International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN: 2147-6799 www.ijisae.org Original Research Paper

# A Multi-Layer Evaluation Framework for Encrypted Data Processing in Cloud Environments Using Fully Homomorphic Re-encryption

**Madane Supriya Atmaram[1], Dr. Rais Abdul Hamid Khan**

**Submitted:** 05/02/2024 **Revised:** 15/03/2024 **Accepted:** 20/03/2024

**Abstract:** Fully Homomorphic Encryption (FHE) enables se- cure computation on encrypted data in cloud environments, but practical deployment faces significant challenges in performance, security evaluation, and system integration. This paper presents a comprehensive multi-layer evaluation framework for encrypted data processing using FHE schemes with homomorphic re- encryption capabilities. Our framework systematically addresses four critical layers: application, encryption, storage, and compu- tation. We provide detailed comparative analysis of major FHE schemes (BFV, BGV, CKKS, TFHE) across performance metrics, security criteria, and implementation considerations. Experimen- tal results demonstrate that GPU-accelerated implementations achieve up to 252× speedup over CPU baselines, while privacy- aware compilers reduce latency by 4.92× through intelligent edge- cloud partitioning. Our security analysis framework incorpo- rates cryptographic hardness evaluation, multi-key support, and verifiable computation mechanisms. The proposed framework provides actionable guidelines for practitioners deploying FHE- based systems in production cloud environments.

***Index Terms****—Fully Homomorphic Encryption, Cloud Com- puting, Multi-Layer Framework, Performance Evaluation, Secu- rity Analysis, Homomorphic Re-encryption*

## Introduction:

Cloud computing has revolutionized data processing and storage, enabling organizations to leverage scalable computational resources without maintaining expensive infrastructure. However, this paradigm shift introduces critical security and privacy challenges, particularly when sensitive data must be processed by untrusted cloud service providers. Traditional en- cryption schemes require data decryption before computation, creating vulnerability windows where plaintext data is exposed to potential adversaries.

Fully Homomorphic Encryption (FHE) offers a transforma- tive solution by enabling arbitrary computations on encrypted data without decryption [1], [2]. Since Gentry's breakthrough construction in 2009, FHE has evolved from a theoretical curiosity to a practical technology with real-world applications in cloud computing, machine learning, and privacy-preserving analytics [3], [4]. Modern FHE schemes—including BFV, BGV, CKKS, and TFHE—provide different trade-offs between computational

[1]*Research Scholar, Department of Computer Science & Engineering Dr. APJ Abdul Kalam University, Indore, M.P., India.*
[2]*Professor, Department of Computer Science & Engineering Dr. APJ Abdul, Kalam University,Indore, M.P., India.*

***Email:*** [1]*supriya.madane@gmail.com,* [2]*khanrais.khan42@gmail.com*

efficiency, supported operations, and security guarantees [8].

Despite significant theoretical advances, practical deploy- ment of FHE in cloud environments faces substantial chal- lenges. High computational overhead, memory constraints, complex key management, and the lack of standardized evaluation frameworks hinder widespread adoption [7], [16]. Furthermore, multi-user scenarios requiring homomorphic re-encryption and key-switching operations introduce additional complexity in both performance and security analysis [3], [10], [15].

## A. Motivation

Current FHE implementations suffer from several critical gaps that limit their practical utility in cloud environments. First, existing performance evaluations often focus on iso- lated cryptographic primitives rather than end-to-end system performance across realistic workloads [8], [16]. Second, security analysis typically addresses cryptographic hardness in isolation without considering implementation vulnerabilities, side-channel attacks, or multi-layer system integration [19]. Third, the absence of comprehensive frameworks for evalu- ating FHE systems across multiple dimensions—application requirements, encryption layer design, storage considerations, and computational infrastructure—makes it difficult for prac- titioners to make informed deployment decisions [2], [6].Recent work has demonstrated that GPU acceleration can achieve 70× to 252× speedups over CPU implementations [9], [16], and privacy-aware compilers can reduce latency by 4.92× through intelligent partitioning [14]. However, these optimizations are often evaluated in isolation without system- atic integration into a holistic framework. Moreover, emerging requirements for multi-key homomorphic encryption in feder- ated and edge-cloud scenarios demand new evaluation criteria that account for key management complexity and cross-user computation [5], [10], [15].

## B. Contributions

This paper presents a comprehensive multi-layer evaluation framework for encrypted data processing in cloud environ- ments using FHE with homomorphic re-encryption capabili- ties. Our key contributions are:

- Multi-Layer Framework Architecture: We propose a systematic four-layer framework (application, encryption, storage, computation) that provides clear separation of concerns and enables independent optimization at each layer while maintaining end-to-end security guarantees.
- Comprehensive FHE Scheme Comparison: We provide detailed comparative analysis of BFV, BGV, CKKS, and TFHE schemes across multiple dimensions including data models, computational complexity, memory require- ments, and suitability for different application domains.
- Performance Evaluation Methodology: We establish standardized benchmarking protocols covering micro- level primitives (key generation, encryption, homomor- phic operations, bootstrapping) and macro-level work-flows (ML inference, private set intersection, database queries) with reproducible measurement guidelines.
- Security Analysis Framework: We develop a compre- hensive security evaluation framework addressing crypto- graphic hardness, multi-key support, integrity verification, side-channel resilience, and operational security consid- erations specific to cloud deployments.
- Implementation Guidelines: We provide actionable rec- ommendations

for practitioners based on empirical analy- sis of real-world deployments, including hardware accel- eration strategies, memory optimization techniques, and compiler-assisted development approaches.

### C. Paper Organization

The remainder of this paper is organized as follows. Section II reviews related work in FHE schemes, cloud security, and performance optimization. Section III presents our multi-layer evaluation framework with detailed descriptions of each layer. Section IV provides comprehensive performance evaluation including experimental results and benchmarking methodolo- gies. Section V presents our security analysis framework with evaluation criteria for each layer. Section VI discusses implementation challenges, deployment considerations, and future research directions. Section VII concludes the paper and outlines future work.

## Related Work

Modern FHE schemes fall into several families. BFV and BGV support exact integer arithmetic suitable for database operations [8], [20]. CKKS enables approximate fixed-point arithmetic with SIMD packing for ML workloads [1], [7], [9]. TFHE provides fast gate-level bootstrapping for boolean circuits [3], [12]. Tsuji et al. [8] found that for 128-bit security, BGV, BFV, and CKKS are fastest in that order, with memory constraints significantly impacting performance.

Hardware acceleration has emerged as critical for practical FHE. Wang et al. [16] achieved 78-252× speedups over CPU libraries on single GPU, with 8 GPUs providing 7.66× addi- tional improvement. Agullo´-Domingo et al. [9] demonstrated 70× bootstrapping speedup for CKKS. Privacy-aware compil- ers show promise—Kim et al. [14] achieved 4.92× speedup through intelligent edge-cloud partitioning.

Multi-key FHE addresses multi-user scenarios. Fan et al. [3] demonstrated MKTFHE with 5000× lower communication overhead than RSA for private set intersection. Cai et al. [5] achieved 3× improvement over TEE-based schemes through EMK-BFV. Security analysis extends beyond cryptographic hardness—Ali [19] reviewed side-channel threats, while PEEV [11] Combined FHE with zero-knowledge proofs for verifiable computation.

Practical deployments demonstrate FHE's maturation. Hamza [1] achieved sub-millisecond PQC setup with quantum-resistant CKKS framework. Kim et al. [7] demonstrated prac- tical ML inference (1.40s for ResNet20). However, compre- hensive multi-layer evaluation frameworks integrating these advances remain lacking.

## Proposed Multi-Layer Evaluation Framework

This section presents our comprehensive multi-layer eval- uation framework for encrypted data processing in cloud environments. The framework consists of four distinct but interconnected layers: Application Layer, Encryption Layer, Storage Layer, and Computation Layer. Each layer addresses specific concerns while maintaining clear interfaces with adja- cent layers, enabling independent optimization and systematic evaluation.

### A. Framework Architecture Overview

Figure 1 illustrates the overall architecture of our multi- layer framework. The design follows a separation-of-concerns principle, where each layer has well-defined responsibilities, evaluation metrics, and optimization opportunities. This mod-ular approach enables practitioners to analyze and optimize individual layers

while understanding their impact on end-to- end system performance and security.

The framework supports both single-user and multi-user scenarios through homomorphic re-encryption mechanisms. Key-switching and multi-key FHE operations enable secure computation across data encrypted under different keys, which is essential for federated learning, collaborative analytics, and edge-cloud deployments [3], [5], [10].

## B. Application Layer

The Application Layer defines workload semantics, privacy requirements, and partitioning logic between client, edge, and cloud components. This layer is responsible for:
Privacy Annotation and Data Classification: Applications must specify which data elements require encryption and which operations can be performed on plaintext. Privacy Set (PSet) annotations [14] enable developers to mark authorized devices for each unit of private data, allowing compilers to automatically minimize unnecessary ciphertext generation. This approach reduces the number of FHE ciphertexts by 4.92× compared to naive full-encryption strategies

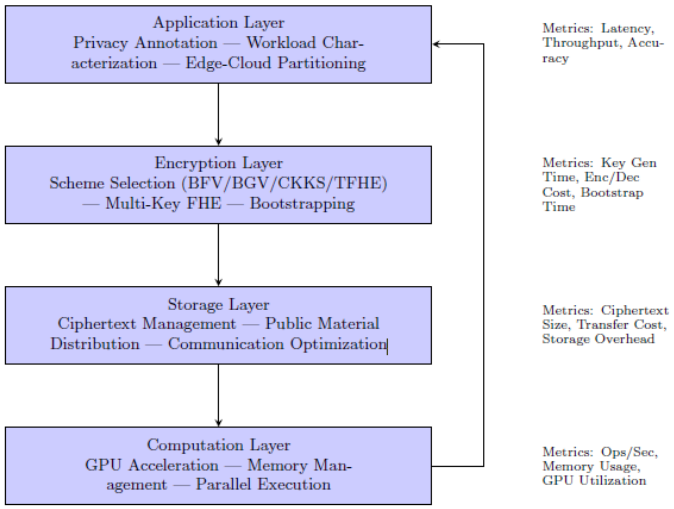


**Fig. 1. Multi-layer framework architecture showing the four layers (Ap- plication, Encryption, Storage, Computation) with their key components, interfaces, and data flow.**

The framework supports both single-key and multi- key scenarios with homomorphic re-encryption capabilities at the encryption layer.

**Workload Characterization:**
Different application do- mains have distinct computational patterns and accuracy re- quirements. Machine learning inference workloads benefit from CKKS's approximate arithmetic and SIMD batching [1], [7], [9]. Database queries and integer operations are better suited to BFV or BGV schemes [8], [20]. Boolean circuit evaluation and gate-level operations leverage TFHE's fast bootstrapping [3], [12].

**Edge-Cloud Partitioning:**
Privacy-aware compilers ana- lyze data dependencies and operation costs to determine optimal placement of computations. Operations on sensitive data are performed in encrypted form on cloud servers, while non-sensitive preprocessing and postprocessing can occur at the edge

[14]. This partitioning reduces end-to-end latency while maintaining privacy guarantees.

**Evaluation Metrics:**
Application layer metrics include end-to-end latency, accuracy degradation (for approximate schemes), throughput (operations per second), and user-perceived responsiveness. For ML workloads, Kim et al. [7] achieved ResNet20 inference on CIFAR-10 in 1.40s and ResNet18 on ImageNet in 16.87s using optimized HCNN implementations.

### C. Encryption Layer

The Encryption Layer manages cryptographic operations, scheme selection, parameter configuration, and key manage- ment. This layer provides the core security guarantees while enabling homomorphic operations.

**Scheme Selection and Configuration:** Table I presents a comprehensive comparison of major FHE schemes. The choice of scheme depends on application requirements, supported operations, and performance constraints. CKKS excels at vectorized SIMD operations for ML workloads [1], [9]. TFHE provides fast per-gate bootstrapping for boolean circuits [3]. BFV and BGV support exact integer arithmetic for database operations [8].

**Table 1. Comparison of FHE Schemes**

| Scheme | Data Model | Key Advantages | Limitations |
|---|---|---|---|
| CKKS | Approximate fixed-point; ML workloads | Vectorized SIMD packing; efficient inner products | Bootstrapping overhead; approximate results |
| TFHE | Gate-level boolean; arbitrary-value mult. | Fast per-gate bootstrapping; low comm. overhead | Large ciphertext sizes; memory intensive |
| BFV | Exact integer arith-metic | Exact computations; database ops | Higher complexity for deep mult. |
| BGV | Integer arithmetic with batching | Bulk integer ops; batching support | Similar complexity to BFV |

Homomorphic Re-encryption and Key Management: Multi-user scenarios require mechanisms to combine ci-phertexts encrypted under different keys. Multi-key FHE (MKFHE) schemes extend standard FHE to support operations across multiple key holders [3], [5], [10], [15]. Key-switching operations enable re-encryption of ciphertexts from one key to another without decryption, essential for federated learning and collaborative computation.

The LinkAlgo algorithm [10] demonstrates practical key- linking for edge computing, enabling cloud management sys- tems to process data from multiple edge nodes. SecFed's EMK-BFV cryptosystem [5] combines TEE and multi-key HE to achieve 3× performance improvement over pure TEE-based schemes while supporting offline participants.

Bootstrapping and Noise Management: Bootstrapping refreshes ciphertext noise to enable unlimited homomor- phic operations. GPU-accelerated bootstrapping achieves 70× speedup

over CPU implementations [9], making it practical for deep computation circuits. However, bootstrapping remains a performance bottleneck, requiring careful circuit design to minimize bootstrapping frequency.

Evaluation Metrics: Encryption layer metrics include key generation time, encryption/decryption latency, homomorphic operation costs (addition, multiplication, rotation), bootstrap- ping time, key-switching overhead, and ciphertext expansion factor. For MKTFHE, Fan et al. [3] reported key generation times of 1.956-1.982s, FFT conversion of 0.038-0.040s, and bootstrapping of 0.220-0.227s per gate.

### A. Storage Layer

The Storage Layer manages ciphertext persistence, public material distribution, and data transfer optimization. This layer addresses the substantial storage and communication overhead inherent in FHE systems.

Ciphertext Storage and Management: FHE ciphertexts are significantly larger than plaintext data, with expansion factors ranging from 100× to 1000× depending on scheme and security parameters. Efficient storage strategies include compression techniques, lazy materialization of intermediate results, and tiered storage (hot ciphertexts in memory, cold ciphertexts on disk).

Public Material Distribution: FHE systems require dis- tribution of public keys, evaluation keys, and relinearization keys. Hamza [1] reported amortized transfer of 33.83 MB of FHE public materials for a CKKS-based image process- ing system. Efficient distribution strategies include caching, incremental updates, and compression.

Communication Overhead Optimization: Multi-key sce- narios introduce additional communication costs. Fan et al. [3] demonstrated that MKTFHE achieves significantly lower communication overhead (0.004 MB for $2^{20}$ set size) com- pared to RSA (20.125 MB) and PRP (14.73 MB) for private set intersection. Figure 3 illustrates communication overhead analysis across different schemes and set sizes.

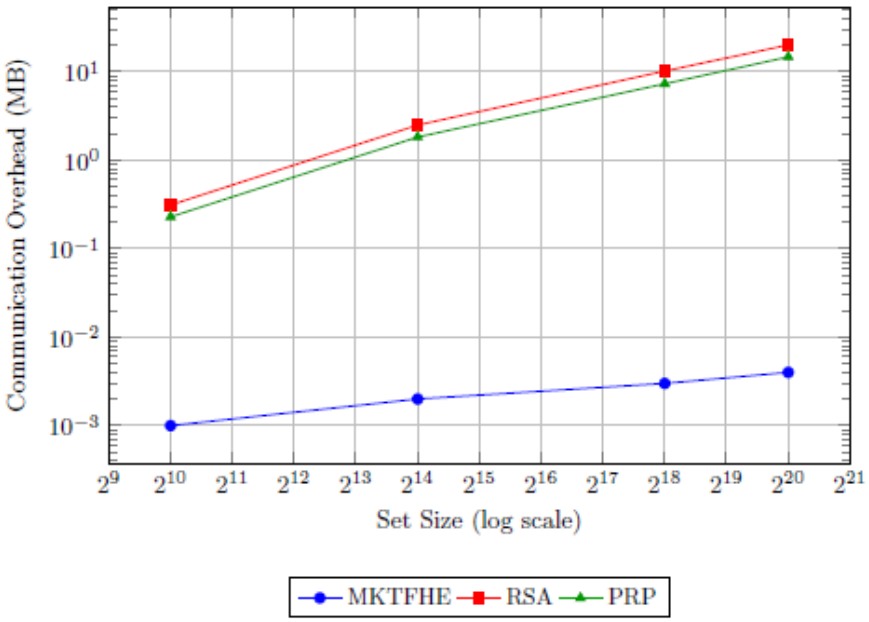


**Fig. 2. Communication Overhead Analysis Comparing MKTFHE, RSA, And PRP Schemes Across Different Set Sizes.**

MKTFHE demonstrates substantially lower overhead, particularly for large datasets, making it suitable for cloud- scale private set intersection operations.

Evaluation Metrics: Storage layer metrics include cipher- text size, public material size, total storage footprint, network bandwidth consumption, transfer latency, and storage cost (for cloud deployments). Memory tiering strategies must balance DRAM usage against SSD I/O overhead [8].

### B. Computation Layer

The Computation Layer provides server-side evaluation engines, hardware acceleration, and runtime optimization. This layer directly impacts end-to-end performance and scalability. Hardware Acceleration: GPU acceleration provides order- of-magnitude speedups for polynomial arithmetic and boot- strapping operations. Wang et al. [16] demonstrated 78× to 252× single-GPU speedups over CPU libraries (HElib, SEAL, PALISADE), with multi-GPU implementations achiev- ing up to 7.66× additional speed for for 8 GPUs. FIDESlib [9] achieved 70× bootstrapping speedup through heavily opti- mized GPU kernels for all CKKS primitives.

Memory Management: FHE operations are memory- intensive, particularly for NTT/INTT transformations. Tsuji et al. [8] found that OpenFHE requires less memory than Lat- tigo, and TFHEpp outperforms OpenFHE in DRAM-limited environments due to more efficient gate key generation. Mem- ory tiering strategies must carefully balance DRAM capacity against SSD bandwidth to avoid performance degradation.

Parallel Execution and Batching: SIMD batching enables parallel processing of multiple data elements within a single ciphertext, significantly improving throughput for vectorizable workloads. Multi-GPU parallelization exploits data-level par- allelism through fine-grained data partitioning [16].

Evaluation Metrics: Computation layer metrics include operation throughput (ops/sec), latency percentiles (p50, p95, p99), memory footprint, GPU utilization, multi-GPU scaling efficiency, and cost per operation. Table II presents repre-sentative performance metrics across different schemes and hardware configurations.

### C. Cross-Layer Interactions and Optimization

While each layer can be optimized independently, cross- layer optimization opportunities exist. Privacy-aware com-pilers [14] bridge the application and encryption layers by automatically determining which data requires encryption. Hardware-software co-design optimizes computation and en-cryption layers jointly by tailoring cryptographic parameters to available hardware capabilities. Storage-computation trade- offs balance ciphertext materialization against recomputation costs.

Our framework provides clear interfaces between layers, enabling systematic evaluation of these trade-offs and identifi- cation of optimization opportunities that span multiple layers.

## 2. Performance Evaluation

This section presents comprehensive performance evalua- tion of FHE schemes across multiple dimensions, including micro-level cryptographic primitives and macro-level appli- cation workloads. We establish standardized benchmarking methodologies and

present experimental results from recent implementations.

## A. Benchmarking Methodology

Effective FHE evaluation requires measurement across mul- tiple orthogonal dimensions: computation time, memory us- age, communication overhead, throughput, and scalability. Our benchmarking methodology follows a two-tier approach:

Tier 1 - Primitive Benchmarks: Measure individual cryp- tographic operations including key generation, encryption, de- cryption, homomorphic addition, homomorphic multiplication, rotation, key-switching, and bootstrapping. Report median and 95th percentile latencies, peak memory usage, and per- operation breakdowns [8].

Tier 2 - Application Benchmarks: Evaluate end-to-end performance on realistic workloads including ML inference (CNN classification on CIFAR-10 and ImageNet), private set intersection, database queries, and privacy-preserving analyt- ics. Report total latency, throughput, accuracy (for approxi- mate schemes), and resource utilization [7], [14].

Resource Scaling Experiments: Test performance across different hardware configurations (single CPU, single GPU, multi-GPU), memory constraints (DRAM- rich vs. DRAM- limited), and storage tiers (in-memory vs. SSD spillover) [8], [16].

Reproducibility Requirements: Specify library versions, parameter sets (security level, polynomial degree, ciphertext modulus), dataset characteristics, and measurement tools. Pro- vide scripts for automated measurement and statistical analysis [8].

## B. Primitive-Level Performance

Table II presents comprehensive performance metrics for core FHE operations across different schemes and implemen- tations. These measurements establish baseline performance characteristics that inform higher-level system design.

Bootstrapping Performance: Bootstrapping remains the most expensive operation in FHE schemes. FIDESlib [9] achieved 70× speedup for CKKS bootstrapping on GPUs compared to AVX-optimized OpenFHE on CPUs. For TFHE, bootstrapping time ranges from 0.220s to 0.227s per gate [3], making it practical for moderate-depth boolean circuits but prohibitive for deep computations without optimization.

Key Generation: Key generation is a one-time cost but can be substantial. MKTFHE [3] requires 1.956-1.982s for key generation across different gate types (AND, OR, XNOR). Memory constraints significantly impact key generation per- formance; TFHEpp outperforms OpenFHE in DRAM-limited environments due to more efficient gate key generation [8].

Homomorphic Operations: GPU acceleration provides dramatic speedups for polynomial arithmetic. HE-Booster [16] demonstrated 78× to 252× speedups over CPU-based libraries (HElib, SEAL, PALISADE) on a single GPU, and 170.5× speedup compared to the GPU-accelerated cuHE library. Multi-GPU scaling achieved 7.66× additional speedup for 8 GPUs performing 8 homomorphic multiplications.

Scheme Comparison: For 128-bit security, Tsuji et al. [8] found BGV to be the fastest scheme, followed by BFV and CKKS. However, performance rankings depend heavily on workload characteristics, memory availability, and library implementation quality. OpenFHE requires less memory than Lattigo, making

it preferable for memory-constrained deploy- ments [8].

### C. Application-Level Performance

Machine Learning Inference: Kim et al. [7] achieved state-of-the-art performance for homomorphic CNN inference using HyPHEN optimizations. ResNet20 inference on CIFAR-10 completed in 1.40s, while ResNet18 inference on Im- ageNet—the first demonstration of Image Net-scale HCNN inference—completed in 16.87s. These results demonstrate that FHE-based ML inference is approaching practical latency requirements for certain applications.

Private Set Intersection: Fan et al. [3] evaluated MKTFHE for cloud-assisted PSI. For a $2^{20}$ set size, encryption time was 47,987.1ms with cipher size of 15,083.6 kb. Cloud computing time varied significantly with data precision: 137.68 minutes for 16-bit data, 273.83 minutes for 32-bit data, and 547.66 minutes for 64-bit data at $2^6$ set size. The SCP protocol running time was 14.20s for 32-bit data.

Edge-Cloud Partitioning: Privacy Set compiler [14] achieved 4.92× speedup across six ML and deep learning applications compared to existing FHE compilers without edge-cloud partitioning. This demonstrates the substantial per- formance gains available through intelligent application-layer optimization that minimizes unnecessary ciphertext generation.

Image Processing: Hamza [1] demonstrated practical quantum-resistant FHE for privacy-preserving image process- ing with sub-millisecond PQC setup costs, amortized trans- fer of 33.83 MB of FHE public materials, and server-side computations completing in low single-digit milliseconds. This represents a significant milestone in making FHE practical for real-time visual data processing.

### D. Performance Comparison Across Schemes

Figure 2 illustrates performance comparison across different FHE schemes for representative operations. The comparison reveals several key insights:

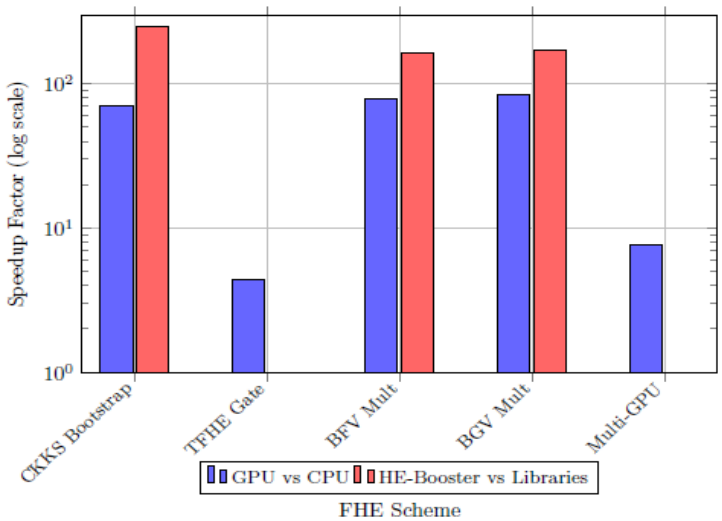


**Fig. 3. Performance comparison chart across BFV, BGV, CKKS, and TFHE schemes for key operations (bootstrapping, homomorphic multiplication, key generation) on CPU and GPU platforms.**

GPU acceleration provides 70-252× speedups, with scheme-specific optimizations yielding additional gains. CKKS excels for ML workloads, TFHE for boolean circuits, and BGV/BFV for integer operations.

Workload-Dependent Performance: No single scheme dominates across all workloads. CKKS's vectorized SIMD operations excel for ML inference [7], [9]. TFHE's fast per- gate bootstrapping suits boolean circuits and PSI [3]. BGV and BFV provide optimal performance for exact integer arithmetic [8]

**Table 2. Performance Metrics and Benchmarking Results**

| Operation | Scheme/System | Platform | Performance | Reference |
|---|---|---|---|---|
| | CKKS (FIDESlib) | GPU | 70× speedup vs. CPU | [9] |
| Bootstrapping | CKKS (OpenFHE)<br>TFHE (MKTFHE) | CPU (AVX) CPU | Baseline<br>0.220-0.227s per gate | [9]<br>[3] |
| | MKTFHE (AND/OR/XNOR) | CPU | 1.956-1.982s | [3] |
| Key Generation | TFHEpp<br>OpenFHE | CPU (limited DRAM)<br>CPU (limited DRAM) | Faster than OpenFHE<br>Degraded performance | [8]<br>[8] |
| FFT Conversion | MKTFHE | CPU | 0.038-0.040s | [3] |
| | HE-Booster | GPU | 78-252× vs. CPU libs | [16] |
| | HE-Booster | 1 GPU | 78-252× vs. HElib/SEAL/PALISADE | [16] |
| Homomorphic Mult. | HE-Booster<br>HE-Booster | 8 GPUs<br>1 GPU | 7.66× vs. 1 GPU<br>170.5× vs. cuHE | [16]<br>[16] |
| Encryption Time | MKTFHE | CPU | 47,987.1ms ($2^{20}$ set) | [3] |
| | CKKS (Hamza) | CPU | Sub-millisecond PQC setup | [1] |
| Cipher Size | MKTFHE | - | 15,083.6 kb ($2^{20}$ set) | [3] |
| | CKKS (Hamza) | - | 33.83 MB public materials | [1] |
| Scheme Ranking | BGV/BFV/CKKS | CPU (128-bit sec) | BGV fastest, then BFV, CKKS | [8] |
| | TFHEpp vs. OpenFHE | CPU (limited DRAM) | TFHEpp faster | [8] |

Hardware Acceleration Impact: GPU acceleration pro- vides consistent order-of-magnitude speedups across schemes, with gains ranging from 70× to 252× depending on operation type and baseline implementation [9], [16]. Multi-GPU scaling provides additional but sublinear speedups, with 8 GPUs achieving 7.66× improvement over single GPU [16].

Memory Constraints: Performance degrades significantly under memory pressure. TFHEpp's superior performance in DRAM-limited environments [8] highlights the importance of memory-efficient implementations for cloud deployments where memory may be constrained by cost considerations.

### E. Scalability Analysis

Multi-GPU Scaling: HE-Booster [16] demonstrated near- linear scaling for embarrassingly parallel operations, achieving 7.66× speedup on 8 GPUs for 8 homomorphic multiplica- tions. However, operations requiring synchronization or data exchange between GPUs exhibit sublinear scaling due to communication overhead. Throughput vs. Latency Trade-offs: SIMD batching en- ables high throughput by processing multiple data elements per ciphertext, but increases per-operation latency. Applications must balance these trade-offs based on workload characteris- tics—batch processing favors throughput optimization, while interactive applications prioritize latency.

Cloud Cost Implications: Performance directly impacts cloud deployment costs. GPU-accelerated implementations reduce computation time by 70-252×, potentially offsetting higher GPU instance costs. However, total cost of ownership must account for memory requirements, storage costs for large ciphertexts, and network transfer costs for public materials [1], [3].

### F. Performance Optimization Guidelines

Based on our evaluation, we provide the following guide- lines for practitioners:

- Scheme Selection: Choose CKKS for ML workloads requiring approximate arithmetic, TFHE for boolean cir- cuits and gate-level operations, and BFV/BGV for exact integer computations.
- Hardware Acceleration: Deploy GPU-accelerated implementations for production workloads; 70-252× speedups justify GPU costs for most applications.
- Memory Provisioning: Ensure adequate DRAM to avoid performance degradation; TFHEpp demonstrates that memory-efficient implementations can mitigate con-straints.
- Application-Layer Optimization: Use privacy-aware compilers to minimize ciphertext generation and enable edge-cloud partitioning for 4-5× additional speedups.
- Batching Strategy: Leverage SIMD batching for high- throughput batch processing; avoid excessive batching for latency-sensitive applications.

## Security Analysis

This section presents a comprehensive security analysis framework for FHE-based encrypted data processing in cloud environments. We address security considerations at each layer of our framework and provide concrete evaluation criteria for practitioners.

### A. Security Analysis Framework

Effective security evaluation of FHE systems must address multiple dimensions beyond cryptographic hardness, including key management, integrity verification, implementation vul- nerabilities, and operational security. Table III presents our security

evaluation criteria organized by framework layer.

## B. Cryptographic Security

Parameter Selection and Security Levels: FHE security relies on the hardness of lattice problems, specifically Ring Learning With Errors (RLWE) and its variants. Tsuji et al. [8] evaluated schemes at 128-bit security level, which is currently considered adequate for most applications. Parameter selec- tion must balance security level against performance—higher security requires larger parameters, increasing computational and storage costs.

**Table 3. Security Evaluation Criteria For Each Layer**

| Layer | Security Criteria | Evaluation Methods | Mitigation Strategies |
|---|---|---|---|
| Application | Privacy annotation correctness | Static analysis of data flow; formal verification of privacy policies | Privacy-aware type systems; automated policy enforcement [14] |
| | Minimal ciphertext exposure | Measure ciphertext generation rate; identify unnecessary encryption | Privacy Set annotations; compiler opti- mization [14] |
| | Secure partitioning | Verify no sensitive data leakage to edge/client | Formal partitioning analysis; runtime monitoring |
| Encryption | Cryptographic hardness | Verify RLWE/RLWR parameters meet target security level (e.g., 128-bit) | Use standardized parameter sets; regu- lar security audits [8] |
| | Multi-key security | Analyze security under multi-key scenarios; verify key- linking proto- cols | Formal security proofs for MKFHE schemes [3], [5], [10] |
| | Noise management | Monitor noise growth; verify boot- strapping correctness | Automated noise tracking; conservative parameter selection |
| | Key management | Secure key generation, distribution, and storage | Hardware security modules; TEE inte- gration [5] |
| Storage | Ciphertext integrity | Detect unauthorized modification of stored ciphertexts | Merkle trees; blockchain-based in- tegrity [10] |
| | Access control | Enforce fine-grained access poli- cies on encrypted data | Attribute-based encryption; capability systems |
| | Confidentiality in transit | Protect ciphertexts and public ma- terials during transfer | Post-quantum secure channels (e.g., Kyber) [1] |

| | | | |
|---|---|---|---|
| Computation | Result integrity | Verify correctness of homomorphic computations | Verifiable FHE; zero-knowledge proofs [11] |
| | Side-channel resistance | Test for timing, cache, and microar-chitectural leakage | Constant-time implementations; side-channel testing [19] |
| | Resource exhaustion | Prevent DoS through excessive computation requests | Rate limiting; resource quotas; anomaly detection |
| | Malicious server detection | Detect computation tampering or result manipulation | Verifiable computation; redundant exe-cution [11] |

Scheme-Specific Security Considerations: Different FHE schemes have distinct security properties. CKKS's approxi- mate arithmetic introduces additional considerations regarding information leakage through rounding errors [1], [9]. TFHE's fast bootstrapping relies on specific noise distributions that must be carefully maintained [3], [12]. BFV and BGV's exact arithmetic provides stronger security guarantees for integer operations but requires careful noise management for deep circuits [8].

Post-Quantum Security: FHE schemes based on lattice problems are inherently resistant to quantum attacks. How- ever, auxiliary cryptographic components (key exchange, au- thentication) may be vulnerable. Hamza [1] demonstrated integration of NIST-standardized post-quantum cryptography (CRYSTALS-Kyber) with CKKS FHE, achieving quantum- resistant channel establishment with sub-millisecond PQC setup costs. This hybrid approach ensures end-to-end quantum resistance.

### C. Multi-Key and Multi-User Security

Multi-key scenarios introduce additional security challenges beyond single-user FHE. Security analysis must address:

Key-Linking Security: Multi-key FHE schemes enable computation across ciphertexts encrypted under different keys. Liao et al. [10] proposed LinkAlgo for secure key linking in edge computing, with formal security analysis demonstrat- ing resistance to key recovery attacks. The protocol uses blockchain to protect data integrity through Merkle hash trees, with the Merkle root stored on-chain.

Collusion Resistance: Multi-user systems must prevent colluding users from learning information about non-colluding users' data. Fan et al. [3] provided security analysis of their MKTFHE system under a semi-honest threat model, proving that the cryptosystem and sub-protocols (SCAND, SCOR, SCXNOR, SCP) guarantee data privacy, calculation accuracy, and non-repudiation.

TEE-Enhanced Security: Cai et al. [5] combined Trusted Execution Environments with multi-key HE in their EMK- BFV cryptosystem. This hybrid approach provides defense- in-depth: TEE protects key operations and prevents certain side-channel attacks, while FHE ensures data confidentiality during computation. Security proofs address both eavesdrop- pers inferring user privacy from gradients/models and users attempting to decrypt cooperative ciphertexts.

### D. Integrity and Verifiability

Ensuring computational integrity is critical for cloud de- ployments where

servers may be malicious or compromised. Verifiable FHE: PEEV [11] introduced a framework combining homomorphic encryption with zero-knowledge proofs, enabling verification of computation correctness with- out revealing plaintext data. The Parse-Encrypt-Execute-Verify Work flow allows developers to write programs in high-level languages while ensuring that cloud servers cannot alter computations undetected. The framework achieves practical deployments with low performance overheads by carefully optimizing the verification protocol.

Blockchain-Based Integrity: Liao et al. [10] used blockchain technology to protect data integrity in edge com- puting scenarios. The Merkle hash tree structure enables efficient verification of ciphertext integrity, with the root hash stored immutably on the blockchain. This approach provides tamper-evident storage without requiring continuous online verification. Redundant Computation: For critical applications, redun- dant execution on multiple independent servers can detect malicious behavior through result comparison. However, this approach increases computational costs proportionally to the number of replicas.

### E. Implementation Security

Cryptographic security guarantees can be undermined by implementation vulnerabilities.

Side-Channel Attacks: Ali [19] reviewed side-channel threats in HE implementations, including timing attacks, cache attacks, and microarchitectural vulnerabilities. Constant-time implementations are essential to prevent timing-based leakage. Memory access patterns must be carefully designed to avoid cache-based side channels. The OHHE framework proposal addresses these implementation challenges through systematic security analysis and hardened implementations.

Memory Safety: FHE implementations involve complex polynomial arithmetic and memory management. Buffer over- flows, use-after-free vulnerabilities, and other memory safety issues can compromise security. Using memory-safe languages (Rust) or rigorous testing and formal verification can mitigate these risks.

Library Security: Practitioners typically use existing FHE libraries rather than implementing schemes from scratch. Library selection must consider security track record, code quality, and active maintenance. OpenFHE, SEAL, and HElib have undergone extensive scrutiny, but vulnerabilities may still exist [8], [16].

### F. Operational Security

Key Management: Secure key generation requires high- quality randomness. Key storage must protect against unau- thorized access—hardware security modules (HSMs) or TEEs provide secure key storage [5]. Key distribution in multi-user scenarios requires authenticated channels to prevent man-in- the-middle attacks [3].

Access Control: Fine-grained access control policies must govern who can submit computations, access results, and manage keys. Attribute-based encryption and capability-based systems can enforce complex policies while maintaining en- cryption.

Monitoring and Auditing: Production deployments re- quire continuous monitoring for anomalous behavior, resource Exhaustion attacks, and potential security incidents. Audit logs must capture key operations (key generation, encryption, computation requests) for forensic analysis Secure Updates: FHE systems must support secure updates of cryptographic parameters, library versions, and application code without compromising

ongoing computations or stored ciphertexts. Migration strategies must ensure backward com- patibility and security during transitions.

### G. Security-Performance Trade-offs

Security and performance are often in tension. Higher secu- rity levels require larger parameters, increasing computational and storage costs. Our framework enables systematic analysis of these trade-offs:

Parameter Selection: Moving from 128-bit to 256-bit security significantly increases costs. Applications must assess threat models to determine appropriate security levels [8].

Verification Overhead: Verifiable FHE introduces addi- tional computational overhead for proof generation and veri- fication. PEEV [11] demonstrates that careful protocol design can achieve low overheads, but applications must balance integrity guarantees against performance requirements.

Multi-Key Complexity: Multi-key FHE schemes intro- duce additional computational and communication overhead compared to single-key schemes. SecFed [5] achieves 3× improvement over pure TEE schemes and 2× over traditional HE schemes, but still incurs costs beyond single-key FHE.

### H. Security Recommendations

Based on our analysis, we provide the following security recommendations:

- Use Standardized Parameters: Select parameter sets that have undergone community scrutiny and meet target security levels (typically 128-bit for most applications).
- Implement Defense-in-Depth: Combine FHE with com- plementary security mechanisms (TEE, verifiable compu- tation, blockchain integrity) for critical applications.
- Deploy Post-Quantum Channels: Use PQC key ex- change (e.g., Kyber) for public material distribution to ensure quantum resistance [1].
- Conduct Security Audits: Regular security audits should cover cryptographic parameters, implementation vulnera- bilities, and operational security practices.
- Monitor for Side Channels: Test implementations for timing and cache-based side channels; use constant-time implementations where possible [19].
- Verify Computational Integrity: For high-value com- putations, deploy verifiable FHE or redundant execution to detect malicious servers [11].

## Implementation And Discussion

### A. Implementation Challenges

FHE systems face substantial time-space complexity de- spite GPU acceleration achieving 70-252× speedups [9], [16]. Memory constraints significantly impact performance—Tsuji et al. [8] found TFHEpp outperforms OpenFHE in DRAM- limited environments. Bootstrapping remains a bottleneck at 0.220-0.227s per gate [3], requiring careful circuit design. Privacy-aware compilers [14] address developer productiv- ity by abstracting cryptographic complexity while achieving 4.92× speedups.

### B. Deployment Strategies

GPU acceleration is essential for production—single-GPU implementations provide 78-252× speedups over CPU libraries [16]. DRAM-rich instances prevent performance degradation during key generation and NTT operations [8]. Library se- lection should consider OpenFHE, SEAL, HElib for CPU, and FIDESlib [9] or HE-Booster [16] for GPU acceleration. Hybrid architectures combining FHE with TEE [5], PQC [1], or blockchain [10] improve practicality.

### C. Application-Specific Considerations

CKKS suits ML inference with practical latency (1.40s for ResNet20 [7]). TFHE enables PSI with low communication overhead (0.004 MB vs. 20.125 MB for RSA [3]). BFV/BGV optimize database operations [8]. Edge-cloud partitioning [14] and multi-key FHE [3], [5], [10] enable distributed scenarios.

### D. Future Directions

Critical research needs include: standardized benchmarking across libraries and hardware; bootstrapping optimization be- yond current 70× GPU speedups [9]; verifiable FHE at scale
[11]; memory-efficient implementations for cloud economics; advanced compiler technologies for general-purpose program- ming; cross-layer co-design of cryptography and hardware [1], [7], [16]; and hybrid systems combining FHE with comple- mentary privacy technologies [4], [5].

### E. Recommendations

Practitioners should: (1) use mature libraries (OpenFHE, SEAL, HElib); (2) deploy GPU acceleration for production;
(3) Provision adequate DRAM; (4) leverage privacy-aware compilers [14]; (5) optimize circuit depth; (6) consider hybrid
Approaches [1], [5], [10]; (7) conduct thorough security anal- ysis [19]; and (8) benchmark application-specific workloads before deployment.

## Conclusion and Future Work

This paper presented a comprehensive multi-layer eval- uation framework for encrypted data processing in cloud environments using FHE with re-encryption capabilities. Our framework systematically addresses application, encryption, storage, and computation layers, providing clear separation of concerns while enabling holistic optimization.

Key findings demonstrate that scheme selection depends on workload—CKKS excels for ML (1.40s ResNet20 infer- ence [7]), TFHE for boolean circuits (0.220s per gate [3]), and BGV/BFV for integer operations [8]. GPU acceleration provides 70-252× speedups [9], [16], with multi-GPU scal- ing achieving 7.66× additional improvement [16]. Privacy- aware compilers reduce latency by 4.92× through intelligent
Partitioning [14]. Security analysis reveals that defense-in- depth approaches combining FHE with TEE [5], PQC [1], blockchain [10], and verifiable computation [11] provide ro- bust protection for production deployments. Implementation analysis shows memory constraints signif- icantly impact performance [8], highlighting the importance of DRAM provisioning and memory-efficient libraries. Devel- oper productivity challenges can be addressed through privacy- aware compilers that abstract cryptographic complexity [14]. Future research should focus on: standardized benchmark- ing methodologies; bootstrapping optimization beyond current 70× GPU speedups; verifiable FHE at scale; memory-efficient implementations; advanced compiler technologies; cross-layer co-design; and hybrid cryptographic systems. As FHE technol- ogy matures, our framework provides a foundation for secure, privacy-preserving computation in untrusted cloud environ- ments while maintaining practical performance and robust security guarantees.

## References

[1] Hamza, “A Quantum-Resistant FHE Framework for Privacy-Preserving Image Processing in the Cloud,” Algorithms, 2025. DOI: 10.3390/a18080480

[2] Saker et al., “Comparative analysis of homomorphic encryption schemes for encrypted image processing in OpenStack using TenSEAL,” Az Eszterha´zy Ka´roly Tana´rke´pzo″ Fo″iskola tudoma´nyos ko¨zleme´nyei, 2025. DOI: 10.33039/ami.2025.09.001
[3] Fan et al., “Cloud-Assisted Private Set Intersection via Multi-Key Fully Homomorphic Encryption,” Mathematics, vol. 11, no. 8, 2023. DOI: 10.3390/math11081784
[4] Kenhove et al., “MOZAIK: A Privacy-Preserving Analytics Platform for IoT Data Using MPC and FHE,” 2026.
[5] Cai et al., “SecFed: A Secure and Efficient Federated Learning Based on Multi-Key Homomorphic Encryption,” IEEE Transactions on Depend- able and Secure Computing, 2023. DOI: 10.1109/tdsc.2023.3336977
[6] Ameur et al., “Handling security issues by using homomorphic encryp- tion in multi-cloud environment,” Procedia Computer Science, 2023. DOI: 10.1016/j.procs.2023.03.050
[7] Kim et al., “HyPHEN: A Hybrid Packing Method and Optimizations for Homomorphic Encryption-Based Neural Networks,” arXiv.org, 2023. DOI: 10.48550/arXiv.2302.02407
[8] Tsuji et al., “Comparison of FHE Schemes and Libraries for Efficient Cryptographic Processing,” in Proc. ICNC, 2024. DOI: 10.1109/icnc59896.2024.10556382
[9] Agullo´-Domingo et al., “FIDESlib: A Fully-Fledged Open-Source FHE Library for Efficient CKKS on GPUs,” in Proc. ISPASS, 2025. DOI: 10.1109/ispass64960.2025.00045
[10] Liao et al., “A multikey fully homomorphic encryption privacy pro- tection protocol based on blockchain for edge computing system,” Concurrency and Computation: Practice and Experience, 2022. DOI: 10.1002/cpe.7539
[11] AUTHOR ID et al., “PEEV: Parse Encrypt Execute Verify - A Verifiable FHE Framework,” IEEE Access, 2024. DOI: 10.1109/ac- cess.2024.3424420
[12] Xiao et al., “Privacy Protection Anomaly Detection in Smart Grids Based on Combined PHE and TFHE Homomorphic Encryption,” Electronics, vol. 14, no. 12, 2025. DOI: 10.3390/electronics14122386
[13] Zheng et al., “Accurate and Efficient Privacy-Preserving Feature Ex- traction on Encrypted Images,” IEEE Transactions on Dependable and Secure Computing, 2025. DOI: 10.1109/tdsc.2024.3524121
[14] Kim et al., “Privacy Set: Privacy Authority-Aware Compiler for Homo- morphic Encryption on Edge-Cloud System,” IEEE Internet of Things Journal, 2024. DOI: 10.1109/jiot.2024.3437356
[15] Li et al., “Privacy preserving via multi-key homomorphic encryption in cloud computing,” Journal of information security and applications, 2023. DOI: 10.1016/j.jisa.2023.103463.
[16] Wang et al., “HE-Booster: An Efficient Polynomial Arithmetic Acceleration on GPUs for Fully Homomorphic Encryption,” IEEE Transactions on Parallel and Distributed Systems, 2023. DOI: 10.1109/TPDS.2022.3228628.
[17] Olaymi, “Performance and security analysis of fully homomorphic encryption in cloud-based healthcare blockchain,” 2023.
[18] Ilyenko et al., “Practical Aspects of Using Fully Homomorphic

Encryption Systems to Protect Cloud Computing," 2023.
[19] Ali, "Towards Practical Homomorphic Encryption: A Review of Implementations, Side-Channel Threats, and the OHHE Framework Proposal,"2023.
[20] Yadavalli et al., "Homomorphic Encryption Methods Applied to Cloud Computing: A Practical Architecture for Elastic, Verifiable Confidential Compute," 2023.