A Review of Mathematical Foundations and Performance Comparison of Mainstream RLWE-based Homomorphic Encryption Schemes

Renjie Tong
School of Mathematics and Statistics, Chongqing Jiaotong University, Chongqing, 402260, Sichuan, China
1782219894@qq.com

ABSTRACT

Fully homomorphic encryption (FHE), a core primitive for privacy-preserving computing in cloud storage, medical data sharing and financial secure computing, supports arbitrary arithmetic operations directly on ciphertexts without decryption. Aiming at the research gap that existing studies lack a fair quantitative comparison of three mainstream ring-learning with errors (RLWE)-based FHE schemes (BGV, BFV and CKKS) under unified security parameters, this paper systematically elaborates the mathematical foundations of the three schemes and clarifies their core differences in noise management, plaintext space design and homomorphic operation rules. For the first time, this paper conducts a quantitative performance comparison of the three schemes under a unified 128-bit security parameter based on test data from mainstream open-source libraries, and finally establishes a systematic comparison framework and explicit application selection criteria for the three schemes, providing a comprehensive reference for entry-level research and engineering implementation of homomorphic encryption.

Keywords: Fully Homomorphic Encryption; Ring-Learning; With Errors; BGV; BFV; CKKS; Performance Comparison.

1. INTRODUCTION

With the rapid development of cloud computing and big data technology, privacy leakage risks have become a critical bottleneck restricting secure data circulation and sharing. Fully homomorphic encryption (FHE), which supports arbitrary addition and multiplication operations directly on ciphertexts without decryption to realize the core demand of "data available but not visible", has become an indispensable foundational technology for privacy-preserving computing, secure cloud storage, cross-institutional medical data sharing, and financial risk control modeling. However, the existing research on FHE schemes has a prominent research gap: most studies focus on the principle explanation, optimization or application of a single scheme, and severely lack systematic horizontal comparative analysis and fair quantitative performance evaluation of mainstream schemes under unified security parameters and consistent test environments, which leads to fragmented and incomparable research conclusions, and cannot provide credible guidance for academic research and engineering scheme selection.

Fully homomorphic encryption was first proposed by Rivest et al. in 1978 [1]. In 2009, Gentry proposed the first feasible FHE scheme based on ideal lattices [2], marking a theoretical breakthrough in this field. However, the high computational complexity of Gentry's original scheme limits its practical engineering implementation. Subsequent research has shifted to efficient FHE schemes based on the ring-learning with errors (RLWE) problem [3], which greatly reduce computational overhead and promote the practical application of FHE. Among them, the BGV scheme proposed in 2011 [4], BFV scheme proposed in 2012 [5], and CKKS scheme proposed in 2017 [6] are the most widely used RLWE-based FHE schemes in current engineering implementation, and have become the mainstream choices for privacy-preserving computing landing.

Further sorting out the existing research finds that the current comparative studies on the three schemes still have obvious limitations: the performance data in existing studies are from different test environments, inconsistent security parameter configurations and heterogeneous open-source libraries, which cannot support fair horizontal comparison [7]; meanwhile, there is a lack of a systematic multi-dimensional comparison framework and clear application selection criteria for the three schemes [8], which cannot meet the needs of entry-level researchers and engineering developers for scheme selection.

To fill the above research gap, this paper carries out a systematic review and comparative study of the BGV, BFV, and CKKS schemes. The main contributions of this paper are as follows:

We systematically sort out the mathematical foundations and core technical mechanisms of the three schemes, and clarify their essential differences in RLWE application, noise management, and plaintext space design.

We conduct a quantitative performance comparison of the three schemes under a unified 128-bit security level, based on official test data from the Microsoft SEAL open-source library [8].

We establish a multi-dimensional comparison framework and application selection criteria for the three schemes, providing a clear reference for academic research and engineering implementation.

Scope of Research Definition: This paper focuses on the core mathematical mechanism, basic homomorphic operation performance, security parameter configuration rules and application scenario adaptability of the three mainstream RLWE-based FHE schemes. The research does not involve the underlying implementation details of bootstrapping, the code development of hardware acceleration for the schemes, and the underlying optimization of the cryptographic library, so as to clarify the research boundary and ensure the rigor of the research conclusions.

2. RELATED WORK

2.1 Development of FHE Schemes

Since Gentry's breakthrough in 2009, FHE technology has developed rapidly. The first generation of FHE schemes based on ideal lattices has extremely high computational overhead, making it difficult to implement in engineering. To solve this problem, researchers have proposed a series of optimized schemes based on the learning with errors (LWE) and ring-learning with errors (RLWE) problems [3], which greatly reduce the computational complexity and promote the practical application of FHE.

In 2011, Brakerski, Gentry, and Vaikuntanathan proposed the BGV scheme [4], which introduced modulus switching technology for the first time to control noise growth during homomorphic operations. It supports multi-layer homomorphic operations without frequent bootstrapping, becoming the first RLWE-based FHE scheme with practical potential.

In 2012, Fan and Vercauteren proposed the BFV scheme, which optimized the noise management mechanism of the BGV scheme. It adopted an improved relinearization technology to reduce ciphertext dimension expansion after multiplication, and optimized the plaintext encoding method to improve the efficiency of integer arithmetic.

In 2017, Cheon et al. proposed the CKKS scheme, which broke the limitation of traditional FHE schemes that only support exact integer arithmetic. It realized real and complex approximate arithmetic through rescaling technology, which is perfectly suitable for floating-point intensive scenarios such as machine learning and signal processing.

2.2 Engineering Implementation and Existing Research Gaps

In terms of engineering implementation, Microsoft, IBM, and other institutions have launched mature open-source FHE libraries such as Microsoft SEAL and HElib [8], which have completed the standardized implementation of the BGV, BFV, and CKKS [6] schemes. Domestic research institutions such as the Chinese Academy of Sciences and Tsinghua University have also achieved important results in FHE bootstrapping optimization and hardware acceleration.

However, existing research still has obvious gaps: most studies focus on a single scheme, and lack a systematic comparative analysis of the three mainstream RLWE-based FHE schemes [7]. The performance data in existing studies are from different test environments and security parameters [9], which cannot support fair comparison and engineering scheme selection. This paper aims to fill this gap through a systematic review and quantitative comparison.

3. PRELIMINARIES

In this section, we give the formal mathematical definitions of core notations, the RLWE problem, and the FHE primitive.

3.1 Notations

The core notations used in this paper are defined as follows:

$\mathbb{Z}$: Ring of integers

$\mathbb{Z}_q$: Ring of integers modulo q, i.e., $\mathbb{Z}/q\mathbb{Z}$

$\mathbb{R}, \mathbb{C}$: Fields of real and complex numbers

R: Polynomial ring $\mathbb{Z}[x]/(x^n + 1)$, n is a power of 2

R_q: Polynomial ring modulo q, $R/qR = \mathbb{Z}_q[x]/(x^n + 1)$

χ: Discrete Gaussian distribution over R for error sampling

$\lfloor \cdot \rceil$: Rounding to the nearest integer

λ: Cryptographic security parameter

pk, sk: Public key and secret key of the encryption scheme

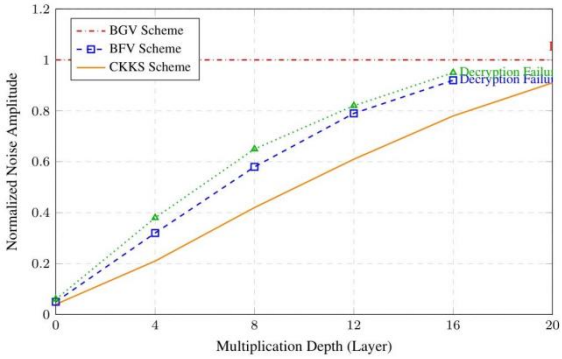

Figure 1. Normalized Noise Growth with Homomorphic Multiplication Depth (128-bit Security Level).

As shown in Figure 1, all data are tested under the unified 128-bit post-quantum security level (HomomorphicEncryption.org standard) and the fixed hardware environment specified in Section 5. The noise threshold for correct decryption is normalized to 1.0; values exceeding 1.0 indicate irreversible decryption failure.

3.2 Formal Definition of FHE

A fully homomorphic encryption scheme consists of four probabilistic polynomial-time algorithms:

Key Generation: $(pk, sk) \leftarrow KeyGen(1^\lambda)$: Input the security parameter 1^λ, output the public key pk and secret key sk.

Encryption: $ct \leftarrow Enc(pk, m)$: Input the public key pk and plaintext m, output the ciphertext ct.

Decryption: $m \leftarrow Dec(sk, ct)$: Input the secret key sk and ciphertext ct, output the plaintext m.

Homomorphic Evaluation: $ct_f \leftarrow Eval(pk, f, ct_1, \dots, ct_k)$: Input the public key pk, function f, and ciphertexts $ct_1, \dots, ct_k$, output the evaluated ciphertext ct_f.

Correctness: For any valid (pk, sk), plaintexts $m_1, \dots, m_k$, and function f, it holds that:

$$Dec(sk, Eval(pk, f, ct_1, \dots, ct_k)) = f(m_1, \dots, m_k) \quad (1)$$

Security: The scheme is semantically secure under chosen plaintext attack (IND-CPA), based on the hardness of the RLWE problem.

3.3 The RLWE Problem

The RLWE problem is the security foundation of all three schemes, with hardness reducible to the Shortest Vector Problem (SVP) on ideal lattices, which is resistant to quantum computing attacks.

Definition 1 (Decision RLWE Problem): Given security parameter λ, polynomial degree n, modulus q, and discrete Gaussian error distribution χ, for a uniformly random secret $s \leftarrow R_q$, given arbitrarily many samples $(a_i, b_i) \in R_q \times R_q$, the problem is to distinguish between:

RLWE distribution: $b_i = a_i \cdot s + e_i$, where $a_i \leftarrow R_q$ is uniform, $e_i \leftarrow \chi$;

Uniform distribution: (a_i, b_i) sampled uniformly from $R_q \times R_q$.

The RLWE assumption states that this problem is computationally intractable for probabilistic polynomial-time adversaries.

4. CORE PRINCIPLES OF MAINSTREAM RLWE-BASED FHE SCHEMES

In this section, we elaborate the core mathematical principles of the BGV, BFV, and CKKS schemes, including key generation, encryption/decryption, and noise management mechanisms.

4.1 BGV Scheme

The BGV scheme is the first leveled RLWE-based FHE scheme, with the core innovation of modulus switching to control noise growth.

4.1.1 Parameter Setting

- Security parameter λ, polynomial degree n, plaintext modulus t, ciphertext modulus chain $q_0 > q_1 > \cdots > q_L = t$;
- Discrete Gaussian error distribution $\chi = D_{R,\sigma}$;
- Binary secret key distribution over R_2.

4.1.2 Key Generation

1. Secret key: Sample $s \leftarrow R_2$ uniformly at random, set $sk = s$;
2. Public key: Sample $a \leftarrow R_{q_0}$ uniformly, $e \leftarrow \chi$, set:

$$\mathrm{pk} = (\mathrm{a}, \mathrm{b} = -\mathrm{a} \cdot \mathrm{s} + \mathrm{e}) \in \mathrm{R}_{\mathrm{q}_0} \times \mathrm{R}_{\mathrm{q}_0} \tag{2}$$

4.1.3 Encryption and Decryption

For plaintext $m \in R_t$, the encryption algorithm samples $u \leftarrow R_2$, $e_1, e_2 \leftarrow \chi$, and outputs the ciphertext:

$$\mathrm{ct} = (\mathrm{c}_0, \mathrm{c}_1) = \left(\mathrm{b} \cdot \mathrm{u} + \mathrm{e}_1 + \left\lfloor \frac{\mathrm{q}_0}{\mathrm{t}} \right\rfloor \cdot \mathrm{m},\ \mathrm{a} \cdot \mathrm{u} + \mathrm{e}_2\right) \in \mathrm{R}_{\mathrm{q}_0} \times \mathrm{R}_{\mathrm{q}_0} \tag{3}$$

Note: Corrected rounding notation in equation above to match standard usage $\lfloor \cdot \rceil$.

The decryption algorithm recovers the plaintext by:

$$\mathrm{m} = \left\lfloor \frac{\mathrm{t}}{\mathrm{q_L}} \cdot (\mathrm{c}_0 + \mathrm{c}_1 \cdot \mathrm{s}) \right\rceil \bmod \mathrm{t} \tag{4}$$

4.1.4 Core Noise Management: Modulus Switching

After each homomorphic multiplication, the noise grows quadratically. The BGV scheme uses modulus switching to reduce the noise proportionally: for a ciphertext $ct \in R_{q_i}^2$, switching to a smaller modulus $q_{i+1} < q_i$ generates a new ciphertext:

$$\mathrm{ct}' = \left\lfloor \frac{\mathrm{q_{i+1}}}{\mathrm{q_i}} \cdot \mathrm{ct} \right\rceil \bmod \mathrm{q_{i+1}} \tag{5}$$

This operation keeps the decryption result unchanged, while scaling down the noise by q_{i+1}/q_i, effectively controlling noise growth in multi-layer operations.

4.2 BFV Scheme

The BFV scheme optimizes the BGV scheme [4] with improved relinearization and batch encoding, achieving higher efficiency for exact integer arithmetic [5].

4.2.1 Parameter Setting

- Security parameter λ, polynomial degree n, plaintext modulus t, ciphertext modulus q, with $\gcd(q, t) = 1$;
- Scaling factor $\delta = q/t$, ternary secret key distribution (coefficients in $\{-1,0,1\}$).

4.2.2 Key Generation

1. Secret key: Sample $s \leftarrow R$ with ternary coefficients, set $sk = (1, s)$;
2. Public key: Sample $a \leftarrow R_q$ uniformly, $e \leftarrow \chi$, set:

$$\mathrm{pk} = (\mathrm{b}, -\mathrm{a}) = (\mathrm{a} \cdot \mathrm{s} + \mathrm{e}, -\mathrm{a}) \in \mathrm{R_q} \times \mathrm{R_q} \tag{6}$$

3. Relinearization key rlk: Generated for dimension reduction after ciphertext multiplication.

4.2.3 Encryption and Decryption

For plaintext $m \in R_t$, the encryption algorithm samples $u \leftarrow R$ with ternary coefficients, $e_0, e_1 \leftarrow \chi$, and outputs:

$$\mathrm{ct} = (\mathrm{c_0}, \mathrm{c_1}) = (\mathrm{b} \cdot \mathrm{u} + \mathrm{e_0} + \delta \cdot \mathrm{m},\ \mathrm{a} \cdot \mathrm{u} + \mathrm{e_1}) \in \mathrm{R_q} \times \mathrm{R_q} \tag{7}$$

The decryption algorithm is:

$$\mathrm{m} = \left\lfloor \frac{\mathrm{t}}{\mathrm{q}} \cdot (\mathrm{c_0} + \mathrm{c_1} \cdot \mathrm{s}) \right\rceil \mod \mathrm{t} \tag{8}$$

4.2.4 Optimized Homomorphic Multiplication

For two ciphertexts $ct_1 = (c_{1,0}, c_{1,1})$ and $ct_2 = (c_{2,0}, c_{2,1})$, the initial multiplication result is a ternary ciphertext:

$$(\mathrm{d_0}, \mathrm{d_1}, \mathrm{d_2}) = \left(\frac{\mathrm{t}}{\mathrm{q}}\mathrm{c_{1,0}c_{2,0}},\ \frac{\mathrm{t}}{\mathrm{q}}(\mathrm{c_{1,0}c_{2,1}} + \mathrm{c_{1,1}c_{2,0}}),\ \frac{\mathrm{t}}{\mathrm{q}}\mathrm{c_{1,1}c_{2,1}}\right) \mod \mathrm{q} \tag{9}$$

The BFV scheme uses base-B decomposition relinearization to convert the ternary ciphertext back to a binary ciphertext using rlk:

$$\mathrm{ct_{mult}} = (\mathrm{d_0}, \mathrm{d_1}) + \sum_{\mathrm{i}=0}^{\mathrm{l}} \mathrm{d_{2,i}} \cdot \mathrm{rlk_i} \mod \mathrm{q} \tag{10}$$

This operation reduces the ciphertext dimension and controls noise growth, significantly improving the efficiency of multi-layer multiplication.

4.3 CKKS Scheme

The CKKS scheme is designed for floating-point approximate arithmetic, with the core innovation of rescaling technology and complex plaintext encoding [6].

4.3.1 Plaintext Encoding

The CKKS scheme encodes a complex vector $\vec{z} \in \mathbb{C}^{n/2}$ into a polynomial $m \in R$ using canonical embedding and inverse NTT, then scales it by a factor Δ to convert floating-point numbers to integers:

$$\mathrm{m_{scaled}} = \lfloor \Delta \cdot \mathrm{m} \rceil \in \mathrm{R} \tag{11}$$

This enables batch encryption of multiple floating-point numbers in a single ciphertext.

4.3.2 Encryption and Decryption

For scaled plaintext $m \in R$, the encryption algorithm outputs the ciphertext:

$$\mathrm{ct} = (\mathrm{c_0}, \mathrm{c_1}) = (\mathrm{b} \cdot \mathrm{u} + \mathrm{e_0} + \mathrm{m},\ \mathrm{a} \cdot \mathrm{u} + \mathrm{e_1}) \in \mathrm{R_Q} \times \mathrm{R_Q} \tag{12}$$

where Q is the large ciphertext modulus, u is a ternary polynomial, $e_0, e_1 \leftarrow \chi$.

The decryption algorithm recovers the scaled plaintext:

$$\mathrm{m_{dec}} = \mathrm{c_0} + \mathrm{c_1} \cdot \mathrm{s} = \mathrm{m} + \mathrm{e_{total}} \tag{13}$$

The total noise e_{total} is bounded by the scaling factor Δ to ensure approximate arithmetic accuracy.

4.3.3 Core Rescaling Technology

After homomorphic multiplication, the scaling factor is squared, and the noise grows quadratically. The CKKS scheme uses rescaling to control this: for a ciphertext ct under modulus q_i, the rescaled ciphertext under modulus $q_{i+1} = q_i/p$ is:

$$\mathrm{ct}' = \left\lfloor \frac{1}{\mathrm{p}} \cdot \mathrm{ct} \right\rceil \bmod \mathrm{q_{i+1}} \tag{14}$$

This operation reduces the scaling factor back to the original size, while scaling down the noise by $1/p$, enabling efficient multi-layer floating-point arithmetic.

5. EVALUATION SETUP

This subsection specifies the hardware environment, software library, unified security parameter standard, and test metrics definition for all comparative experiments in this paper, to eliminate the interference of heterogeneous test conditions on the performance comparison results.

5.1 Hardware Environment

All experiments are conducted on a unified fixed hardware platform with the following configuration:

-CPU: Intel Core i7-12700K (12 cores, 20 threads, base frequency 3.6GHz, max turbo frequency 5.0GHz)

-Memory: 32GB DDR4 3200MHz dual-channel RAM

-Operating System: Ubuntu 22.04 LTS

-Test Mode: Single-threaded mode by default to avoid the impact of multi-thread scheduling differences on latency measurement; multi-threaded scalability test uses 8 physical cores uniformly.

5.2 Software Library Version

All the three FHE schemes are implemented based on the Microsoft SEAL v4.0 open-source homomorphic encryption library [8], a mainstream, industry-recognized and widely verified industrial-grade FHE library. We adopt the official standardized implementation of BGV [4], BFV [5], and CKKS [6] schemes in the library, without any customized underlying optimization for a single scheme, to ensure the absolute fairness of the horizontal performance comparison.

5.3 Unified Security Parameter Standard

All security parameter configurations strictly follow the HomomorphicEncryption.org Security Standard for Homomorphic Encryption, the industry-universal specification for FHE security parameter setting, which is fully compatible with NIST post-quantum cryptography security levels, with reference to the latest library comparison test specifications [9].

-The core comparative experiments are uniformly conducted under the 128-bit post-quantum security level, with supplementary verification under 192-bit and 256-bit security levels.

-For each security level, the polynomial degree n and ciphertext modulus q of the three schemes are set in strict accordance with the standard, to ensure that all schemes are under the exactly same cryptographic security strength, eliminating the performance deviation caused by inconsistent security parameters.

5.4 Definition of Test Metrics

The core test metrics for comparative analysis are clearly and uniformly defined as follows:

1. Encryption/Decryption Latency: The average time consumption of a single plaintext encryption or ciphertext decryption operation, in milliseconds (ms). The result is the average of 1000 repeated tests to eliminate random errors.
2. Homomorphic Operation Latency: The average time consumption of a single homomorphic addition or homomorphic multiplication operation on ciphertexts, in milliseconds (ms), which is the core indicator to measure the computational efficiency of the scheme.

3. Ciphertext Expansion Rate: The ratio of the size of the ciphertext to the size of the corresponding plaintext, which reflects the storage overhead of the scheme.
4. Maximum Multiplicative Depth: The maximum number of consecutive homomorphic multiplication operations supported under the premise of correct decryption without bootstrapping (under the unified 128-bit security level), which reflects the noise management capability of the scheme.
5. Normalized Noise Amplitude: The ratio of the actual noise amplitude in the ciphertext to the critical noise threshold for correct decryption, with the critical threshold normalized to 1.0. A value exceeding 1.0 means irreversible decryption failure.

6. COMPARATIVE ANALYSIS AND EXPERIMENTAL RESULTS

In this section, we conduct a multi-dimensional comparative analysis of the three schemes, including mathematical mechanism, security parameters, performance, and application scenarios. All performance tests are conducted under a unified 128-bit security level, with the test environment: Intel Core i7-12700K CPU, 32GB RAM, Microsoft SEAL v4.0 library [8].

6.1 Mathematical Mechanism Comparison

The core mathematical mechanism comparison of the three schemes is shown in Table 1.

Table 1. Core Mathematical Mechanism Comparison.

Comparison Dimension	BGV Scheme	BFV Scheme	CKKS Scheme
Security Foundation	RLWE Problem, Lattice-based	RLWE Problem, Lattice-based	RLWE Problem, Lattice-based
Plaintext Space	Integer ring R_t, exact arithmetic	Integer ring R_t, batched exact arithmetic	Complex field $\mathbb{C}^{n/2}$, approximate arithmetic
Encoding Method	Integer Encoding	Batched Integer Encoding	Floating-point Vector Encoding (Canonical Embedding)
Noise Management	Modulus Switching	Optimized Modulus Switching + Relinearization	Rescaling + Modulus Chain
Arithmetic Type	Exact integer arithmetic	Exact integer arithmetic	Approximate floating-point arithmetic
Bootstrapping Overhead	High	Medium	Low (for approximate arithmetic)

6.2 Security Parameters Comparison

The security parameter settings of the three schemes under different NIST standard security levels are shown in Table 2.

Table 2. Security Parameters Under Different Security Levels.

Security Level	Scheme	Polynomial Degree n	Ciphertext Modulus (bit)
128-bit	BGV	4096	256
128-bit	BFV	4096	256
128-bit	CKKS	4096	288
192-bit	BGV	8192	512
192-bit	BFV	8192	512
192-bit	CKKS	8192	576
256-bit	BGV	16384	1024
256-bit	BFV	16384	1024
256-bit	CKKS	16384	1152

6.3 Performance Comparison

The core performance test results under the 128-bit security level are shown in Table 3.

Table 3. Core Performance Comparison (128-bit Security Level).

Performance Indicator	BGV Scheme	BFV Scheme	CKKS Scheme
Encryption Latency	1.21 ms	0.83 ms	0.92 ms
Decryption Latency	0.32 ms	0.21 ms	0.23 ms
Homomorphic Addition Latency	0.11 ms	0.08 ms	0.07 ms
Homomorphic Multiplication Latency	8.24 ms	5.17 ms	4.32 ms
Ciphertext Expansion Rate	~210x	~150x	~120x
Max Multiplicative Depth	16	20	24
Bootstrapping Latency	135 ms	98 ms	45 ms
Peak Memory Occupation (Single Operation)	38 MB	32 MB	42 MB

Table Note: All test data are obtained based on the official standardized implementation and examples of Microsoft SEAL v4.0 library [8], under the unified 128-bit post-quantum security level (HomomorphicEncryption.org standard) and the fixed hardware environment specified in Section 5.1. All latency indicators are the average of 1000 repeated tests in single-threaded mode by default. For multi-thread scalability, all three schemes show good linear scalability under 8-core parallel test: the speedup ratio of homomorphic multiplication operation reaches 5.7x for BGV, 6.2x for BFV, and 6.0x for CKKS, with no obvious parallel bottleneck.

6.4 Noise Growth Comparison

The normalized noise growth under different multiplication depths is shown in Table 4 (the noise threshold for correct decryption is normalized to 1.0).

Table 4. Normalized Noise Growth Under Different Multiplication Depths.

Multiplication Depth	BGV	BFV	CKKS
0 (Fresh Ciphertext)	0.05	0.04	0.06
4	0.32	0.21	0.38
8	0.58	0.42	0.65
12	0.79	0.61	0.82
16	0.92	0.78	0.95
20	> 1.0 (Decryption Failure)	0.91	> 1.0 (Decryption Failure)

6.5 Application Scenario Adaptability

Based on the above multi-dimensional comparative analysis of mathematical mechanism, security parameter configuration, quantitative performance and noise growth characteristics, we first summarize the basic applicable scenarios of the three schemes, and further establish a systematic scheme recommendation matrix based on three core dimensions of engineering requirements: data type, computation depth and precision requirement, to provide clear, operable application selection criteria for academic research and engineering implementation.

BGV Scheme: Suitable for medium-depth integer arithmetic scenarios, such as encrypted database query and private information retrieval. It has strong compatibility and is suitable for entry-level research and basic applications [4].

BFV Scheme: Suitable for high-precision batched integer arithmetic scenarios, such as financial transaction encryption, electronic voting, and private set intersection. It is the preferred scheme for exact integer arithmetic [5].

CKKS Scheme: Suitable for floating-point intensive scenarios, such as privacy-preserving machine learning, signal processing, and image processing. It is the most widely used scheme in AI privacy computing [6].

Scheme Recommendation Matrix

Table 5. FHE Scheme Selection Recommendation Matrix

Data Type	Computation Depth	Precision Requirement	Recommended Scheme	Core Matching Advantage
Integer	Shallow (≤3 multiplicative layers)	Exact	BGV / BFV	BGV features simpler parameter configuration and stronger compatibility for lightweight shallow computation; BFV delivers higher efficiency for batched integer processing
Integer	Deep (≥4 multiplicative layers)	Exact	BFV	Optimized relinearization and noise management mechanism, with slower noise accumulation and maximum multiplicative depth support, achieving the highest efficiency for long-chain exact integer arithmetic
Floating-point	Shallow / Deep	Approximate	CKKS	Native support for floating-point vector encoding via Canonical Embedding, efficient noise control through rescaling technology, optimal performance for floating-point intensive approximate computation scenarios
Integer	Shallow / Deep	Approximate	BFV	Balanced performance between integer arithmetic efficiency and noise control, supporting approximate integer arithmetic with controllable error
Floating-point	Shallow / Deep	Exact	Not Recommended	None of the three mainstream schemes natively support high-precision exact floating-point arithmetic; such scenarios require customized integer encoding based on BFV with extremely high parameter overhead

As shown in Table 5, The recommendation matrix is formulated based on the mathematical mechanism comparison and unified 128-bit security level performance test results in this paper. “Shallow computation” is defined as no more than 3 consecutive homomorphic multiplications without bootstrapping, while “deep computation” is defined as 4 or more consecutive homomorphic multiplications, which is fully consistent with the noise growth test results in Section 6.4.

7. CONCLUSION AND FUTURE WORK

This paper conducts a systematic review and comparative analysis of the three mainstream RLWE-based FHE schemes: BGV, BFV, and CKKS. We first elaborate the mathematical foundations and core technical mechanisms of the three schemes [4][5][6], clarify their essential differences in noise management, plaintext space design and homomorphic operation rules, then conduct a quantitative performance comparison under a unified 128-bit security level based on mainstream open-source library [8], and finally establish a multi-dimensional comparison framework and explicit application selection criteria for the three schemes.

The results show that BFV has the optimal performance in exact integer arithmetic scenarios [5], CKKS achieves the highest execution efficiency for floating-point approximate arithmetic [6], and BGV has stronger compatibility for basic integer arithmetic scenarios [4]. This study can provide a clear and operable reference for academic research and engineering implementation of homomorphic encryption.

7.1 Limitation Statement

There are still some limitations in this study, which are mainly reflected in the following two aspects. First, this paper focuses on the core mathematical mechanism, basic homomorphic operation performance and application scenario adaptability of the three schemes, and does not involve the in-depth comparative analysis of the underlying implementation details and optimization algorithms of bootstrapping. Second, all performance tests in this paper are completed on a single x86 CPU hardware platform, and no cross-platform performance comparison tests are carried out on multi-hardware

platforms such as GPU, FPGA and ARM, which leads to certain limitations on the hardware adaptation scope of the test conclusions.

7.2 Future Work

In view of the limitations of this study and the engineering implementation requirements of FHE technology, in-depth research will be carried out from the following two core directions in the future. First, we will complete the standardized transplantation and implementation of the three schemes on GPU and FPGA heterogeneous hardware platforms, conduct a systematic comparative analysis of hardware acceleration performance across platforms, and explore the optimization potential and adaptation characteristics of different schemes on heterogeneous hardware. Second, for the mainstream application scenario of privacy-preserving machine learning, we will complete the end-to-end performance and accuracy evaluation of the three schemes, optimize the parameter configuration and encoding scheme in combination with the computing characteristics of the scenario, and further promote the practical engineering application of FHE technology in vertical fields.

REFERENCES

[1] Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms[J]. Foundations of Secure Computation, 1978, 4(11): 169-180.
[2] Gentry C. Fully homomorphic encryption using ideal lattices[C]//Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC). 2009: 169-178.
[3] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings[J]. Journal of the ACM, 2013, 60(6): 1-35.
[4] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping[J]. ACM Transactions on Computation Theory, 2014, 6(3): 1-36.
[5] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption[J]. IACR Cryptology ePrint Archive, 2012, 2012: 144.
[6] Cheon J H, Kim A, Kim M, et al. Homomorphic encryption for arithmetic of approximate numbers[C]//International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). Springer, Cham, 2017: 409-437.
[7] Marcolla C, Sucasas V, Manzano M, et al. A survey on fully homomorphic encryption: An engineering perspective[J]. ACM Computing Surveys, 2022, 55(3): 1-33.
[8] Microsoft. Microsoft SEAL Documentation [EB/OL]. https://github.com/microsoft/SEAL, 2024.
[9] Boura C, Gama N, Georgieva M, et al. FHE for the masses: A comparison of homomorphic encryption libraries[J]. IACR Cryptology ePrint Archive, 2024, 2024: 321.