Published online on the page : https://ejournal.uinbukittinggi.ac.id/index.php/ijokid/

Knowbase : International Journal of Knowledge in Database

| ISSN (Print) 2798-0758 | ISSN (Online) 2797-7501 |

BFV Homomorphic Encryption Algorithm as a Proposed Encryption Mechanism for the Votenow System of PT XYZ

Muhamad Ikmal Wiawan[1,], Agi Agus Setiawan Sufyan[2]*
[1,2]Universitas Islam Nusantara, Bandung, Indonesia

Article Information

Article History:

Submitted: October 8, 2025
Revision: December 15, 2025
Accepted: December 23, 2025
Published: December 30, 2025

Keywords

Homomorphic 1
BFV 2
E-Voting 3
Chipertext Verification 4
Decryption Verfication 5

Correspondence

*E-mail: ikmal.wiawan@gmail.com**

A B S T R A C T

Confidentiality and integrity of voting results constitute major challenges in web-based e-voting systems, as vote tallying in conventional approaches still requires data decryption. This condition potentially enables intervention by private key holders and reduces trust in election outcomes. The votenow e-voting platform of PT XYZ does not yet support vote tallying in an encrypted state; therefore, an alternative encryption mechanism is required that does not significantly alter the existing system workflow. This study aims to evaluate the BFV (Brakerski–Fan–Vercauteren) Homomorphic Encryption algorithm as a proposed encryption mechanism for the PT XYZ e-voting system (product name anonymized). A controlled experimental method was applied using a testing prototype with encryption and decryption modules implemented in C++ based on the Microsoft SEAL library, while a PHP-based web interface was employed for data input and visualization. The evaluation assessed the time required to input 50,000 encrypted votes, vote tally accuracy using both decryption-based counting and direct ciphertext computation without decryption, total ciphertext size, verification time for encrypted data validity, ciphertext decryption time, and vote result presentation time. The results indicate that the input of 50,000 votes was completed within 5 minutes, meeting the 10-minute target. Vote tally accuracy reached 100% for both counting methods, and the ciphertext size of 383.4 MB remained below the 512 MB threshold. Furthermore, the encrypted data verification time was recorded at 225.8 seconds, ciphertext decryption time at 5 minutes and 15 seconds, and vote result presentation time via decryption at 13.816 seconds, all of which fall within acceptable operational limits. Based on these findings, the BFV algorithm is considered suitable for adoption as an encryption mechanism in the PT XYZ e-voting system, as it enables vote tallying in the encrypted domain while preserving the confidentiality and integrity of voter data.

1. Introduction

. The rapid advancement of information technology has gradually encouraged various stakeholders to transition toward web-based electronic voting systems. E-voting is considered capable of improving the efficiency, transparency, and accuracy of the electoral process; however, it simultaneously introduces significant challenges related to data security and the integrity of voting results. One fundamental weakness in many conventional e-voting systems is their reliance on decryption during the vote tallying phase, which technically creates opportunities for intervention by private key holders and may consequently reduce public trust in the authenticity of election outcomes[1]. Amid the growing need for secure tallying mechanisms, Homomorphic Encryption (HE) technology offers an approach that enables computations to be performed directly on ciphertext without revealing the underlying data through decryption[2]. The Brakerski–Fan–Vercauteren (BFV) algorithm is a Homomorphic Encryption (HE) scheme designed to efficiently support

 Creative Commons Attribution-ShareAlike 4.0 International License. *Some rights reserved*

integer arithmetic operations on encrypted data, making it well suited for e-voting scenarios that require a balance among security, accuracy, and performance[3].

Previous studies have demonstrated the potential of Homomorphic Encryption (HE) to enhance the security of voting processes; however, most of these works remain limited to prototype implementations or constrained simulations. Zhan et al.[1] demonstrated that HE is capable of preserving the privacy of voters' ballots, whereas Hu et al. [2] developed a multi-key BFV variant that enhances security aspects but still faces challenges related to the complexity of the relinearization process. Yuan et al. [3] emphasized that integrating Homomorphic Encryption with decentralization approaches can reduce reliance on third parties. Kraavi dan Willemson highlighted the need for cryptographic proofs to verify the correctness of votes, while Umar et al. [5] demonstrated that the Paillier scheme can preserve privacy but produces larger ciphertexts and is less efficient for large-scale deployments. Manzanares-Lopez dan Cano investigated the performance of the Microsoft SEAL library for Fully Homomorphic Encryption (FHE) on large datasets; however, their work did not relate the evaluation to web-based e-voting implementations. In addition, several other studies have emphasized the presence of side-channel and timing attacks in traditional encryption mechanisms, further underscoring the necessity of tallying mechanisms that do not require decryption. Collectively, these studies indicate that although HE approaches, including BFV, are built on strong theoretical foundations, there remains a research gap in evaluating BFV implementations directly within web-based e-voting systems operating in near-realistic environments. This study addresses this gap by adapting a web application environment and a representative number of voters aligned with the real-world e-voting requirements of PT XYZ.

In the Indonesian context, PT XYZ (name anonymized) is a company that develops an e-voting platform known as VoteNow, which is utilized by various institutions for online voting. The system is functionally stable; however, the entire vote recapitulation process still relies on decryption, thereby leaving residual risks of intervention by private key holders. Addressing this gap, this study proposes the application of the BFV algorithm to construct a high-performance secure comparison protocol that enables comparison operations to be performed entirely on encrypted data without revealing the actual values. [7]. The protocol evaluated in this study focuses on accelerating large-integer comparison using BFV ciphertexts by measuring parameters such as the number of required encryptions, computation time, ciphertext size, and decryption efficiency, thereby providing a more comprehensive performance assessment than previous studies, which have largely been confined to small-scale algorithmic aspects.

Studies such as Al Badawi et al. [8], k laine et al. [9] , and Pedrouzo-Ulloa et al. [10] have established strong mathematical and algorithmic foundations, as well as insights into BFV and RLWE-based security; however, they primarily focus on theoretical aspects and computational optimizations and do not address direct implementation within e-voting systems. Building upon this gap, the present study provides an empirical contribution by evaluating the application of BFV on a commercial, web-based e-voting platform. The results demonstrate that vote tallying can be performed entirely within the encrypted domain with 100% accuracy for a large number of votes (50,000 ballots), efficient processing time, and storage requirements that remain within system constraints. These findings confirm the feasibility of BFV for deployment in production environments and highlight its potential to enhance e-voting security, both in the PT XYZ case study and more broadly for similar e-voting software products in the future.

2. Method

This study employs a controlled experimental method to evaluate the application of the BFV Homomorphic Encryption algorithm within the context of a web-based e-voting system at PT XYZ. This approach is used to examine how the BFV algorithm operates under conditions that closely resemble a real operational environment, particularly in the processes of vote encryption, encrypted vote data processing, and decryption of aggregated tally results. In deploying BFV within an operational setting, noise growth remains an aspect that must be considered, although it is not quantitatively analyzed in this study. Aggarwal et al.

noted that noise introduced during encryption and homomorphic operations can potentially affect the success of the decryption process, both when using the secret key and when decrypting final computation results [11]. Therefore, although noise is not treated as an analyzed variable, verification of the computation results—including the processing of encrypted votes remains necessary to ensure that the homomorphic processes do not produce decryption errors due to potential noise accumulation.

Prior to testing, a public–private key pair was generated along with the configuration of BFV parameters, including the polynomial degree, coefficient modulus, and plaintext modulus, all selected to meet a minimum 128-bit security standard. These configurations were kept consistent to ensure objective evaluation of the experimental results. The subsequent stage involved the creation of a dataset comprising 50,000 dummy entries, each consisting of a randomly generated 16-digit National Identification Number (NIK) and a candidate choice valued from 1 to 5, in order to simulate a large-scale voting process aligned with the operational requirements of PT XYZ..

The testing procedure was designed to follow the e-voting workflow as described in the scheme evaluated by Chillotti et al [12] , It includes the process of encrypted vote input, accuracy verification through decryption of the final results, and correctness validation via direct operations on ciphertext, as performed during the homomorphic tallying stage. In addition, evaluations of ciphertext size and decryption time were conducted to examine the characteristics of the final computation results, in line with the implementation practices presented in the referenced study. To ensure that the experiments were conducted in a measurable and systematic manner, testing parameters, targets, and hypotheses were defined based on the evaluation approach recommended in homomorphic encryption–based e-voting schemes (VoteNow). The following table summarizes all testing parameters used as references in the experiment.

Table 1. Experimental Parameters and Evaluation Criteria

No.	Testing Parameter	Parameter Description	Measurement Tool / Method	Target / Hypothesis
1	Vote Input Time (including parameters)	Time required to input 50,000 vote records into the e-voting system with randomly selected candidates	Web API client using curl on Linux CLI	Maximum 10 minutes
2	Vote Accuracy Using Decryption Method	Comparison of vote tally results for all candidates between plaintext data (without encryption) and ciphertext (Homomorphic BFV) through the decryption process	Web-based decryption feature integrated with BFV and Microsoft SEAL C++ using the private key, compared with plaintext tallying from the database	100% accuracy
3	Vote Accuracy Using Ciphertext Without Decryption	Comparison of vote tally results for all candidates between plaintext data and results obtained directly from ciphertext computation without decryption	Web-based "Vote Counting" feature integrated with BFV without using the private key, compared with plaintext results	100% accuracy
4	Encrypted Data Size (Ciphertext)	Total size of encrypted data for 50,000 votes	Inspection of ciphertext file/table size in the storage system	Maximum 512 MB
5	Encryption Validity Verification Time	Time required to verify the validity of encrypted vote data for 50,000 votes	Measurement of verification processing time through the BFV module	Maximum 10 minutes
6	Ciphertext Decryption Time	Time required to decrypt encrypted vote data for 50,000 votes using the BFV decryption method	Web-based decryption feature using the private key configured on the server	Maximum 10 minutes
7	Vote Result Presentation Time (Decryption Method)	Time required to display vote tally results after the decryption process	Measurement of execution time for the vote result	Maximum 10 minutes

			display feature on the web interface	

After the testing parameters were defined as summarized in the table, the experiment was designed to be executed sequentially to evaluate all aspects of the BFV algorithm implementation. The first test focused on the vote processing time, or encrypted vote input time, namely the duration required to encrypt and store a given number of votes, for example 50,000 entries. This approach follows BFV benchmarking practices such as those employed in FHEBench, where encryption latency and throughput are quantitatively evaluated [13]. During this stage, the system recorded the total processing time and the average time per entry and compared them against the predefined maximum target time (e.g., 10 minutes), similar to the latency metrics measured in BFV implementation studies on GPUs. This evaluation aimed to assess the capability of the BFV algorithm to handle large-scale vote input processing in accordance with the established testing parameters. The testing parameters, targets, and hypotheses were formulated based on evaluation practices recommended in the literature to ensure that the experiments were conducted in a measurable and systematic manner.

In the second stage, all ciphertexts were decrypted using the private key, and the results were verified one by one against the original plaintext to ensure full equivalence. This procedure is consistent with analyses of modified BFV schemes designed to preserve decryption correctness even under randomized evaluation conditions [14], and is further supported by practical experiments demonstrating that BFV decryption can maintain high accuracy after various encrypted computational operations [15]. A target accuracy of 100% was selected to preserve system integrity and prevent any distortion of vote values.

The third stage evaluated vote tally accuracy through direct operations on ciphertext without performing decryption during the process. This approach follows the homomorphic e-voting scheme described by Chillotti et al., in which tallying is carried out entirely within the encrypted domain and only the final result is decrypted to verify its consistency with plaintext computations[12]. The BFV algorithm supports ciphertext-to-ciphertext addition, enabling vote aggregation to be performed securely without revealing individual values. Therefore, a target accuracy of 100% was established to ensure that all addition operations within the encrypted domain produce results that remain consistent when verified through final decryption.

Storage efficiency analysis was conducted by summing the file sizes of all 50,000 serialized ciphertexts and comparing the result with the 512 MB capacity limit enforced by the PT XYZ system. This procedure follows standard practices in modern FHE benchmarking studies, which commonly evaluate ciphertext size as part of assessing storage overhead and implementation scalability [13][16][17]. In addition, this analysis considers findings from studies that investigate the deployment of FHE on resource-constrained infrastructures, particularly with respect to the relationship between processing time characteristics, ciphertext size, and storage requirements [18].

The subsequent stage measured the decryption time of the encrypted vote tally results, namely the duration required to transform the aggregated ciphertext into plaintext. As this is the only process that requires the private key, a maximum time limit of 10 minutes was established to ensure that the recapitulation does not impede the e-voting workflow. In addition, the decryption time of individual votes was also evaluated to simulate the need for granular auditing. This approach aligns with findings from previous studies indicating that BFV decryption can be a critical aspect to consider in practical deployments [16][19][20].

The entire testing procedure was repeated multiple times to obtain more stable and representative average values. The evaluated variables included vote input time, decryption accuracy, ciphertext-based tally accuracy, ciphertext size, final result decryption time, and individual vote decryption time. All experimental outcomes were subsequently compared against the predefined performance targets established by PT XYZ. This comparative analysis was conducted to assess the suitability of the BFV algorithm in meeting the system's security requirements and operational constraints when employed as an encryption mechanism for e-voting. To facilitate understanding of the research workflow, the overall methodology is illustrated through the following activity diagram

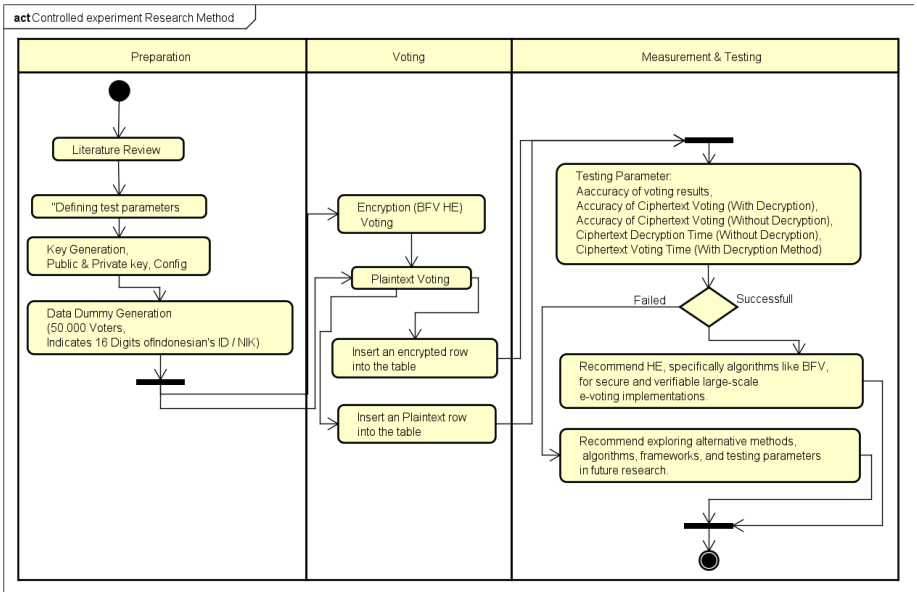


***Figure 1**. Research Logic Flow*

3. Results and Discussion

3.1 Encryption Algorithm

The Brakerski–Fan–Vercauteren (BFV) algorithm is a homomorphic encryption scheme specifically designed to enable data processing without revealing the underlying plaintext values [8]. Its fundamental principle is that mathematical operations, such as addition and multiplication, can be executed directly on encrypted data, yielding results equivalent to those obtained when the same operations are performed on unencrypted data [8]. This approach is particularly critical in electronic voting systems, as it preserves the confidentiality of voters' ballots while simultaneously enabling secure and transparent verification of election results [8].

In practice, the BFV algorithm operates by leveraging complex polynomial structures and the security foundations of the Ring Learning with Errors (RLWE) problem, which is considered computationally intractable using conventional methods[21]. ncrypted data remain in ciphertext form throughout the entire computation process, allowing vote tallying to be performed entirely within the encrypted domain[22]. Only after all votes have been aggregated is the final result decrypted, revealing the total number of valid votes without ever exposing individual voters' selections [23].

One of the primary advantages of the BFV scheme lies in its ability to perform exact integer computations without relying on rounding operations that could otherwise introduce result discrepancies [14]. Experimental studies have demonstrated that BFV achieves efficient performance with manageable encrypted data sizes and processing times that are suitable for large-scale systems[23]. More recent developments further indicate that BFV efficiency can be enhanced through hardware-level optimizations and the adoption of parallel computing techniques to accelerate both encryption and encrypted data processing tasks [24]

Nevertheless, the deployment of BFV introduces specific challenges, as each homomorphic operation increases the noise level within the ciphertext, which may degrade result correctness if not properly managed [25]. Consequently, careful selection of encryption parameters and the application of appropriate algorithmic optimization strategies are essential to ensure that the scheme remains stable and accurate when applied to large-scale voting systems [26]. Overall, the BFV algorithm provides a robust solution for preserving the confidentiality and integrity of votes in e-voting applications, while supporting the advancement of more secure and trustworthy digital democracy systems [27]. The algorithm comprises the following core components:

3.1.1 Secret Key Generation

The secret key generation process constitutes the initial stage of the BFV homomorphic encryption scheme and is responsible for producing the private key. This key serves as a fundamental element used throughout both encryption and decryption procedures to ensure data confidentiality and consistency [28]. As the core component of the security mechanism, the secret key grants exclusive capability to recover encrypted data into its original plaintext form, making it accessible only to the legitimate key holder [10]. In the BFV algorithm, the secret key is generated using a randomized approach grounded in mathematical distribution theory. The resulting random polynomial is embedded within a specific modular space known as a polynomial ring, with the generation process adhering to security principles based on the Ring Learning with Errors (RLWE) problem [29]. This structure ensures that, even if an adversary gains access to the ciphertext, deriving the secret key directly remains computationally infeasible due to the high mathematical complexity involved. Furthermore, the security strength provided by the secret key is highly dependent on the parameter choices employed during its construction. Parameters such as the polynomial degree and modulus size must be carefully selected to achieve an optimal balance between security and system efficiency [17]. Choosing parameters that are too small may weaken security guarantees, whereas excessively large parameters can significantly degrade system performance. In BFV, the secret key is defined within a polynomial ring framework, which can be formally expressed as follows:

$$\boldsymbol{s = Zq[X]/(XN + 1)} \qquad \textit{(1)}$$

The primary parameter required during secret key generation is the value NNN, which represents the polynomial degree and must be a power of two, such as 1024, 2048, 4096, 8192, and so forth. As the polynomial degree increases, the resulting ciphertext size also grows accordingly. The following table illustrates the ciphertext sizes produced under different polynomial degree configurations during encryption.

Table 2. Ciphertext Size and Storage Requirements for Different Polynomial Degrees in BFV

Polynomial Modulo (n-bit)	Ciphertext Size	100K Data Size
1024	~8kb	~800MB
2048	~16kb	~1.6 GB
4096	~66kb	~6.6 GB
8192	~263kb	~26.3 GB

3.1.2 Public Key Generation

The public key is constructed using the following mathematical formulation:

$$(a.b = -a.s + e) \qquad (2)$$

In the formulation above, the secret key sss is utilized in the construction of the public key. This public key subsequently serves as the primary component for encrypting data within the BFV encryption process.

3.1.3 Encryption

During the encryption process, the ciphertext is denoted by ccc and is generated using the following formulation, where r represents a randomly sampled value:

$$(c = a \cdot r + e1, b \cdot r + e2 + \Delta \cdot m) \qquad (3)$$

3.1.4 Decryption

The decryption stage in the BFV algorithm is responsible for transforming ciphertext back into plaintext so that the original data can be interpreted. In this phase, the ciphertext is processed using the secret key and the parameters defined in the system configuration, resulting in a recovered message that is identical to the data prior to encryption. In addition, BFV supports direct mathematical operations on ciphertexts, such as addition and multiplication—without requiring intermediate decryption. This capability makes the BFV algorithm highly effective for applications that demand strong security and efficient encrypted data processing, including its deployment in e-voting systems.

3.2 Initial Implementation and Functional Testing of BFV

The interpreter and interface of this study were developed using native PHP and C++ with support from the Microsoft SEAL library. Native PHP serves as the user-facing interface for the voting process, as well as for managing data input and storage within the database, while the C++ program functions as the cryptographic module responsible for encryption, decryption, and vote computation using the BFV algorithm. At this stage, the implementation was focused on preliminary testing to verify that the BFV-based C++ program operates correctly and communicates reliably with the PHP application. Candidate selection data transmitted from PHP are encrypted by the C++ module, and the resulting ciphertext is returned to PHP for storage. This initial testing was conducted prior to large-scale deployment to ensure the reliability of the cryptographic mechanism before proceeding to more extensive system evaluations.

3.2.1 General Description of C++ Module

In this study, the encryption process is executed using an executable file named he_bfv, which is compiled from a C++ program. The smallest polynomial degree, namely 1024, was selected in order to minimize the resulting ciphertext size. The following example illustrates the execution of the module:he_bfv <keydir> keygen

- he_bfv <keydir> encrypt <cand1..5>
- he_bfv <keydir> decrypt < <b64cipher>
- he_bfv <keydir> calculate [--file=PATH] [--json]
- he_bfv <pubkeydir> calculate_enc [--file=PATH]
- he_bfv <keydir> decrypt_vec < <b64cipher>

Explanation of Functions in the he_bfv Program :

3.2.1.1 he_bfv <keydir> keygen Functions

The he_bfv <keydir> keygen command is the initial function that must be executed prior to the encryption process. This command generates three files — params.seal, public.seal, and secret.seal — which are stored in the directory specified by the <keydir> parameter. The roles of these files are described as follows:

- **params.seal** stores the cryptographic parameters and configuration settings associated with the BFV homomorphic encryption scheme. These parameters define the operational environment of the encryption process**.**
- **public.seal** contains the public key, which is utilized for encrypting data and performing homomorphic computations directly on ciphertexts. These computations enable vote tallying for each candidate without requiring decryption, and the resulting output remains in encrypted form.
- **secret.seal** holds the private key, which is used to decrypt individual candidate identifiers as well as to decrypt the encrypted results produced by the homomorphic computations performed using the public key.

3.2.1.2 he_bfv <keydir> encrypt Funcion

The proposed e-voting system applies encryption to secure voter selections by encoding each candidate as a numerical identifier. Upon vote submission, this value is forwarded to an encryption module based on the BFV homomorphic encryption scheme, which supports direct computation over encrypted values without decryption, thereby maintaining confidentiality during processing [25]. The encryption output is a ciphertext encoded in Base64 format, with an observed size of approximately ±8 KB per vote, considerably larger than plaintext data. While ciphertext files are generated for experimental evaluation and size analysis, they are not used in the system's core storage mechanism. In practice, encrypted votes are stored directly in the database as long strings within the encrypted vote table. The substantial storage overhead is an inherent trade-off of BFV-based homomorphic encryption, balancing storage cost against secure encrypted computation capabilities, as similarly documented in previous large-scale and sensitive data applications [25].

Table 3. ncrypted and Hashed Candidate Data

No	Function Under Test	Result
1	Encryption and Hashing of a Candidate (ID = 1)	:~/BFV$ ./he_bfv keys encrypt 1 XqEQBAECAACrFQAAAAAAACi1L/1gsh+NrADcAwF2m4eEg+ET58Z8M4Et5wEBe14kud7C+Wg0I AACzFfxBCgAAAOao9xd8UvSvDAAAANlKIZ4IAAAA99zEnwsAAAAUC3lMDgAAAEx31Z4UfJczD KezpwBcBZeQE7EgLCQAAAEXLr78il3pMu4Xv2hbg8mRUdrY61fJ8hrgbj4czrzdObU12xgK7G 0Jc+WYq5BgAAAIynifGHoNjXBGrQ8beVOOc6y75xAB/pTJd9RVHtzFAjQNbRIAUAAAAS/hujS 31/cpilTbRSJFD5mlPK6ZqsXW7HcxRhWSVi1pYTHpPGPpl+BRybaGf29pBKWRau5OR+9AyHQR
2	Candidate Encryption for an Individual Voter with a Size Limit of ±8 KB	:~/BFV$ ./he_bfv keys encrypt 1 > pemilih1 seeded zstd size=8477 :~/BFV$ du -sh pemilih1 8.0K pemilih1

3.2.1.3 he_bfv <keydir> decrypt

The decryption function is applied to recover plaintext data from ciphertext generated by the he_bfv <keydir> encrypt operation. This step ensures that the encryption mechanism preserves data integrity and that encrypted values can be accurately restored. Decryption is performed using the command ./he_bfv keys decrypt < pemilih1, where pemilih1 contains the ciphertext produced earlier using ./he_bfv keys encrypt 1 > pemilih1. The program reads the ciphertext from the input file and applies the appropriate cryptographic key to reconstruct the original plaintext. In this test case, the decrypted output is the value 1, confirming the correctness of the encryption–decryption workflow.

Table 4. Decrypted Data of a Single Vote

No	Function Under Test	Result
1	Single Vote Decryption Process	:~/BFV$ ./he_bfv keys decrypt < pemilih1 1

3.2.1.4 he_bfv <keydir> calculate

The BFV calculate operation enables encrypted vote aggregation by performing homomorphic computations on ciphertexts generated through the BFV encryption scheme. This mechanism supports secure vote counting without exposing plaintext data at any stage of the computation.

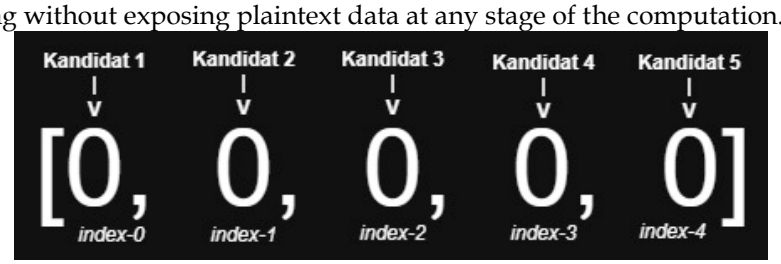


Figure 2. Angka index vector sederhana

Votes are encoded as index-based vectors in which each position represents a candidate. A vote for the first candidate is expressed by assigning a value of one to the zeroth index and zero to all remaining positions, effectively forming a digital ballot representation.

Encrypted vote aggregation is achieved by homomorphically summing these encrypted vectors across all participants. The BFV homomorphic encryption scheme enables this computation to be executed without disclosing any individual selections. The aggregated result remains encrypted and is accessible in plaintext only to authorized parties holding the secret key, thus ensuring both privacy preservation and result integrity.

Table 5. Single-Vote Test Results

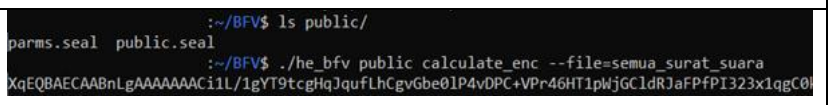
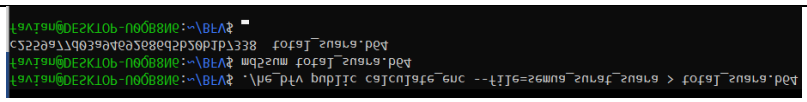
No	Function Under Test	Result
1	Single-Vote Calculation Test Without Decryption (Ciphertext-Based Computation)	:~/BFV$ ./he_bfv keys calculate --file=pemilih1 [1,0,0,0,0]

3.2.1.5 he_bfv <pubkeydir> calculate_enc

The calculate_enc operation computes encrypted vote totals using homomorphic addition over ciphertexts, producing an encrypted aggregate result. Unlike plaintext-based tallying, this process does not require decryption and relies solely on the public key, thereby preventing exposure of individual votes. The encrypted output can be verified and stored before being decrypted only by authorized entities holding the secret key, ensuring both correctness and privacy preservation.[26].

The following table summarizes the experimental validation of the he_bfv <pubkeydir> calculate_enc function:

Table 6. Evaluation Results (calculate_enc)

No	Function Under Test	Result
1	An experimental setup was created to verify that the calculate_enc operation can be performed without access to the private key. A separate directory, denoted as public, was prepared to include only the parameter and public key files. This setup confirms that encrypted vote aggregation can be carried out using only public information.	
2	After execution, the calculate_enc function produces its output in ciphertext form rather than as a plaintext vector, as generated by the he_bfv <keydir> calculate function. This result confirms that vote aggregation is successfully performed directly over encrypted data without requiring any decryption step.	
3	The encrypted output is saved as total_suara.b64 and its integrity is verified using a checksum prior to being decrypted with decrypt_vec for final result validation.	

3.2.1.6 he_bfv <keydir> decrypt_vec

The decrypt_vec operation decrypts the encrypted aggregation result generated by calculate_enc using the secret key, producing a plaintext vector of vote counts. The equivalence between this output and the plaintext result produced by calculate verifies the correctness of encrypted computation under the BFV scheme and confirms its applicability to privacy-sensitive systems such as e-voting. To verify the correctness of the implemented algorithm, the following table summarizes the experimental validation of the he_bfv <keydir> decrypt_vec function.

Table 7. Evaluation Results (decrypt_vec)

No	Function Under Test	Result
1	Decrypting the output of calculate_enc using decrypt_vec yields the same vector as that produced by plaintext computation, thereby confirming the correctness of the homomorphic encryption-based computation.	:~/BFV$:~/BFV$ md5sum total_suara.b64 c2559a77d03a94692686d5b20b1b7338 total_suara.b64 :~/BFV$ ./he_bfv keys decrypt_vec < total_suara.b64 {"counts":[1,4,1,1,2]} :~/BFV$

These C++ modules implement the essential cryptographic operations required for BFV-based encryption, decryption, and encrypted-domain computation.

3.2.2 Large-Scale Testing with 50,000 Votes

As described previously, the PHP application serves as the user interface for vote submission and for inserting voting data into the database. In this project, the he_bfv program is invoked by native PHP through a class named BFVProvider, which acts as an interface between the application layer and the cryptographic computation module.

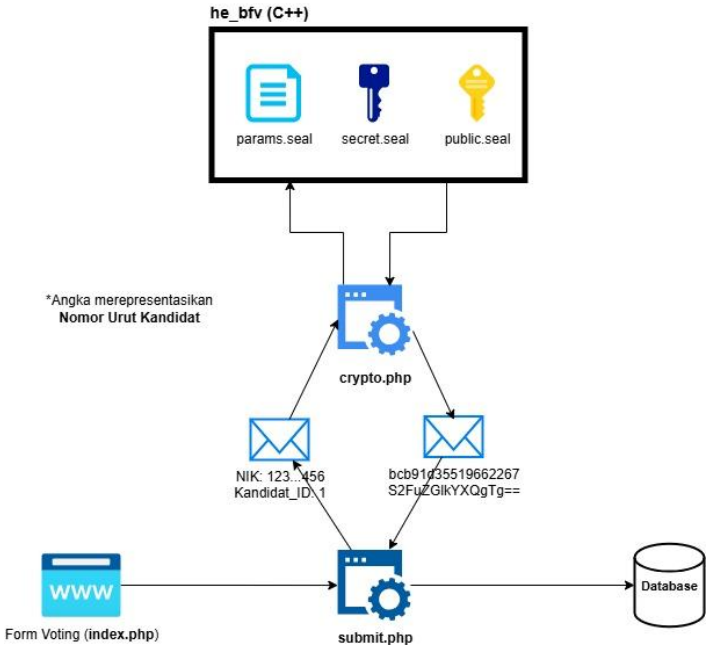


Figure 3. PHP-Side Voting Submission Workflow

1. A web interface for manual candidate selection.

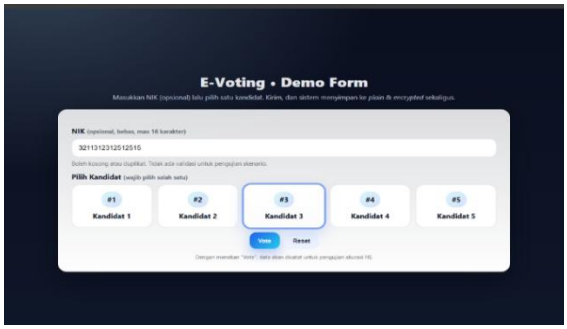


Figure 4. Interface Web PHP untuk melakukan pemilihan secara manual

2. Voting Process with 50,000 Dummy Votes
   50,000 dummy votes were processed through the BFV Linux CLI with parallel input support provided by Python tools, and the results were later verified in the subsequent validation phase.
3. The admin.php interface provides access to the vote totals recorded in both the plaintext (tabel_plain) and encrypted (tabel_encrypted) database tables

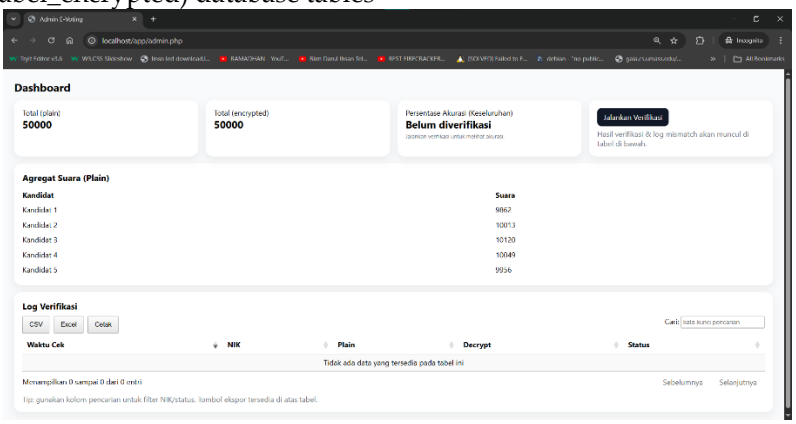


Figure 5. Vote Data Display on the Admin Page

4. Upon completion of the verification stage, a JSON output is produced to report the verification outcome.

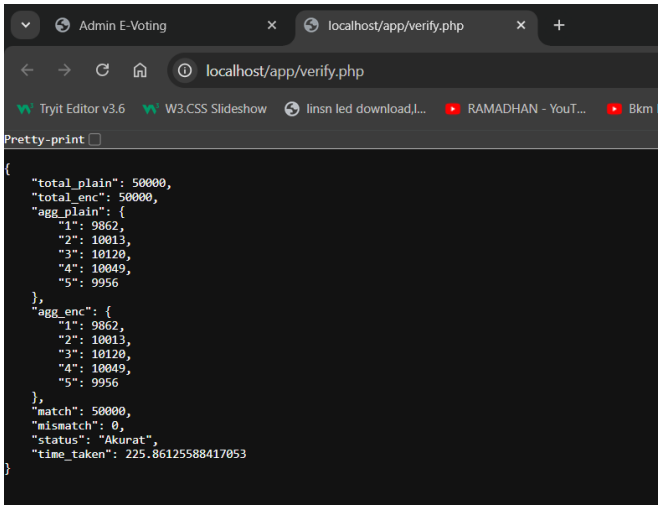


Figure 6. Vote Record Verification Process

The experimental results demonstrate that the system requires approximately 225.861 seconds to handle 50,000 votes, including data insertion into both plaintext and encrypted storage tables

5. An accuracy evaluation page demonstrating 100% correctness for both non-encrypted and homomorphic voting.

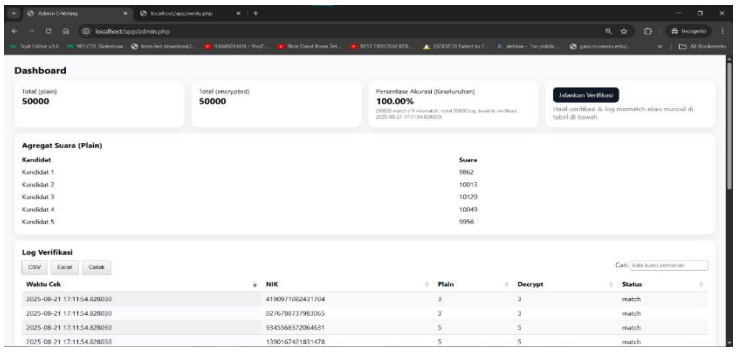


Figure 7. Vote Accuracy Information on the Admin Page

The experimental findings indicate that the vote input mechanism maintains perfect accuracy (100.00%) across 50,000 submitted votes.

6. Encrypted Data Storage View

The figure below demonstrates that both voter identifiers (NIK) and selected candidates are stored in encrypted form.

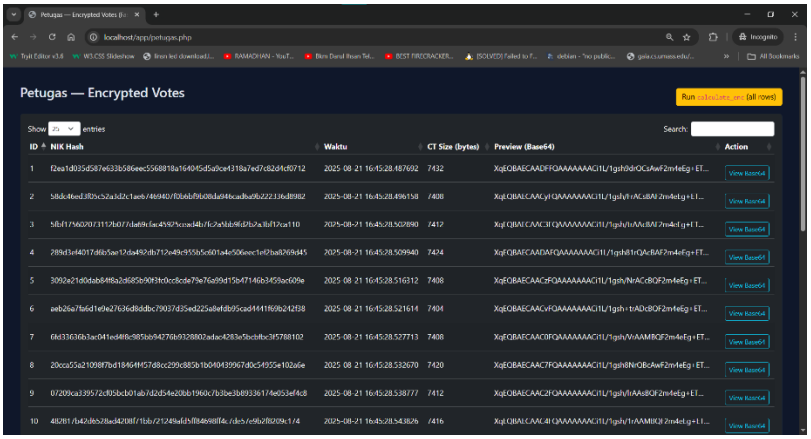


Figure 8. Hashed and Encrypted Votes and NIK Display on the Officer Page

7. Encrypted Vote Tallying Process Without Decryption

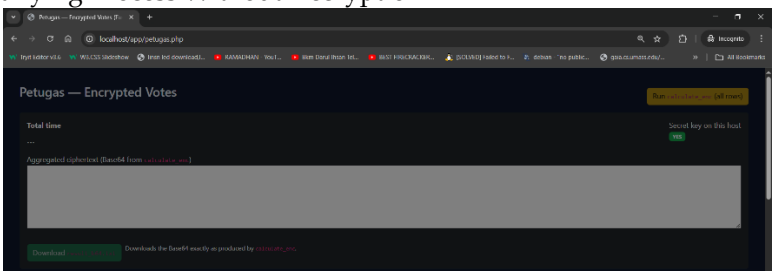


Figure 9. Direct Ciphertext-Based Vote Calculation Without Decryption

The experimental results indicate that encrypted-domain vote aggregation is completed in approximately 13.816 seconds.

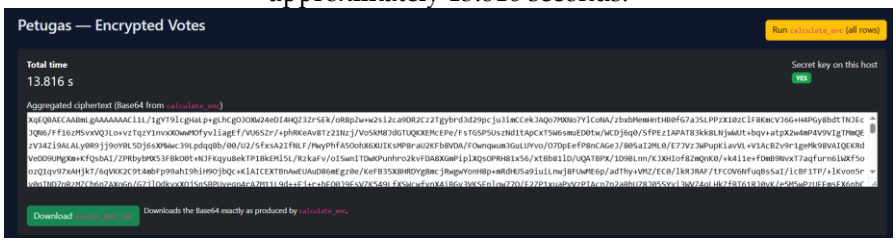


Figure 10. The aggregation of all election results (50,000 votes) requires only 13.816 seconds.

8. Decryption Results of Ciphertext

The following figure shows the decryption-based vote counting procedure, resulting in the cumulative vote totals for all participating candidates.

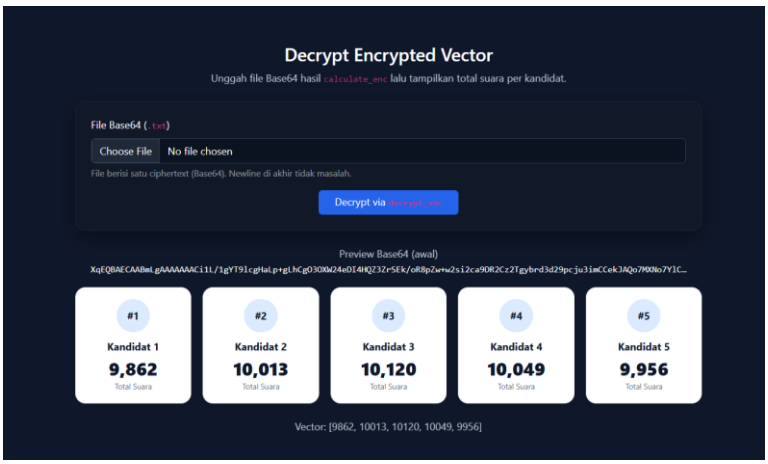


Figure 11. Vote tally results for each candidate obtained by decrypting the ciphertext output of the BFV homomorphic computation.

3.3 Analysis and Evaluation of Test Data

3.3.1 Evaluation of Accuracy Test Results

The verification accuracy results are presented as follows:

- Analysis of Verification Test Results for Each Vote Record Using the Decryption Method

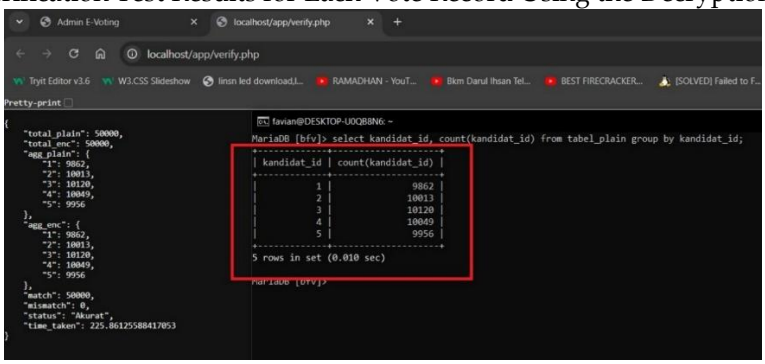


Figure 12. Analysis of the test results shows that the vote tally for each candidate is accurate using the decryption method.

This test aims to compare votes recorded in the ciphertext table (using BFV homomorphic encryption) with votes entered directly into the plaintext table (without encryption). The evaluation was successful, as the verification results matched the data stored in the database

- Analysis of Verification Test Results for Vote Counting Using the Ciphertext Method
The test was successful, as the decrypted results of the encrypted vote counts matched the database records, achieving 100% accuracy.

Figure 13. The analysis results demonstrate that the vote aggregation performed directly on ciphertext is accurate.

3.3.2. Evaluation of Performance Test Results

The test was successful according to the hypothesis, which aimed to complete the input of 50,000 records within 10 minutes. As shown in the figure above, it took 5 minutes and 15 seconds to input all 50,000 records, corresponding to an input rate of 158.5 records per second. The table storing these 50,000 entries has the following size

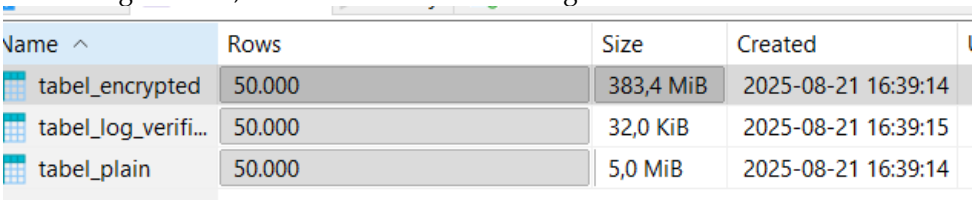


Figure 15. Data Usage Comparison Between Homomorphic Encrypted and Plaintext Tables

The tabel_encrypted is larger in size because it stores BFV ciphertexts, each approximately 8,000 bytes. Multiplying the size of a single ciphertext by 50,000 records and converting to megabytes results in an estimated storage requirement of approximately 381.469 MB.

The encrypted vote calculation for 50,000 votes was completed in under one minute, indicating that the test met the expected performance target.

Table 8. Summary of Experimental Results Analysis

No	Parameter	Test Description	Target / Hypothesis	Result	Conclusion
1	Voting Time (including parameters)	Input 50,000 votes into candidate data randomly	≤ 10 minutes	5 minutes	Achieved
2	Vote Accuracy (Decryption Method)	Compare total votes per candidate between plaintext and ciphertext (Homomorphic) using decryption	100% Accurate	100% Accurate	Achieved
3	Vote Accuracy (Ciphertext / Without Decryption)	Compare total votes per candidate between plaintext and ciphertext without decryption	100% Accurate	100% Accurate	Achieved
4	Data Size	Storage requirement for 50,000 votes meets expected target	512 MB	383.4 MB	Achieved
5	Verification Time (Encrypted Data)	Verify validity of 50,000 encrypted votes	≤ 10 minutes	225.8 seconds	Achieved
6	Decryption Time (Ciphertext / Without Decryption)	Decrypt 50,000 votes using ciphertext without encryption	≤ 10 minutes	5 minutes 15 seconds	Achieved
7	Decryption Time (Decryption Method)	Decrypt 50,000 votes using decryption method	≤ 10 minutes	13.816 seconds	Achieved

3.3.3. Evaluation Using Blockchain-Based Algorithms in Prior Research

From a business-process perspective, the blockchain-based algorithm presented in this study aligns with prior research utilizing the BFV homomorphic encryption scheme, as both approaches ensure that data are encrypted from the initial stage and processed without ever being revealed in plaintext formAccording to the study by Raj, Peker, and Mutlu [30], data "are encrypted before they are shared and remain encrypted… in transit, at rest, and in use", thereby preserving data confidentiality throughout the entire processing lifecycle. The key difference lies in the computational execution model. According to Raj, Peker, and Mutlu, [30], statistical computations are executed directly on the blockchain through smart

contracts, whereby both algorithm integrity and computation results are ensured by the blockchain's immutability and are thus highly resistant to manipulation. In contrast, the BFV-based approach employed in this study generally performs computations off-chain, making integrity more dependent on trust in the executing system. Furthermore, Raj, Peker, and Mutlu, [30] emphasize that implementing FHE on blockchain platforms still faces technical limitations, including the lack of support for division operations on encrypted values and high computational and gas costs, which reduce efficiency for large-scale datasets. Consequently, although both approaches share similar business workflows and security objectives, they differ in the trade-offs between blockchain-based decentralized integrity and the computational efficiency and flexibility offered by the BFV scheme. Further research is required to evaluate the applicability of blockchain-based algorithms for future e-voting systems.

4. Conclusion

This study provides a scientific contribution by demonstrating the feasibility of applying the BFV (Brakerski–Fan–Vercauteren) homomorphic encryption scheme to a web-based e-voting system through a structured evaluation of predefined testing parameters. Experimental results indicate that voting and vote-processing mechanisms operate within the intended operational limits, showing that homomorphic encryption can be integrated into existing e-voting workflows without disruption, even at the scale of tens of thousands of votes. These findings confirm the practical suitability of the BFV scheme for real-world e-voting systems. From the perspective of vote-counting reliability, the results show consistent outputs between decrypted processing and direct computation over ciphertexts, providing empirical evidence that vote tallying can be performed entirely in the encrypted domain without compromising accuracy while minimizing exposure of sensitive data. This supports the adoption of end-to-end security principles in electronic voting systems. Furthermore, the evaluation of encrypted data size and cryptographic processing time demonstrates that resource requirements remain within acceptable system limits. Overall, this study confirms that the BFV algorithm effectively preserves vote confidentiality and integrity and can be realistically deployed in medium-scale e-voting systems. The primary contribution lies in providing measurable evaluation results that may serve as a reference for future research and development of secure and trustworthy e-voting solutions.

References

[1] Y. Zhan, W. Zhao, C. Zhu, Z. Zhao, N. Yang, and B. Wang, "Efficient Electronic Voting System Based on Homomorphic Encryption," pp. 1–19, 2024.
[2] S. Hu, R. Huang, and L. Zhou, "An efficient multi - key BFV fully homomorphic encryption scheme with optimized relinearization," *Cybersecurity*, 2025, doi: 10.1186/s42400-024-00337-2.
[3] K. Yuan, P. Sang, S. Zhang, X. Chen, and W. Yang, "An electronic voting scheme based on homomorphic encryption and decentralization," pp. 1–21, 2023, doi: 10.7717/peerj-cs.1649.
[4] T. Kraavi and J. Willemson, "Proving vote correctness in the IVXV internet voting system," pp. 1–14, 2025.
[5] B. U. Umar, O. M. Olaniyi, and D. O. Olajide, "Paillier Cryptosystem Based ChainNode for Secure Electronic Voting," vol. 5, no. June, pp. 1–11, 2022, doi: 10.3389/fbloc.2022.927013.
[6] P. Manzanares-lopez and M. Cano, "applied sciences Empirical Study of Fully Homomorphic Encryption Using," 2024.
[7] T. Kuo and J. Wu, "A High Throughput BFV-Encryption-Based Secure Comparison Protocol," pp. 1–28, 2023.
[8] A. Al Badawi, Y. Polyakov, K. Mi, M. Aung, B. Veeravalli, and K. Rohloff, "Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme," 2018.
[9] H. Chen, K. Laine, and R. Player, "Simple Encrypted Arithmetic Library - SEAL," 2013.
[10] A. Pedrouzo-ulloa, J. R. Troncoso-pastoriza, N. Gama, M. Georgieva, and F. Pérez-González, "Revisiting Multivariate Ring Learning with Errors and Its Applications on Lattice-Based Cryptography," pp. 1–42, 2021.
[11] A. Aggarwal, Y. Li, and S. Swain, "Improved Noise Bound in BFV Homomorphic Encryption and Its Application to Multiplication," pp. 1–14.
[12] I. Chillotti, N. Gama, M. Georgieva, and M. Izabach, "An homomorphic LWE based E-voting Scheme," pp. 1–21.
[13] L. Jiang and L. Ju, "FHEBench : Benchmarking Fully Homomorphic Encryption Schemes," pp. 2–5.
[14] C. Ayduman and E. Savas, "Homomorphic Encryption on GPU," pp. 1–17.
[15] R. Koseki, A. Ito, R. Ueno, M. Tibouchi, and N. Homma, "Homomorphic encryption for stochastic computing," *J. Cryptogr. Eng.*, vol. 13, no. 2, pp. 251–263, 2023, doi: 10.1007/s13389-022-00299-6.
[16] T. Rahman, "Benchmarking Fully Homomorphic Encryption Libraries in IoT Devices," pp. 16–23, doi: 10.1145/3704522.3704546.
[17] D. S. Pambudi, B. A. Pratomo, and D. Purwitasari, "Performance Analysis of Leading Homomorphic Encryption Libraries : A Benchmark Study of SEAL , HElib , OpenFHE , and Lattigo," pp. 25–30, doi: 10.1145/3729706.3729711.

[18] H. Khatusuriya and M. Parmar, "Benchmarking Homomorphic Encryption on Low- Power Devices : Trade-offs Between PHE and FHE," 2025.
[19] K. D. More, D. Pramod, and R. A. Patil, "INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Homomorphic Encryption with SEAL : Investigating Security and Performance," vol. 12, no. 3, pp. 2174–2181, 2024.
[20] N. B. Njungle and M. A. Kinsy, "A Safety-Centric Analysis and Benchmarks of Modern Open-Source Homomorphic Encryption Libraries," no. Secrypt, pp. 978–989, 2025, doi: 10.5220/0013626400003979.
[21] H. Encryption, "Federated Learning: An approach with Hybrid Homomorphic Encryption," pp. 1–19.
[22] R. Ko, "The Beginner ' s Textbook for Fully Homomorphic Encryption".
[23] S. Min, "Ciphertext-Simulatable HE from BFV with Randomized Evaluation".
[24] J. López, "An Overview on Homomorphic Encryption Algorithms OF An Overview on Homomorphic Encryption Algorithms," 2018.
[25] C. X. Gao, "Privacy-Preserving Railway Data Sharing : A Comparative Study of Homomorphic Encryption Schemes," vol. 24, pp. 329–337, 2024.
[26] Y. Kho and S. Heng, "SS symmetry A Review of Cryptographic Electronic Voting," pp. 1–33, 2022.
[27] F. Wibawa, F. O. Catak, S. Sarp, and M. Kuzlu, "BFV-Based Homomorphic Encryption for Privacy-Preserving CNN Models," pp. 1–14, 2022.
[28] "Introduction to the BFV FHE Scheme," pp. 1–11.
[29] B. Mennink, "On the Collision and Preimage Security of MDC-4 in the Ideal Cipher Model," vol. 2, 2007.
[30] R. Raj, "Blockchain and Homomorphic Encryption for Data Security and," 2024.