


# Integrating Fully Homomorphic Encryption and Zero-Knowledge Proofs for Efficient Verifiable Computation

UmmeAmmara Qureshi, Bhumika Doshi, Aditya More, Kashyap Joshi, and Kapil Kumar \*

Department of Biochemistry and Forensic Science, Gujarat University, Ahmedabad, Gujarat 380009, India;  
e-mail : qureshiammara084@gmail.com; bhumikasdoshi@gmail.com; moreadityarajesh@gmail.com;  
kashyapjoshi.it@gmail.com; kkforensic@gmail.com

\* Corresponding Author : Kapil Kumar 

**Abstract:** Fully Homomorphic Encryption (FHE) enables computation on encrypted data with end-to-end confidentiality; however, its practical adoption remains limited by substantial computational costs, including long encryption and decryption times, high memory consumption, and operational latency. Zero-Knowledge Proofs (ZKPs) complement FHE by enabling correctness verification without revealing sensitive information, although they do not support encrypted computation independently. This study integrates both techniques to enable encrypted computation with verifiably consistent results. A prototype system is implemented in Python using Microsoft SEAL for homomorphic encryption and PySNARK for Zero-Knowledge Proof verification. Experiments are conducted on standard consumer-grade hardware (Intel i5, 8 GB RAM, Ubuntu 22.04) using datasets ranging from 100 MB to 1 GB. The evaluation focuses on encryption and decryption time, homomorphic computation latency, memory usage, and proof generation overhead. Experimental results show that integrating ZKPs introduces a moderate and stable runtime overhead of approximately 15–20%, as analyzed in Section 4, while enabling verification without plaintext disclosure. Ciphertext expansion remains a notable limitation, with observed growth of approximately 30–40× relative to plaintext size, consistent with prior FHE implementations. Despite these overheads, the system demonstrates feasible scalability for datasets up to 1 GB on mid-level hardware. Overall, the results indicate that the integrated FHE+ZKP approach provides a practical balance between confidentiality, verifiability, and performance, supporting its applicability to privacy-preserving scenarios such as secure cloud computation, encrypted data analytics, and confidential data processing under realistic resource constraints.

**Keywords:** Cryptography; Fully Homomorphic Encryption; Microsoft SEAL; Privacy-Preserving Systems; Python; Secure Computation; Zero-Knowledge Proofs; zk-SNARK.

Received: August, 13<sup>th</sup> 2025

Revised: December, 6<sup>th</sup> 2025

Accepted: December, 8<sup>th</sup> 2025

Published: December, 27<sup>th</sup> 2025



**Copyright:** © 2025 by the authors.  
Submitted for possible open access  
publication under the terms and  
conditions of the Creative Commons  
Attribution (CC BY) licenses  
(<https://creativecommons.org/licenses/by/4.0/>)

## 1. Introduction

The unprecedented growth of digital communication and computing technologies has introduced significant challenges to privacy and security. Traditional encryption methods, such as AES or RSA, maintain data confidentiality by encrypting data at rest or in transit. However, these schemes require decryption before computation, which exposes plaintext and consequently compromises security during the processing stage [1], [2].

Fully Homomorphic Encryption (FHE) addresses this limitation by enabling computations to be performed directly on encrypted data without requiring decryption, thereby ensuring end-to-end data confidentiality [3]–[5]. Recent studies highlight the growing adoption of FHE across domains such as encrypted databases and healthcare applications, demonstrating notable practical advances despite persistent computational challenges [6]. Subsequent works have further improved efficiency by transitioning to Ring-LWE–based constructions, which now form the foundation of most modern FHE schemes [7].

Despite its transformative potential, FHE remains burdened by substantial computational overhead, including slow encryption and decryption processes, high memory consumption, and latency even for basic homomorphic operations [8], [9]. These limitations significantly hinder its adoption in real-world scenarios, particularly within resource-constrained

environments such as edge devices or student-grade computing systems. Efforts to mitigate these overheads have primarily focused on hardware-conscious optimizations, indicating that both FHE and Zero-Knowledge Proofs (ZKPs) can be adapted for energy-constrained IoT and edge computing applications [10].

In contrast, ZKPs are cryptographic techniques that allow a prover to demonstrate to a verifier that a computation has been executed correctly without revealing any underlying sensitive information [11], [12]. While ZKPs do not support encrypted computation on their own, they provide a powerful mechanism for ensuring verifiability. More recently, ZKP frameworks have been applied to privacy-preserving artificial intelligence models to enable verifiable inference without exposing sensitive training data [13].

The integration of FHE and ZKPs combines the strengths of both approaches, FHE facilitates computation over ciphertext, while ZKPs ensure that such computations are verifiable and trustworthy. This combination is particularly relevant for applications such as secure cloud computing, privacy-preserving machine learning, and encrypted financial systems, where both confidentiality and correctness are equally critical requirements [14], [15]. By combining encrypted computation with cryptographic verification, this work provides practical insights into building trustworthy privacy-preserving systems. The main contributions of this work are as follows:

- **Performance Analysis:** Measuring encryption and decryption time, homomorphic operation latency, memory usage, and proof generation overhead under typical system constraints.
- **Comparative Analysis:** Conducting a formal comparison between FHE-only and FHE+ZKP implementations in terms of security, correctness, and computational efficiency trade-offs.

Overall, this work demonstrates that although integrating ZKPs introduces additional overhead, it significantly enhances the verifiability and trustworthiness of computation within the encrypted domain. The results provide strong evidence that FHE+ZKP systems are practical for real-world privacy-preserving applications, even in moderately resource-constrained environments, thereby addressing an important gap in the secure computation research landscape.

## 2. Related Works

Research on FHE and ZKPs has expanded rapidly in recent years, positioning both techniques at the core of privacy-preserving computation. Existing studies can broadly be categorized into foundational surveys, performance optimization efforts, integration of FHE with ZKPs, and applications in real-world privacy-critical systems. This section reviews the most relevant works within these categories to contextualize the present study.

Several comprehensive surveys have examined the theoretical foundations and evolution of FHE. Acar et al. [4] provided an in-depth overview of homomorphic encryption schemes, highlighting key limitations such as high computational overhead and scalability challenges. Gentry et al. [16] analyzed FHE constructions based on the Learning With Errors (LWE) problem, outlining both theoretical advances and unresolved challenges. Peikert [17] offered a detailed review of lattice-based cryptography, which underpins most modern FHE schemes. In addition, Fang and Qian [18] discussed the application of homomorphic encryption in machine learning, with particular emphasis on practical constraints and performance trade-offs encountered in real-world deployments.

Beyond theoretical foundations, a substantial body of work has focused on improving the performance of privacy-preserving computation. Ameur [19] explored cryptographic optimization techniques to enhance the efficiency of secure computation, while Lavin et al. [20] surveyed recent ZKP applications that leverage batching strategies and parameter tuning to reduce latency. Martin and Amos [21] demonstrated that carefully selecting the polynomial and coefficient modulus parameters can significantly affect runtime performance in FHE systems. Furthermore, hardware-aware acceleration has received growing attention; Li and Zong [22] implemented FHE computations on GPUs and FPGAs, achieving notable reductions in computation time for encrypted workloads.

More recent research has emphasized integrating FHE with ZKPs to enable verifiable encrypted computation. Khandait and Pattanshetti [23] proposed lightweight ZKP protocols optimized for IoT environments, enabling correctness verification with minimal resource

consumption. Xing et al. [24] conducted a comprehensive survey on the use of ZKPs to verify machine learning computations, laying the groundwork for verifiable encrypted inference models. Gouert et al. [25] benchmarked multiple FHE libraries using standardized benchmarks, reporting moderate overheads ( $\approx 10\text{--}30\%$ ) while confirming their practicality for real-world secure computation. Additionally, Keršič et al. [26] investigated hybrid zk-SNARK and FHE frameworks in the context of blockchain scalability, reporting promising improvements in both performance and reliability.

FHE and ZKP techniques have also been applied across a wide range of privacy-critical domains. Gorantala et al. [27] highlighted their use in secure cloud computing environments, where confidentiality is preserved alongside verifiable correctness. Recent studies further indicate applications in privacy-preserving multi-party computation and encrypted deep learning, in which FHE ensures data confidentiality while ZKPs provide trust in the validity of the computation results.

Despite these advances, several challenges remain unresolved. Ciphertext expansion, memory overhead, and proof generation costs continue to pose significant barriers to scalability. Moreover, existing studies have largely focused on high-end or specialized hardware platforms, with limited benchmarking of integrated FHE+ZKP systems on low-resource or consumer-grade environments. This work aims to address these gaps by systematically evaluating the performance and practical feasibility of FHE+ZKP integration under realistic computational constraints, thereby complementing prior research with empirical evidence from moderate, widely accessible hardware.

### 3. Methodology

This study adopts a structured experimental methodology to evaluate the feasibility of integrating FHE with ZKP in realistic, resource-constrained computing environments. The methodology is designed to address two primary challenges: (i) the substantial computational overhead inherent in FHE-based encrypted computation, and (ii) the lack of intrinsic verifiability in conventional encrypted processing pipelines. To address these challenges, a working prototype was implemented in Python using Microsoft SEAL for homomorphic encryption and PySNARK for ZKP generation and verification.

#### 3.1. System Design and Architecture

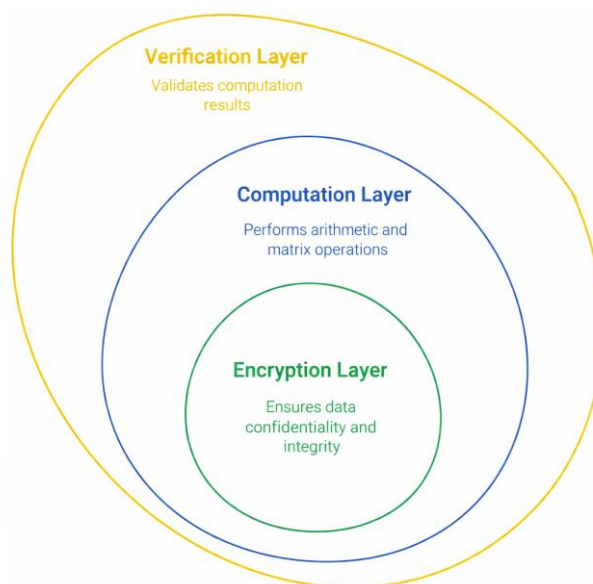
The proposed system follows a layered architecture consisting of three logical components: an encryption layer, a computation layer, and a verification layer. This architectural separation is illustrated in Figure 1, which highlights the functional responsibilities of each layer and their role in maintaining confidentiality, correctness, and modularity.

The encryption layer ensures data confidentiality by encrypting sensitive inputs using Fully Homomorphic Encryption schemes. The computation layer performs arithmetic and matrix-based operations directly on ciphertexts, thereby preventing any exposure of plaintext during processing. The verification layer verifies the correctness of encrypted computations using ZKPs without requiring access to the underlying data. This layered abstraction simplifies system extensibility and clearly separates cryptographic responsibilities across the pipeline.

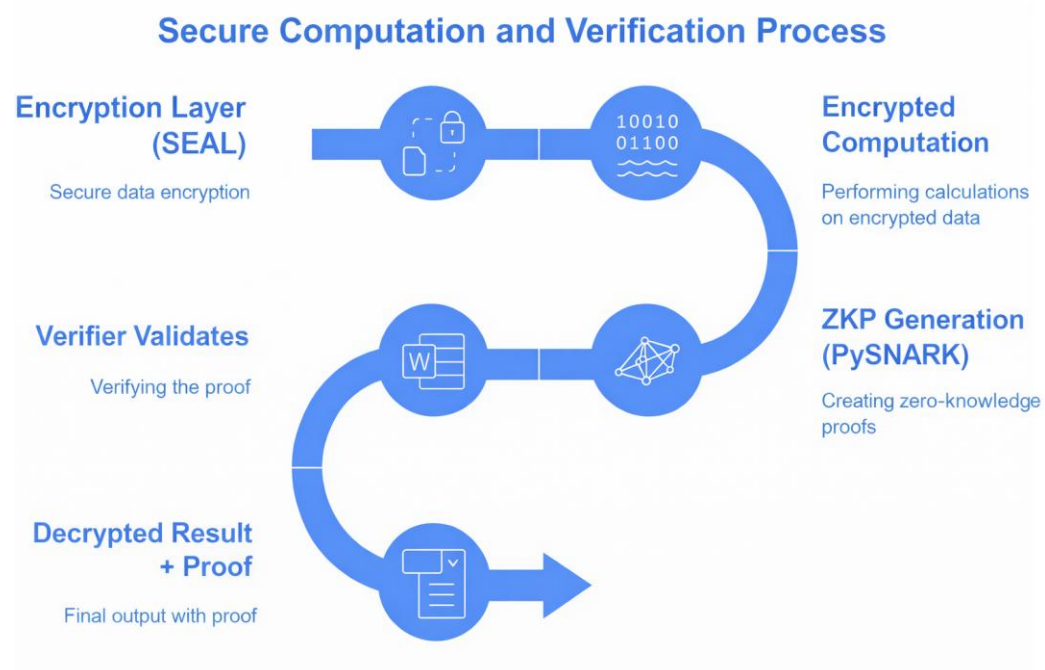
#### 3.2. Secure Computation and Verification Workflow

While Figure 1 presents a structural view of the system, Figure 2 illustrates the execution flow of secure computation and verification, depicting the end-to-end processing pipeline. In this workflow, data are first encrypted using Microsoft SEAL and then subjected to homomorphic computation. The resulting ciphertexts are subsequently used to generate Zero-Knowledge Proofs, which allow a verifier to confirm correctness without accessing plaintext values. Only after successful verification is the ciphertext optionally decrypted and returned together with its corresponding proof.

This dual-view representation, architectural (Figure 1) and procedural (Figure 2), ensures both conceptual clarity and precise understanding of execution order within the proposed framework.



**Figure 1.** Layered architecture illustrating the encryption, computation, and verification components of the proposed system.



**Figure 2.** Secure computation and verification process, showing the sequential flow from encryption and homomorphic computation to ciphertext-level proof generation, verification, and result decryption.

### 3.2.1. Encryption Computation Pipeline

Homomorphic computations were performed using Microsoft SEAL (version 4.1), supporting both the BFV scheme for integer arithmetic and the CKKS scheme for approximate real-number operations. All experiments were conducted on standard consumer-grade hardware consisting of an Intel Core i5 processor (2.5 GHz, 4 cores), 8 GB RAM, and Ubuntu 22.04 LTS, using Python bindings for SEAL and PySNARK.

The encryption parameters were selected to balance security and performance. A polynomial modulus degree of 8192 and a coefficient modulus totaling 218 bits (composed of 60-, 40-, 40-, and 78-bit primes) were used. For CKKS, a scaling factor of  $2^{40}$  and a batch size of 4096 was applied to enable efficient vectorized computation. These parameters follow established recommendations from FV-style implementations [28] and provide approximately

128-bit security. Additional experiments using a higher polynomial modulus degree (16384) were conducted to analyze latency–memory trade-offs. Benchmark datasets ranging from 100 MB to 1 GB were encrypted and processed to assess scalability.

### 3.2.2. ZKP Integration and Proof Generation

After encrypted computation, the resulting ciphertexts are forwarded to the verification stage rather than decrypted immediately. PySNARK (version 0.6.3) is employed to generate non-interactive Zero-Knowledge Proofs that certify the correctness of the homomorphic operations. This design enables third-party verification while preserving strict confidentiality, as no plaintext values are revealed during proof generation or verification.

It is important to emphasize that the Zero-Knowledge Proofs generated in this framework are not applied to decrypted outputs. Instead, the proofs are constructed directly from the homomorphic computation process, ensuring correctness without revealing any plaintext values at any stage of execution.

### 3.2.3. Ciphertext-Level ZKP Verification

Unlike conventional approaches that verify computation results after decryption, the proposed framework enforces verification directly over encrypted values. The Zero-Knowledge Proof validates the algebraic correctness of homomorphic operations in the ciphertext domain, thereby binding the proof to the encrypted computation itself rather than its decrypted outcome.

In the proposed framework, Zero-Knowledge Proof verification is performed directly in the ciphertext domain. Instead of verifying correctness after decryption, the proof is constructed from the homomorphic computation itself, ensuring that verification remains tightly coupled with encrypted execution. Formally, given encrypted inputs  $Enc(a)$ ,  $Enc(b)$ , and a homomorphic result  $Enc(c)$ , the ZKP circuit commits to the ciphertext witnesses and enforces the encrypted algebraic relation:

$$Enc(a) \oplus Enc(b) = Enc(c) \quad (1)$$

Where  $\oplus$  denotes the corresponding homomorphic operation (addition or multiplication). The prover generates an SNARK proof asserting that the encrypted outputs satisfy the claimed algebraic relation, using the same computation graph used during homomorphic evaluation. At no point are the plaintext values  $a$ ,  $b$ , or  $c$  decrypted or exposed. It is important to note that the ZKP circuit does not re-execute the internal cryptographic primitives of Microsoft SEAL. Instead, it verifies consistency relations among ciphertext representations generated by the homomorphic computation, thereby providing assurance that the reported encrypted results correspond to the claimed computation. This design enables correctness verification to be performed without plaintext access, while remaining compatible with scalable SNARK constructions for arithmetic verification [29]. In this framework, the ZKP is bound to the specific ciphertext outputs produced during homomorphic evaluation, ensuring that verification corresponds to the reported encrypted results rather than a generic or replayed computation.

### 3.2.4. Performance Evaluation and Comparative Baseline

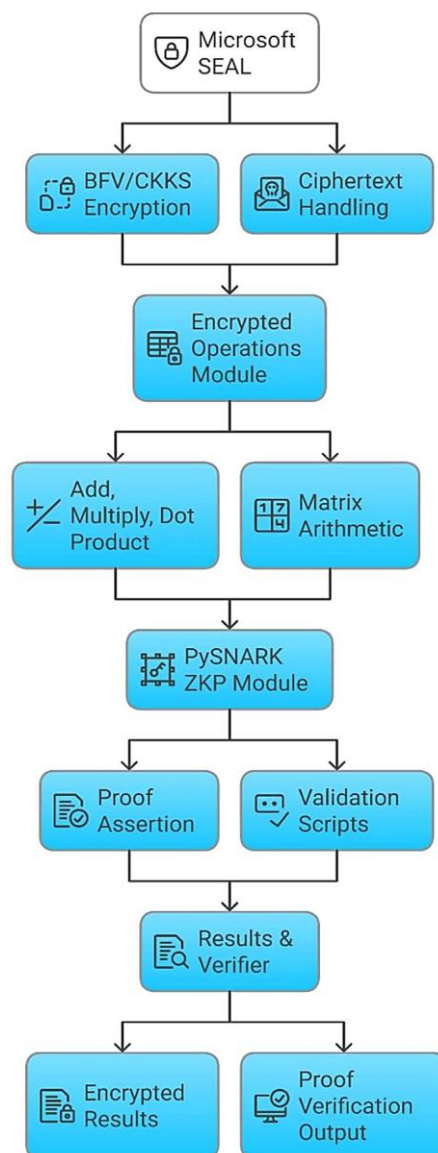
System performance was evaluated using four metrics: encryption and decryption time (milliseconds), computation latency (milliseconds), memory usage (megabytes), and proof generation overhead (milliseconds). All experiments were conducted under both FHE-only and FHE+ZKP configurations using identical datasets and parameter settings.

The comparative baseline distinguishes between pure encrypted computation (FHE-only) and verifiable encrypted computation (FHE+ZKP). In the FHE-only setting, correctness relies solely on trusted execution. In contrast, in the FHE+ZKP configuration, each homomorphic operation is accompanied by a Zero-Knowledge Proof that enables independent verification without access to plaintext.

## 3.3. System Architecture and Component Flow

The proposed system architecture integrates FHE with ZKP to jointly ensure confidentiality and verifiability of computations performed on ciphertexts. The architecture is designed in a modular and layered manner, clearly separating encryption, computation, and verification

responsibilities to improve clarity, extensibility, and implementation efficiency. An overview of the system components and their interactions is illustrated in Figure 3.



**Figure 3.** System architecture illustrating the integration of Microsoft SEAL-based encrypted computation with PySNARK-based zero-knowledge proof generation and verification.

### 3.3.1. Encryption Layer

The encryption layer employs Microsoft SEAL to encrypt and encode sensitive inputs before any computation is performed. Support is provided for both the BFV scheme, which enables exact integer arithmetic, and the CKKS scheme, which supports approximate real-number computations. All plaintext inputs are transformed into ciphertexts at this stage and retained exclusively in encrypted form throughout subsequent processing. As a result, no plaintext data are exposed during computation, thereby preserving end-to-end data confidentiality.

### 3.3.2. Computation Layer

The computation layer is responsible for executing arithmetic and matrix-based operations directly on ciphertexts. Encrypted data are processed by the Encrypted Operations Module, which performs homomorphic addition, multiplication, dot products, and matrix arithmetic using SEAL's evaluation primitives. All operations remain strictly within the encrypted domain, ensuring that intermediate values are never decrypted or revealed during execution.

### 3.3.3. Verification Layer

The verification layer utilizes PySNARK to generate Zero-Knowledge Proofs that certify the correctness of computations performed on ciphertexts. Proof assertions and validation scripts are generated to enable independent verification by third parties without disclosing any underlying sensitive information. This layer ensures that correctness guarantees are cryptographically enforced while maintaining full data privacy.

### 3.3.4. End-to-End Execution Flow

The end-to-end execution pipeline begins with BFV or CKKS encryption of input data, followed by homomorphic arithmetic and matrix computations within the encrypted domain. After computation, Zero-Knowledge Proofs are generated to verify correctness at the ciphertext level. Upon successful verification, the encrypted results may be decrypted, and the decrypted output is returned together with its corresponding proof. This design ensures that both confidentiality and trust are preserved throughout the computation lifecycle.

### 3.3.5. Key Design Goals

Four primary design goals guide the architecture:

- Confidentiality: Ensure that plaintext data are never exposed at any stage of the computation pipeline.
- Verifiability: Provide Zero-Knowledge Proofs that enable correctness verification without revealing sensitive information.
- Efficiency: Minimize computational and memory overheads to make the FHE+ZKP integration practical even in moderately resource-constrained environments.
- Modularity: Isolate encryption, computation, and verification components to facilitate extensibility and seamless integration of alternative FHE schemes or ZKP libraries.

This modular design provides a flexible foundation for future extensions, enabling additional homomorphic operations or alternative verification frameworks to be incorporated with minimal architectural changes. Moreover, the clear separation of concerns simplifies benchmarking and performance evaluation across different system configurations.

## 4. Results and Discussion

### 4.1. Encryption and Decryption Time

Encryption and decryption times were measured to evaluate the baseline cost of homomorphic processing and the additional overhead introduced by ZKP integration. As expected, FHE incurs substantially higher encryption and decryption costs compared to conventional cryptographic schemes due to polynomial modulus operations and ciphertext expansion. The integration of ZKPs further increases encryption time, as additional commitments and constraints must be generated to support proof construction.

Table 1 compares encryption and decryption times between FHE-only and FHE+ZKP configurations across different dataset sizes. The results show that ZKP integration introduces a moderate overhead of approximately 14–17%, which remains relatively stable as dataset size increases. These results indicate that although ZKP integration increases cryptographic cost, the overhead remains within practical limits even for larger datasets.

**Table 1.** Encryption and Decryption Time (ms) for FHE-only and FHE+ZKP across dataset sizes

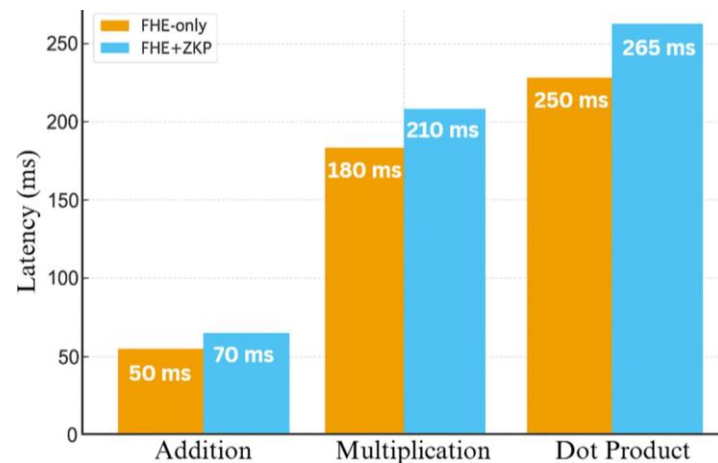
Dataset Size	FHE-only (Enc)	FHE+ZKP (Enc)	FHE-only (Dec)	FHE+ZKP (Dec)	Overhead (%)
100 MB	720	820	600	690	+15%
250 MB	890	1010	720	830	+14%
500 MB	1050	1200	820	960	+16%
1 GB	1320	1520	1080	1250	+17%

### 4.2. Homomorphic Operation Latency

The latency of core homomorphic operations, addition, multiplication, and dot product, was evaluated under both FHE-only and FHE+ZKP configurations. Operation latency is a critical factor in determining the feasibility of encrypted computation in real-world



applications. Figure 4 compares the latencies of the two configurations. Addition operations exhibit the lowest latency and scale efficiently, whereas multiplication and dot-product operations incur higher costs due to ciphertext size growth and increased arithmetic complexity. The inclusion of ZKP introduces an additional latency overhead of approximately 15–20% across all operations. Despite this overhead, the added latency is compensated by the guarantee of verifiable correctness, which is essential in untrusted execution environments.



**Figure 4.** Comparison of homomorphic operation latency (addition, multiplication, dot product) for FHE-only and FHE+ZKP

### 4.3. Memory Consumption

Memory consumption was measured to assess the feasibility of deploying the proposed system on moderate hardware platforms. As dataset size increases, memory usage grows due to ciphertext expansion and intermediate buffers required for encrypted computation. ZKP integration further increases memory demand, primarily due to proof-related storage requirements.

Table 2 summarizes memory usage under both configurations. The results show that ZKP integration increases memory usage by approximately 11–16%. Importantly, total memory consumption remained below 6 GB even for 1 GB datasets, indicating that the system remains feasible on mid-level hardware.

**Table 2.** Memory usage (MB) for FHE-only and FHE+ZKP

Dataset Size	FHE-only	FHE+ZKP	Increase (%)
100 MB	700	780	+11.4%
250 MB	750	850	+13.3%
500 MB	820	940	+14.6%
1 GB	930	1080	+16.1%

### 4.4. Proof Generation Overhead

To isolate the cost of ZKP integration, proof-generation overhead was measured separately. The average proof size ranged from 2–4 KB, and proof verification took less than 0.5 seconds, even for the largest dataset tested. These results demonstrate that ZKP verification remains lightweight relative to the overall computation cost. The observed performance of proof generation and verification aligns with trends reported in widely used zk-SNARK frameworks such as libsnark [30], supporting the practicality of integrating ZKPs with FHE-based computation. It is important to note that the reported proof generation and verification costs correspond to ciphertext-level verification, as the Zero-Knowledge Proofs are constructed directly over homomorphic relations without accessing decrypted outputs.

### 4.5. Ciphertext Size Analysis

Ciphertext size plays a critical role in both computation latency and memory consumption. Under the BFV scheme, ciphertexts were observed to be smaller than those produced



by CKKS, reflecting BFV's suitability for integer arithmetic. However, CKKS provides greater flexibility for real-number and approximate computations. Across all experiments, ciphertext expansion remained in the range of  $30\text{--}40\times$  relative to plaintext size. While this behavior is consistent with prior FHE implementations, it continues to represent the primary scalability bottleneck for large-scale encrypted computation.

#### 4.6. Ablation Study

An ablation study was conducted to examine the isolated effects of parameter tuning on system performance and to analyze how tuning interacts with Zero-Knowledge Proof (ZKP) integration. Four configurations were evaluated: FHE-only (default), FHE-only (tuned), FHE+ZKP (default), and FHE+ZKP (tuned). The results are summarized in Table 3.

**Table 3.** Ablation study results

Configuration	Encryption Time (ms)	Decryption Time (ms)	Latency (ms)	Memory (MB)	Proof Overhead (ms)
FHE-only (Default)	1200	950	180	820	–
FHE-only (Tuned)	980	780	150	760	–
FHE+ZKP (Default)	700	615	110	930	150
FHE+ZKP (Tuned)	640	580	95	880	140

The results indicate that parameter tuning consistently reduces encryption time, decryption time, and homomorphic computation latency across both FHE-only and FHE+ZKP configurations. As expected, tuning improves performance by adjusting polynomial modulus degree, coefficient modulus selection, and batching behavior, albeit at the cost of reduced security margins.

Notably, some FHE+ZKP configurations exhibit lower latency than the untuned FHE-only baseline. This observation should not be interpreted as ZKP integration improving raw homomorphic performance. Instead, the effect arises from differences in parameter configurations and batching efficiency between the default and tuned settings, as well as runtime variability inherent to encrypted computation. When comparing configurations under equivalent parameter settings, ZKP integration consistently introduces additional overhead, as reported in Sections 4.1–4.4.

Overall, the ablation study highlights the critical role of parameter selection in practical FHE and FHE+ZKP deployments. Performance variations across configurations underscore the need for careful tuning to balance security guarantees, computational efficiency, and verification overhead in real-world encrypted computation systems.

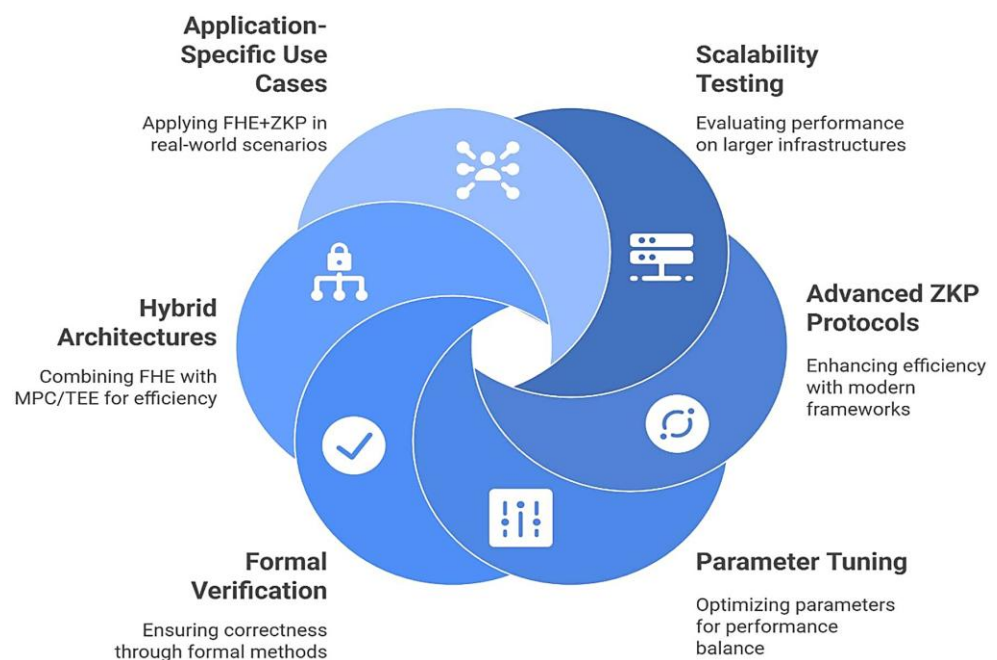
#### 4.7. Discussion and Key Insights

Overall, the experimental results confirm that the proposed integration of Fully Homomorphic Encryption (FHE) and Zero-Knowledge Proofs (ZKPs) achieves a practical balance between security guarantees and system performance. While additional computational overhead is unavoidable due to proof generation, the observed costs remain stable as dataset sizes increase, indicating predictable, manageable scalability. Importantly, performing verification directly at the ciphertext level ensures that computational correctness can be verified without compromising data confidentiality. Several key insights emerge from the experimental evaluation:

- **Established Trade-off:** Integrating ZKPs introduces a consistent runtime overhead of approximately 15–20%, primarily attributable to proof constraint generation. However, this overhead enables cryptographic verification of encrypted computations without relying on trusted execution environments or plaintext access.
- **Scalability Feasibility:** The system successfully processed datasets up to 1 GB on moderate, consumer-grade hardware, demonstrating suitability for student-level experimentation and small- to medium-scale enterprise deployments.
- **Low Verification Cost:** Proof verification remained lightweight, requiring less than 0.5 seconds per proof, which supports near-real-time third-party verification in practical deployment scenarios.

- **Persistent Bottlenecks:** Ciphertext expansion, observed at approximately 30–40× relative to plaintext size, remains the dominant limitation. This suggests that compression-aware FHE schemes, hardware acceleration, or hybrid cryptographic approaches will be essential to improve scalability further.

These insights naturally motivate several future research directions aimed at improving scalability, efficiency, and practical deployment of verifiable encrypted computation, as summarized in Figure 5.



**Figure 5.** Summary of experimental findings and prospective research directions for secure, scalable, and verifiable encrypted computation, highlighting performance trade-offs, scalability limits, and future optimization pathways.

## 5. Conclusions

This paper presents a comprehensive study of integrating FHE and ZKPs to enable secure computation with verifiable correctness. By combining homomorphic computation with ciphertext-level proof verification, the proposed framework ensures that sensitive data remains confidential throughout the computation process while allowing independent verification of computational correctness. A prototype implementation based on Microsoft SEAL and PySNARK was evaluated under realistic hardware constraints. The experimental results demonstrated that ZKP integration introduces a moderate and predictable performance overhead, while verification costs remain lightweight and scalable. These findings confirm that verifiable encrypted computation is not only theoretically sound but also practically achievable on moderate computing platforms.

Despite these promising results, this study is subject to several limitations that should be acknowledged. First, all experiments were conducted in a single-node environment using consumer-grade hardware, with datasets limited to 1 GB, which limits conclusions about large-scale, distributed, or high-throughput deployment scenarios. Second, the evaluation focused exclusively on the BFV and CKKS encryption schemes in combination with PySNARK, and did not consider alternative FHE constructions, more recent ZKP frameworks, or hardware-accelerated implementations. Third, the assessment relied primarily on empirical performance measurements; while ciphertext-level correctness was verified via Zero-Knowledge Proofs, formal verification of the overall encrypted computation pipeline was not explored.

These limitations naturally motivate several directions for future research. Scalability beyond single-node environments should be investigated through experiments on distributed systems, GPU/TPU platforms, or FPGA accelerators. Incorporating more advanced Zero-Knowledge Proof protocols, such as zk-STARKs, Bulletproofs, or Halo2, may further reduce

proof size and verification latency compared to PySNARK. In addition, adaptive parameter-tuning strategies could be developed to balance security guarantees and performance based on workload characteristics dynamically. Further extensions include formal verification of encrypted computation pipelines, exploration of hybrid architectures combining FHE with Multi-Party Computation or Trusted Execution Environments, and application-specific evaluations in domains such as encrypted machine learning, secure cloud analytics, e-voting, and confidential financial transactions. Overall, this work demonstrates that integrating FHE and ZKP provides a viable foundation for building secure, trustworthy, and privacy-preserving computation systems, while also identifying key technical challenges that must be addressed to support broader real-world adoption.

**Author Contributions:** Conceptualization: U.Q.; Methodology: U.Q.; Validation: B.D., A.M., K.J.; Formal analysis: U.Q., A.M.; Investigation: B.D.; Resources: K.K.; Data curation: K.J.; Writing—original draft: U.Q.; Writing—review & editing: K.K., A.M., B.D.; Visualization: K.J.; Supervision: A.M., K.J.; Project administration: K.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Acknowledgments:** The authors express their sincere gratitude to the Department of Biochemistry and Forensic Science, Gujarat University, for providing the necessary facilities, technical resources, and administrative support that greatly contributed to the completion of this research. The team acknowledges the constructive guidance and encouragement from Dr. Kapil Kumar throughout the research process.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- [1] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On Data Banks and Privacy Homomorphisms," in *Foundations of Secure Computation*, New York: Academic Press, 1978, pp. 169–180.
- [2] M. E. Zhao and Y. Geng, "Homomorphic Encryption Technology for Cloud Computing," in *Procedia Computer Science*, Jan. 2018, vol. 154, pp. 73–83. doi: 10.1016/j.procs.2019.06.012.
- [3] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, May 2009, pp. 169–178. doi: 10.1145/1536414.1536440.
- [4] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–35, Jul. 2019, doi: 10.1145/3214303.
- [5] T. V. T. Doan, M.-L. Messai, G. Gavin, and J. Darmont, "A survey on implementations of homomorphic encryption schemes," *J. Supercomput.*, vol. 79, no. 13, pp. 15098–15139, Sep. 2023, doi: 10.1007/s11227-023-05233-z.
- [6] M. Maulyanda, R. Deviani, and A. Afdhaluzzikri, "Enhancing Medical Data Privacy: Neural Network Inference with Fully Homomorphic Encryption," *J. Electr. Eng. Comput.*, vol. 7, no. 1, pp. 114–124, Apr. 2025, doi: 10.33650/jeecom.v7i1.10875.
- [7] Z. Brakerski and V. Vaikuntanathan, "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages," in *Advances in Cryptology – CRYPTO 2011*, 2011, pp. 505–524. doi: 10.1007/978-3-642-22792-9\_29.
- [8] J. Zhang, X. Cheng, L. Yang, J. Hu, X. Liu, and K. Chen, "SoK: Fully Homomorphic Encryption Accelerators," *ACM Comput. Surv.*, vol. 56, no. 12, pp. 1–32, Dec. 2024, doi: 10.1145/3676955.
- [9] S. Gupta, R. Cammarota, and T. Šimunić, "MemFHE: End-to-end Computing with Fully Homomorphic Encryption in Memory," *ACM Trans. Embed. Comput. Syst.*, vol. 23, no. 2, pp. 1–23, Mar. 2024, doi: 10.1145/3569955.
- [10] V. Biksham and D. Vasumathi, "A lightweight fully homomorphic encryption scheme for cloud security," *Int. J. Inf. Comput. Secur.*, vol. 13, no. 3/4, p. 357, 2020, doi: 10.1504/IJICS.2020.109482.
- [11] N. Sheybani, A. Ahmed, M. Kinsy, and F. Koushanfar, "Zero-Knowledge Proof Frameworks: A Systematic Survey," *arXiv*. Apr. 27, 2025. [Online]. Available: <http://arxiv.org/abs/2502.07063>
- [12] E. Morais, T. Koens, C. van Wijk, and A. Koren, "A survey on zero knowledge range proofs and applications," *SN Appl. Sci.*, vol. 1, no. 8, p. 946, Aug. 2019, doi: 10.1007/s42452-019-0989-z.
- [13] Aruna Rao S L, "Zero Knowledge Proof for Privacy Preserving for Federated Learning in Healthcare Systems," *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 48s, pp. 970–977, May 2025, doi: 10.52783/jisem.v10i48s.9714.
- [14] A. Kokaj and E. Mollakuqe, "Mathematical Proposal for Securing Split Learning Using Homomorphic Encryption and Zero-Knowledge Proofs," *Appl. Sci.*, vol. 15, no. 6, p. 2913, Mar. 2025, doi: 10.3390/app15062913.
- [15] J. G. Dhokrat, N. Pulgam, T. Maktum, and V. Mane, "A Framework for Privacy-Preserving Multiparty Computation with Homomorphic Encryption and Zero-Knowledge Proofs," *Informatica*, vol. 48, no. 21, Nov. 2024, doi: 10.31449/inf.v48i21.6562.
- [16] C. Gentry, A. Sahai, and B. Waters, "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based," in *Advances in Cryptology – CRYPTO 2013*, 2013, pp. 75–92. doi: 10.1007/978-3-642-40041-4\_5.

- [17] C. Peikert, “A Decade of Lattice Cryptography,” *Found. Trends Theor. Comput. Sci.*, vol. 10, no. 4, pp. 283–424, Mar. 2016, doi: 10.1561/04000000074.
- [18] H. Fang and Q. Qian, “Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning,” *Futur. Internet*, vol. 13, no. 4, p. 94, Apr. 2021, doi: 10.3390/fi13040094.
- [19] Y. Ameur, “Exploring the Scope of Machine Learning using Homomorphic Encryption in IoT/Cloud,” HESAM Université, 2023. [Online]. Available: <https://theses.hal.science/tel-04587371>
- [20] R. Lavin, X. Liu, H. Mohanty, L. Norman, G. Zaarour, and B. Krishnamachari, “A Survey on the Applications of Zero-Knowledge Proofs,” *arXiv*, Aug. 01, 2024. [Online]. Available: <http://arxiv.org/abs/2408.00243>
- [21] M. Andersson and A. H. C. Ng, “Parameter tuning evolutionary algorithms for runtime versus cost trade-off in a cloud computing environment,” *Simul. Model. Pract. Theory*, vol. 89, pp. 195–205, Dec. 2018, doi: 10.1016/j.simpat.2018.10.003.
- [22] Q. Li and R. Zong, “CAT: A GPU-Accelerated FHE Framework with Its Application to High-Precision Private Dataset Query,” *arXiv*, Mar. 28, 2025. [Online]. Available: <http://arxiv.org/abs/2503.22227>
- [23] A. U. Khandait and T. R. Pattanshetti, “Lightweight Authentication System Based on Zero Knowledge Proof for IoT Devices,” in *2024 Global Conference on Communications and Information Technologies (GCCIT)*, Oct. 2024, pp. 1–5. doi: 10.1109/GCCIT63234.2024.10862207.
- [24] Z. Xing *et al.*, “Zero-Knowledge Proof-Based Verifiable Decentralized Machine Learning in Communication Network: A Comprehensive Survey,” *IEEE Commun. Surv. Tutorials*, pp. 1–1, Mar. 2025, doi: 10.1109/COMST.2025.3561657.
- [25] C. Gouert, D. Mouris, and N. Tsoutsos, “SoK: New Insights into Fully Homomorphic Encryption Libraries via Standardized Benchmarks,” *Proc. Priv. Enhancing Technol.*, vol. 2023, no. 3, pp. 154–172, Jul. 2023, doi: 10.56553/popets-2023-0075.
- [26] V. Keršič, S. Karakatić, and M. Turkanović, “On-chain zero-knowledge machine learning: An overview and comparison,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 9, p. 102207, Nov. 2024, doi: 10.1016/j.jksuci.2024.102207.
- [27] S. Gorantala, R. Springer, and B. Gipson, “Unlocking the Potential of Fully Homomorphic Encryption,” *Communications of the ACM*, 2023. <https://cacm.acm.org/research/unlocking-the-potential-of-fully-homomorphic-encryption/>
- [28] J. Fan and F. Vercauteren, “Somewhat Practical Fully Homomorphic Encryption,” *Cryptology ePrint Archive*, 2012. [Online]. Available: <https://eprint.iacr.org/2012/144>
- [29] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Scalable Zero Knowledge via Cycles of Elliptic Curves,” *Cryptology ePrint Archive*, 2014. [Online]. Available: <https://eprint.iacr.org/2014/595>
- [30] H. Wu, “libsark: a C++ library for zkSNARK proofs,” *GitHub*, 2014. <https://github.com/scipr-lab/libsark>