lower latency, remaining below 0.005 s for all voter counts due to efficient polynomial-based decryption implemented in the SEAL backend. Although BFV is faster in absolute terms, the difference is marginal in practical voting scenarios, as decryption is performed only once on the final tally.

This outcome may initially appear counterintuitive, given BFV’s heavier encryption and aggregation workloads. However, BFV’s decryption is designed to be lightweight and efficient, assuming well-chosen parameters and sufficient remaining noise budget.

For the Paillier cryptosystem, each encrypted ballot is represented by a single ciphertext due to the use of positional plaintext encoding. Under the evaluated configuration, the ciphertext size per vote is approximately 4096 bits, corresponding to the modulus size of the scheme. Since only one ciphertext is generated per ballot, the storage overhead grows linearly with the number of voters and remains relatively compact. In contrast, BFV ciphertexts generated with a plaintext modulus of 1032193 and batching disabled require approximately 3–5 KB per encrypted ballot. The larger ciphertext size is primarily a consequence of the polynomial representation used in lattice-based homomorphic encryption, which stores multiple coefficient vectors along with auxiliary metadata and parameters managed internally by the TenSEAL library.

While BFV ciphertexts are significantly larger, their structure enables batching optimizations. In this study, batching was intentionally disabled to ensure one-to-one vote representation, but enabling batching could reduce ciphertext count and amortize overhead.

When analysing the performance of these encryption schemes, the encryption time is less critical for server scalability but remains relevant for client-side usability. In real-world scenarios, encryption is typically performed on the client side, and encrypted ballots are transmitted to the server in advance. As a result, server-side computational resources are not utilized during the encryption process. Therefore, the primary focus should be placed on the stages of homomorphic aggregation and decryption of the final voting results. According to the experimental results, as the number of voters increases, BFV demonstrates faster encryption and decryption performance. However, during the

vote aggregation phase, it requires greater computational resources and is less efficient compared to the Paillier scheme. This behaviour is primarily due to the computational complexity inherent in BFV. It operates on polynomial structures, performs arithmetic over modular rings, and requires continuous management of ciphertext noise. These factors introduce substantial overhead as data volume increases.

7. LIMITATIONS

One important consideration in this study is the omission of batching in the BFV implementation. This decision is consistent with the operational principles of e-voting systems, where each voter encrypts and transmits a single vote independently. While batching techniques are effective in scenarios where multiple messages originate from the same source and can be aggregated prior to encryption, such conditions do not hold in decentralized voting environments. As a result, the use of non-batched ciphertexts provides a more accurate representation of the performance characteristics expected in real-world e-voting applications.

Additionally, the Paillier implementation used 2048-bit keys to maintain a practical balance between performance and security. While still not post-quantum secure, this size provides acceptable baseline security for symmetric equivalence and fair comparison. Future work could explore stronger Paillier variants and hybrid protocols.

8. CONCLUSION

The experimental evaluation presented in this study provides a comparative analysis of the Paillier and BFV homomorphic encryption schemes within a unified e-voting simulation model. The experimental data confirm that Paillier and BFV exhibit distinct computational profiles arising from their underlying algebraic constructions.

In terms of computational complexity, aggregation in the Paillier scheme is implemented via modular multiplications, while BFV aggregation relies on polynomial additions in a lattice-based ring structure.

Formally, both schemes exhibit linear scalability in the number of voters. However, their practical computational costs differ. In BFV, additional overhead arises from the representation of coefficients in the residue number system (RNS) and the need to control noise growth in the ciphertext. These factors increase the constant complexity factors and make aggregation in the lattice scheme more resource-intensive compared to the classical additive Paillier cryptosystem.

Future work may further extend the analysis to more complex voting rules, such as ranked-choice or multi-selection ballots, and explore advanced techniques including batching, ciphertext packing, and alternative homomorphic encryption schemes such as CKKS or BGV. A deeper investigation into parameter selection, noise growth behaviour, and ciphertext compression strategies would further support the development of scalable and secure e-voting systems.

REFERENCES

- [1] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *EUROCRYPT '99, Lecture Notes in Computer Science*, vol. 1592, Springer, Berlin, Heidelberg, 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16.
- [2] J. Fan, F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, Report 2012/144, 2012 (Online), Available: <https://eprint.iacr.org/2012/144>
- [3] M. R. Albrecht, R. Player, S. Halevi, et al. Homomorphic encryption security standard. *HomomorphicEncryption.org White Paper* [Online]. Available: <https://homomorphicencryption.org>. (Visited on 20.02.2026)
- [4] OpenMined. TenSEAL: A library for homomorphic encryption on tensors using Microsoft SEAL, GitHub Repository. [Online]. Available: <https://github.com/OpenMined/TenSEAL>. (Visited on 20.02.2026)
- [5] Salman, W.D. Analysis of Benaloh and Paillier homomorphic cryptosystems for building an electronic voting system. *Problems of Information Security. Computer Systems*, vol. 7, no. 2, 2021, pp. 102–109 (in Russian).
- [6] F. Doro, M. Barbosa, D. Stebila. Comparative performance of homomorphic encryption schemes for secure machine learning. *Proceedings on Privacy Enhancing Technologies (PoPETS)*, vol. 2021, no. 1, 2021, pp. 5–24. DOI: 10.2478/popets-2021-0002.
- [7] M. Kim, Y. Song, S. Wang, Y. Xia, X. Jiang. Secure logistic regression based on homomorphic encryption: Design and evaluation. *BMC Medical Genomics*, vol. 14, no. 1, 2021, pp. 1–10. DOI: 10.1186/s12920-021-00916-5.
- [8] Alizada, J. Comparative analysis of Paillier and BFV homomorphic encryption schemes for electronic voting (source code). GitHub repository, 2025. [Online] Available: <https://github.com/futures1mple/pailliersbfv> (Visited on 20.02.2026).
- [9] Y. Zhan, K. Sang, S. Zhang, X. Chen, W. Yang. Efficient electronic voting system based on homomorphic encryption and decentralization. *Electronics*, vol. 13, no. 2, Feb. 2024, Article 286. DOI: 10.3390/electronics13020286
- [10] B. Jayakumari, S. L. Sheeba, M. Eapen, J. Anbarasi, V. Ravi, A. Suganya, M. Jawahar. E-voting system using cloud-based hybrid blockchain technology. *Journal of Safety Science and Resilience*, vol. 5, no. 1, Mar. 2024, pp. 102–109. DOI: 10.1016/j.jnlssr.2024.01.002
- [11] Pambudi, D. S., Studiawan, H., Pratomo, B. A., Purwitasari, D. Performance analysis of leading homomorphic encryption libraries: A benchmark study of SEAL, HELib, OpenFHE, and Lattigo. *Proceedings of the 2025 4th International Conference on Cyber Security, Artificial Intelligence and the Digital Economy (CSAIDE 2025)*, Malaysia, 2025, pp. 25–30. DOI: 10.1145/3729706.3729711.
- [12] Emami, A., Yajam, H., Akhaee, M. A., Asghari, R. A scalable decentralized privacy-preserving e-voting system based on zero-knowledge off-chain computations. *Journal of Information Security and Applications*, vol. 79, Dec. 2023, Article 103645. DOI: 10.1016/j.jisa.2023.103645.

Information about the authors:

Jalal Alizadeh – PhD student in Information Systems and Processes. Affiliation: Azerbaijan Technical University, Azerbaijan. Research interests include the development and optimization of cryptographic algorithms for electronic voting systems, homomorphic encryption, and information security.

Ali Huseynli – PhD student in Information Systems and Processes. Affiliation: Azerbaijan Technical University, Azerbaijan. Research interests include the development and optimization of cryptographic algorithms for resource-constrained IoT devices, lightweight cryptography, information security, software engineering and distributed systems.

Manuscript received on 08 February 2026