Federated Learning Using Fully Homomorphic Encryption

(Enhancing Data Privacy in Collaborative Machine Learning)

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

in

Cryptology and Security

submitted by

Sk Golam Kuddus
(Roll No. CrS2318)

Under the esteemed guidance of

Captain Manish Khanna
HOD
WESEE, Ministry of Defence

Dr. Mriganka Mandal
Cryptology and Security Research Unit
Indian Statistical Institute, Kolkata

Indian Statistical Institute
Kolkata – 700108, India

July, 2025

DECLARATION

I, **Sk Golam Kuddus (Roll No: CrS2318)**, hereby declare that, this report entitled **"Federated Learning Using Fully Homomorphic Encryption"** submitted to Indian Statistical Institute, Kolkata towards the fulfilment of the requirements for the degree of **Master of Technology in CSRU**, is an original work carried out by me under the supervision of **Captain Manish Khanna, Lt. Cdr. Keval Krishan and Dr. Mriganka Mandal** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.

Kolkata - 700 108

July 2025

Sk Golam Kuddus

Certificate

This is to certify that the work contained in this project report entitled **"Federated Learning Using Fully Homomorphic Encryption"** submitted by **Sk Golam Kuddus (Roll No. CrS2318)** to the Indian Statistical Institute, Kolkata towards the fulfilment of the requirements for the degree of **Master of Technology in CSRU** has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

Dr. Mriganka Mandal
Assistant Professor
Cryptology and Security Research Unit
Indian Statistical Institute
Kolkata-700108,INDIA

ACKNOWLEDGEMENTS

The completion of this master's thesis would not have been possible without the invaluable support and guidance of numerous individuals and institutions, to whom I owe my deepest gratitude.

I extend my most sincere appreciation to my esteemed supervisors at WESEE and the Indian Statistical Institute, Kolkata, whose scholarly expertise, unwavering dedication, and visionary guidance have been instrumental in shaping this research. Their constructive feedback, intellectual rigor, and continuous mentorship throughout this academic journey have profoundly enriched both my understanding of the subject matter and my development as a researcher.

My heartfelt thanks go to my colleagues and friends, whose steadfast encouragement, thoughtful discussions, and unwavering moral support have sustained me throughout the challenging phases of this research. Their camaraderie and intellectual companionship have been a source of inspiration and motivation.

Finally, I acknowledge with profound gratitude all those who, directly or indirectly, contributed to making this academic endeavor a success. Any errors or omissions in this work remain entirely my own responsibility.

Kolkata - 700 108

Sk Golam Kuddus

July 2025

Sk Golam Kuddus

Abstract

Traditional machine learning approaches require centralizing data for training, which raises significant privacy concerns when dealing with sensitive information. Federated learning (FL) addresses this by keeping data local and enabling multiple users to collaboratively train a shared machine learning model. In spite of this, FL remains vulnerable to inference attacks, as sensitive information can still be extracted from the model's learned parameters. While traditional privacy-enhancing techniques such as differential privacy introduce noise to model updates to obscure individual data points, they often present a fundamental trade-off between privacy and utility. Furthermore, these approaches still carry risks of data leakage if implementation is flawed or adversaries possess sophisticated attack capabilities. To address these limitations, we propose a novel federated learning framework that integrates Homomorphic Encryption and Secret Sharing to provide robust privacy guarantees. Our approach ensures that both raw data and model updates remain confidential throughout the learning process. By enabling computations on encrypted data, our framework allows the aggregation server to perform model updates without ever accessing plaintext information. We evaluate our framework on the CIFAR10 and MNIST handwritten digit classification dataset, demonstrating that it achieves comparable accuracy to traditional FL while providing substantially stronger privacy protections. Performance analysis shows that our approach introduces acceptable computational overhead, making it practical for real-world applications. The framework is especially valuable in sensitive domains such as healthcare, defence, finance, and personal monitoring systems where data confidentiality is paramount. Our contribution advances the state of the art in privacy-preserving machine learning by offering a comprehensive solution that maintains utility while providing cryptographic privacy guarantees that protect against both honest-but-curious aggregators and potential adversaries.

Keywords: Federated Learning, Homomorphic Encryption, CKKS, Privacy-Preserving Machine Learning, Threshold Cryptography, Secure Aggregation.

Contents

1 Introduction **6**

2 Related Work **8**

3 Preliminaries **9**
3.0.1 Federated Learning 9
3.0.2 Gradient Leakage Attacks in Federated Learning 11
3.0.3 Homomorphic Encryption: Foundations and Evolution 13

4 Workflow of the System **17**
4.0.1 Clients 18
4.0.2 Server 21
4.0.3 Security Analysis 23
4.0.4 Computational Optimization 24

5 Experimental Framework **25**
5.0.1 Experimental Datasets 25
5.0.2 Model Architecture 25
5.0.3 FHE Context Selection 25
5.0.4 Data Partitioning for Experiments 27
5.0.5 Performance and Overhead Analysis 27
5.0.6 Client Participation Impact on Model Performance 28

6 Summary of Contributions **30**

7 conclusion **31**
7.1 Limitations 31
7.2 Future Research Directions 31

Chapter 1

Introduction

The proliferation of edge computing devices and the exponential growth of decentralized data generation have fundamentally transformed the landscape of machine learning. As organizations increasingly recognize data as a strategic asset, the traditional paradigm of centralizing information for model training faces mounting challenges related to privacy regulations, bandwidth limitations, and computational sovereignty. This paradigm shift has catalyzed the development of distributed learning architectures that respect data locality while enabling collaborative intelligence.

Federated learning represents a revolutionary approach to this challenge, enabling multiple stakeholders to jointly train machine learning models without compromising data privacy or ownership. By keeping sensitive information at its source and sharing only model updates, this framework addresses critical concerns in sectors where data confidentiality is paramount—from healthcare institutions protecting patient records to financial organizations safeguarding transaction data. However, despite its privacy-preserving architecture, federated learning remains vulnerable to sophisticated inference attacks that can reconstruct sensitive information from seemingly innocuous gradient updates.

Google demonstrated the practical viability of federated learning in 2016 through their mobile keyboard application, where the challenge of personalizing text predictions without compromising user privacy catalyzed a fundamental shift in distributed machine learning architectures. This real-world implementation demonstrated that millions of devices could collaboratively refine a shared predictive model while maintaining complete data sovereignty—each device contributing learned patterns without exposing individual typing behaviors. The success of this deployment, processing contributions from diverse global users while preserving their conversational privacy, validated federated learning as more than a theoretical construct. This pioneering application illuminated the path for our current research, showing how cryptographic enhancements could further strengthen privacy guarantees in such distributed systems.

The integration of cryptographic techniques, particularly homomorphic encryption, emerges as a promising solution to fortify federated learning against these vulnerabilities. Among various homomorphic encryption schemes, the Cheon-Kim-Kim-Song (CKKS) framework stands out for its unique capability to perform approximate arithmetic operations on encrypted real numbers—a critical requirement for machine learning computations. This approximate homomorphic encryption scheme enables secure aggregation of model updates while maintaining computational efficiency, making it particularly suitable for federated learning environments where both privacy and performance are essential.

This thesis investigates the synergistic integration of CKKS homomorphic encryption within federated learning frameworks to achieve robust privacy guarantees without sacrificing model utility. We address the fundamental question of whether cryptographically secure federated learning can be practically deployed in real-world scenarios, considering the inherent trade-offs between security, accuracy, and computational overhead. Our research contributes to the

growing body of knowledge in privacy-preserving machine learning by demonstrating that strong cryptographic protection can coexist with efficient distributed training protocols.

The significance of this work extends beyond theoretical contributions, as the demand for privacy-preserving collaborative learning continues to surge across industries. From enabling cross-institutional medical research without sharing patient data to facilitating secure financial fraud detection across competing banks, the applications of our framework address pressing societal needs. As regulatory frameworks like the Digital Personal Data Protection Act (DPDPA) of 2023, the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA) impose stricter data protection requirements, our approach provides a technically sound solution that aligns with both legal compliance and ethical data stewardship principles.

Chapter 2

Related Work

The field of privacy-preserving machine learning has seen rapid advancement, driven by the need to train models on sensitive, decentralized data. Federated Learning (FL) was introduced as a foundational paradigm to address this challenge by training on local data without centralizing it. It has found applications in diverse domains such as smart cities [2] and healthcare [3]. However, the standard FL framework is not immune to privacy breaches. The core assumption that sharing model updates is inherently safe has been challenged by the discovery of sophisticated inference attacks.

A critical vulnerability in standard FL is "gradient leakage," where sensitive training data can be reconstructed from the shared model gradients. The Deep Leakage from Gradients (DLG) attack demonstrated the feasibility of recovering both training images and their labels from a single gradient update [18]. Subsequent research, such as the improved Deep Leakage from Gradients (iDLG) attack, refined these techniques by analytically extracting ground-truth labels, making the reconstruction more accurate and efficient [6]. These attacks underscore the insufficiency of data locality alone and motivate the need for stronger, cryptographic protections.

To counter these threats, researchers have turned to cryptographic methods, with Homomorphic Encryption (HE) emerging as a leading solution. HE allows for computation directly on encrypted data, making it possible for an untrusted server to aggregate model updates without accessing the plaintext gradients. The concept originated with the idea of "privacy homomorphisms" [8] and was fully realized with Gentry's breakthrough construction of a fully homomorphic encryption scheme [9]. The evolution of HE has produced various schemes, including BGV [20], GSW [21], and DGHV [19]. However, the Cheon-Kim-Kim-Song (CKKS) [14] scheme is particularly well-suited for machine learning, as it operates on approximate real numbers, aligning with the inherent noise tolerance of gradient-based optimization [7]. A comprehensive overview of these schemes can be found in surveys on the topic [15].

Recent work has focused on integrating these cryptographic tools into practical FL frameworks. Some have explored combining HE with other technologies, such as using blockchain to further decentralize trust and record-keeping in FL for healthcare data [22]. Others have looked toward future threats by developing post-quantum secure FL systems to ensure long-term data confidentiality [16].

While these approaches advance the state of the art, this thesis presents a comprehensive framework that synergistically combines the efficiency of the CKKS scheme with a robust, decentralized trust model. Unlike systems that may rely on a trusted entity for key management, our work implements a Distributed Key Generation (DKG) and Threshold Decryption protocol. This design eliminates the single point of failure associated with a central key server and distributes trust among the participating clients. By providing a complete workflow—from collaborative key generation to secure aggregation and threshold decryption—and empirically validating it on the CIFAR-10 and MNIST datasets, our work demonstrates a practical and cryptographically secure solution to the privacy challenges inherent in federated learning.

Chapter 3

Preliminaries

This section provides the necessary background and mathematical foundations for understanding federated learning and homomorphic encryption, particularly the CKKS scheme[14] used in this thesis.

3.0.1 Federated Learning

Federated learning aims to build global AI models for multiple parties with diverse interests. Instead of collecting and combining data from different locations and gathering them in a central location, FL processes data from the place where it ensures the security of sensitive data and models.

Characteristics of federated learning

Through our comprehensive analysis of federated learning deployments, we identified several architectural characteristics that distinguish this paradigm from conventional distributed training approaches. The multi-node collaborative structure we implemented creates a unique computational ecosystem where geographically dispersed participants contribute to a shared learning objective without centralizing their data. Our framework employs a coordinator-participant model that delegates specific computational tasks while maintaining data sovereignty at each node. The cross-organizational nature of our system enables entities with potentially competing interests to collaboratively develop AI models. This aspect particularly influenced our security design, as participants require guarantees that their contributions cannot be reverse-engineered by other participants or the coordinator. Our empirical observations revealed extreme data heterogeneity as a defining challenge. Unlike traditional distributed systems where data is randomly partitioned, federated environments exhibit natural clustering based on participant characteristics. This non-IID distribution significantly impacted our encryption strategy, as we needed to preserve statistical properties while preventing information leakage. We architected our system with a pseudo-decentralized structure that balances coordination efficiency with participant autonomy. While the parameter server facilitates aggregation, it operates without access to raw data or unencrypted gradients, enforcing a trust boundary that our homomorphic encryption maintains throughout the learning process.

Categorization of federated learning

Federated learning architectures can be broadly categorized into two fundamental paradigms based on how data is distributed across participating entities. Understanding these distinctions is crucial for selecting appropriate algorithms and privacy-preserving techniques.

- **Horizontal federated learning (HFL):**

Also known as sample-based or homogeneous federated learning, it represents the most common deployment scenario. In this configuration, participating clients share identical feature spaces but possess different data samples. Consider multiple hospitals collaborating to train a diagnostic model—each institution collects the same types of medical measurements, such as blood pressure, temperature, and lab results, but for different patient populations. The data partition occurs horizontally across the sample dimension, where each participant contributes unique instances with consistent attributes. This architecture naturally arises in scenarios involving mobile devices, IoT sensors, or distributed organizations performing similar functions. The primary challenge in HFL stems from the non-IID nature of data distributions across clients, as local datasets often reflect geographic, demographic, or behavioral variations specific to each participant.

- **Vertical federated learning (VFL):** Also known as feature-based federated learning, it addresses scenarios where participants hold different features for overlapping sets of data samples. This architecture is particularly relevant in cross-organizational collaborations where different entities possess complementary information about the same subjects. For instance, a bank and an e-commerce platform might have different attributes—financial history versus purchasing behavior—for the same customer base, enabling them to jointly build more comprehensive predictive models without sharing their proprietary data. VFL introduces unique technical challenges, including the need for secure entity matching protocols to identify common samples across organizations while preserving privacy, and the complexity of performing computations across distributed features without revealing individual values. Applications of VFL span cross-industry risk assessment, multi-modal healthcare analysis where different institutions hold various types of medical data, and collaborative financial fraud detection systems.

In this thesis, we primarily focus on horizontal federated learning scenarios, as they represent the predominant use case in practical deployments and pose unique challenges for homomorphic encryption integration.

The Federated Averaging Algorithm

Federated Averaging (FedAvg) stands as the foundational algorithm in horizontal federated learning, employing a coordinator-participant framework that has become the de facto standard for distributed model training. This algorithm elegantly balances computational efficiency with communication constraints through its iterative approach to collaborative learning.

The algorithm begins with an initialization phase where the central coordinator establishes a global model $\mathcal{W}$ with randomly initialized parameters. The overarching goal is to discover optimal model parameters that minimize the aggregate loss across all participating nodes. Mathematically, this optimization problem seeks to minimize the global objective function $L(\mathcal{W})$, expressed as the weighted average of individual participant loss functions: $\min_{\mathcal{W}} L(W) = \frac{1}{K}\sum_{k=1}^{K} L_k(\mathcal{W})$, where K denotes the number of participants and $L_k(\mathcal{W})$ represents the loss function specific to participant k's local dataset.

During each communication round t, selected participants independently optimize their local models using their private data. Each participant k computes updates to the model parameters through gradient-based optimization, typically employing stochastic gradient descent (SGD). The local update rule follows: $\Delta\mathcal{W}_k^t = -\eta\nabla L_k(\mathcal{W})$, where η represents the learning rate controlling the magnitude of parameter updates, and $\nabla L_k(\mathcal{W})$ denotes the gradient of the local loss function with respect to the model parameters. Participants perform multiple iterations of local training, allowing the model to adapt to their specific data distributions before communicating with the coordinator.

Upon completion of local training, participants transmit their model updates to the central coordinator, which performs the critical aggregation step. The coordinator combines these

distributed updates to form an improved global model for the subsequent round:

$$\mathcal{W}^{t+1} = \mathcal{W}^t + \frac{1}{M}\sum_{k=1}^{M} \Delta \mathcal{W}_k^t$$

where M is the number of participants selected in the current round. This averaging mechanism ensures that the global model incorporates knowledge from all participants while respecting the privacy of their underlying data. The iterative process of local computation, communication, and aggregation continues across multiple rounds until the global model achieves satisfactory convergence or reaches a predetermined number of iterations. This approach effectively transforms the distributed optimization problem into a series of local optimization tasks coordinated through periodic synchronization, making it particularly suitable for scenarios with limited communication bandwidth and privacy constraints.

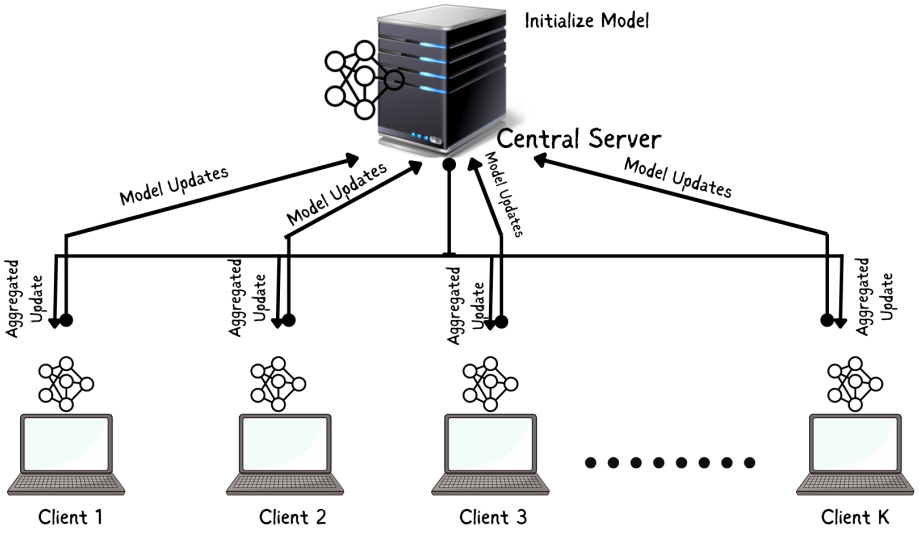


Figure 3.1: Basic diagram of standard, centralized FL

3.0.2 Gradient Leakage Attacks in Federated Learning

Our security analysis of federated learning protocols revealed a critical vulnerability that undermines its privacy promises: the gradient updates themselves encode sufficient information to reconstruct training data. Through our investigation, we discovered that an adversary with access to these updates can exploit the deterministic relationship between data and its corresponding gradients. This finding fundamentally challenges the assumption that data locality alone provides adequate privacy protection. Our experiments confirmed that sophisticated optimization techniques can inverse-engineer the gradient computation process, extracting sensitive information that federated learning was designed to protect. This vulnerability serves as the primary motivation for integrating homomorphic encryption into our framework.

Attack Mechanism and Threat Model

Gradient leakage attacks exploit the mathematical relationship between training data and the gradients computed during backpropagation. Consider a scenario where an adversary intercepts gradient updates transmitted from a client to the central server. The attacker, possessing knowledge of the model architecture F and the transmitted gradients ∇W, can formulate an optimization problem to reconstruct the original training data.

The attack proceeds by generating synthetic data samples (x', y') and computing their corresponding gradients $\nabla W'$ through the known model. The adversary then iteratively refines these synthetic samples to minimize the distance between the computed gradients and the intercepted real gradients:

$$\min_{x',y'} ||\nabla W' - \nabla W||^2 \tag{3.1}$$

When this distance becomes sufficiently small, the synthetic data (x', y') closely approximates the original training samples (x, y), effectively breaching the privacy of the federated learning participant.

Evolution of Attack Techniques

The seminal Deep Leakage from Gradients (DLG) attack, proposed by Zhu et al. (2019)[18], demonstrated the feasibility of reconstructing both training inputs and their corresponding labels from gradient information alone. However, DLG exhibited limitations, particularly in accurately recovering labels during the reconstruction process, often generating incorrect label assignments.

To overcome these limitations, researchers proposed improved Deep Leakage from Gradients (iDLG)[6],which introduced an analytical approach for extracting ground-truth labels. iDLG leverage the relationship between the probability distribution over output labels and the gradients of the corresponding output neurons to infer the correct label. In the optimization process:

$$x^*, y^* \triangleq \arg\min_{x',y'} \left\|\nabla W' - \nabla W\right\|^2 = \arg\min_{x',y'} \left\| \frac{\partial (F(x', W'))_{y'}}{\partial W} - \nabla W \right\|^2, \tag{3.2}$$

they only need to update x' to realize the reconstruction of x with higher accuracy. By analytically extracting the correct label from the cross-entropy loss gradients, iDLG[6] only needs to optimize for x', significantly improving reconstruction accuracy and computational efficiency. This advancement demonstrated that certain architectural properties could be exploited to facilitate more effective privacy breaches.

Critical Factors in Attack Success

The effectiveness of gradient leakage attacks depends on several interconnected factors:

Batch Size Impact: Single-sample gradients provide the most information for reconstruction, as the gradient directly corresponds to one data point. Larger batch sizes introduce ambiguity through gradient averaging, though recent research shows that even moderate batch sizes remain vulnerable to sophisticated attack variants.

Model Complexity: The relationship between model parameters and input dimensionality affects reconstruction difficulty. Models with fewer parameters than input features create an under-determined system, potentially enabling unique solutions for data recovery.

Gradient Magnitude: Early training phases typically produce larger gradients containing more information about individual samples. As training progresses and the model converges, gradient magnitudes decrease, offering marginal protection against reconstruction attempts.

Data Properties: Structured data with known constraints, such as natural images or text following grammatical rules, can be more readily reconstructed using domain-specific priors and regularization techniques.

Understanding these factors is essential for both designing effective defenses and evaluating the real-world risk posed by gradient leakage attacks in FL systems.

Implications for Privacy-Preserving Federated Learning

These attacks reveal that federated learning's fundamental assumption—that keeping data local ensures privacy—is insufficient without additional protective measures. The ability to reconstruct training data from gradients necessitates stronger privacy guarantees through cryptographic or noise-based defenses.

The vulnerability exposed by gradient leakage attacks motivates our investigation into homomorphic encryption as a defense mechanism. By ensuring that gradient computations and aggregations occur entirely in the encrypted domain, we prevent any party, including the central server, from accessing raw gradient values that could be exploited for data reconstruction. This cryptographic approach provides provable security guarantees against gradient leakage while maintaining the collaborative learning capabilities essential to federated learning systems.

3.0.3 Homomorphic Encryption: Foundations and Evolution

Our investigation into privacy-preserving federated learning naturally leads us to examine homomorphic encryption—a cryptographic technique that fundamentally alters how we approach secure computation. Unlike traditional encryption methods that create a barrier between data protection and data utility, homomorphic schemes maintain computational capabilities even when data remains in its encrypted form. This unique property aligns perfectly with our federated learning requirements, where model updates must be aggregated without exposing individual contributions. Through our analysis, we discovered that this cryptographic approach offers a mathematically rigorous solution to the gradient leakage vulnerabilities identified in standard federated protocols.

The theoretical underpinnings of our cryptographic approach stem from early work in the late 1970s, when researchers first envisioned encryption systems that could preserve computational relationships. Rivest and colleagues [8] introduced the notion of privacy homomorphisms—a concept that remained largely theoretical until recent computational advances made it practical for our federated learning applications. Their observation that certain algebraic structures could be maintained through encryption transformation laid the groundwork for what would eventually become a crucial component in our privacy-preserving framework. However, these early systems supported only limited operations, constraining their applicability to our complex machine learning scenarios.

A pivotal advancement emerged in 2009 that directly influenced our approach to securing federated learning. Gentry's construction [9] of a fully homomorphic system—capable of supporting arbitrary computational depth—offered the theoretical foundation we needed for processing complex neural network operations on encrypted gradients. While Gentry's original framework suffered from computational limitations that would have made our federated learning implementation impractical, it established the mathematical feasibility of our proposed approach. The scheme's reliance on ideal lattice problems provided security guarantees that align with our threat model, though significant optimization was required before such systems could handle the computational demands of modern machine learning workloads.

Subsequent research focused on achieving practical efficiency while maintaining security guarantees. Gentry, Halevi, and Smart [11] introduced significant optimizations in 2012, developing FHE schemes based on the ring learning with errors (RLWE) problem that dramatically reduced ciphertext expansion and computational complexity. These improvements, combined with techniques like modulus switching and bootstrapping optimization, brought homomorphic encryption closer to practical viability for specialized applications.

The evolution of homomorphic encryption has produced several distinct scheme families, each optimized for different use cases and offering various trade-offs between functionality, efficiency, and security:

BGV (Brakerski-Gentry-Vaikuntanathan) [10] scheme leverages polynomial rings and

modulus switching techniques to manage noise growth, making it particularly suitable for circuits with predetermined multiplicative depth.

GSW (Gentry-Sahai-Waters) [21] introduced a novel approach using matrix-based ciphertexts that simplified bootstrapping and enabled more efficient homomorphic operations on boolean circuits.

DGHV (van Dijk-Gentry-Halevi-Vaikuntanathan) [19] scheme operates over the integers rather than polynomial rings, offering conceptual simplicity while maintaining security based on the approximate greatest common divisor problem.

Among the various homomorphic schemes we evaluated for our federated learning framework, the CKKS construction [14] emerged as uniquely suited to our requirements. Our selection was driven by a critical insight: machine learning algorithms inherently tolerate approximation errors, making exact arithmetic unnecessary and computationally wasteful. The CKKS framework[14] exploits this tolerance by encoding real numbers as polynomials and performing approximate operations that maintain sufficient precision for gradient computations while dramatically reducing computational overhead. This alignment between the scheme's approximate nature and the inherent noise tolerance of stochastic gradient descent makes it the optimal choice for our privacy-preserving federated learning implementation.

Formal Framework and Properties

From a mathematical perspective, a homomorphic encryption scheme [15] is formally defined as a tuple of algorithms operating over three fundamental algebraic structures: the message space $\mathcal{M}$ containing all possible plaintext values, the ciphertext space $\mathcal{C}$ comprising all valid encrypted representations, and the key space $\mathcal{K}$ containing cryptographic key material.

The scheme consists of four probabilistic polynomial-time algorithms:

Key Generation (KeyGen): This randomized algorithm generates cryptographic keys, typically producing a public-private key pair (pk, sk) where $sk \in \mathcal{K}$ is selected uniformly at random. The public key enables encryption and homomorphic operations, while the secret key permits decryption.

Encryption (Enc): Given a plaintext message $m \in \mathcal{M}$ and the public key pk, this algorithm produces a ciphertext $c \in \mathcal{C}$ through the transformation $c = \text{Enc}(pk, m)$. The encryption process incorporates randomness to ensure semantic security.

Decryption (Dec): Using the secret key sk, this deterministic algorithm recovers the original message from a ciphertext: $m = \text{Dec}(sk, c)$. Correctness requires that for all valid key pairs and messages, $\text{Dec}(sk, \text{Enc}(pk, m)) = m$.

Homomorphic Evaluation (Eval): The defining feature of homomorphic encryption, this algorithm takes as input the public key pk, a collection of ciphertexts $\{c_1, c_2, \ldots, c_n\}$, and a circuit description f, producing a new ciphertext $c' = \text{Eval}(pk, f, c_1, \ldots, c_n)$ such that $\text{Dec}(sk, c') = f(m_1, \ldots, m_n)$, where $m_i = \text{Dec}(sk, c_i)$.

The homomorphic properties manifest through structure-preserving operations in the ciphertext space:

Additive Homomorphism: For any plaintexts $m_1, m_2 \in \mathcal{M}$ with corresponding ciphertexts $c_1, c_2 \in \mathcal{C}$, there exists an operation $\oplus$ in the ciphertext space such that:

$$\text{Dec}(sk, c_1 \oplus c_2) = m_1 + m_2 \tag{3.3}$$

This property enables secure addition of encrypted values without intermediate decryption.

Multiplicative Homomorphism: Similarly, there exists an operation $\otimes$ in the ciphertext space satisfying:

$$\text{Dec}(sk, c_1 \otimes c_2) = m_1 \times m_2 \tag{3.4}$$

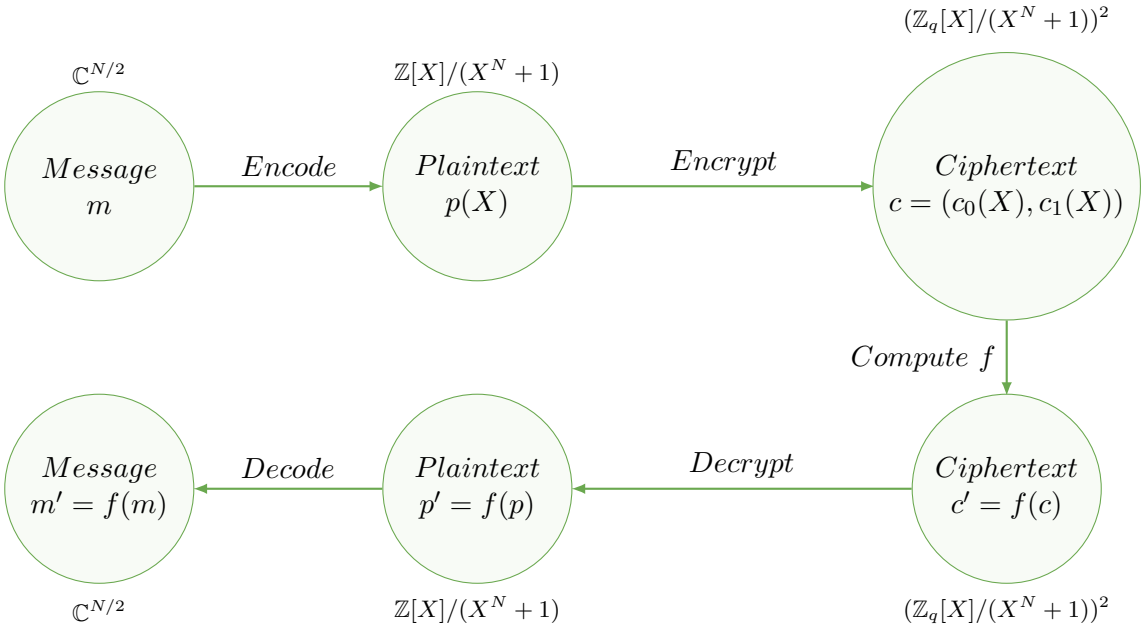
Schemes supporting both additive and multiplicative homomorphisms with unbounded depth are termed **fully homomorphic**, while those supporting limited operations are **partially or somewhat homomorphic**.

These algebraic properties enable computation directly on encrypted data [15], facilitating secure outsourcing of sensitive computations to untrusted environments—a capability fundamental to privacy-preserving machine learning and federated learning applications. However, practical deployment faces challenges including substantial computational overhead, ciphertext expansion, and noise management requirements that must be carefully balanced against security and functionality needs.

The CKKS scheme[14] adopted in this thesis addresses these practical concerns by supporting approximate arithmetic operations, trading exact precision for dramatically improved efficiency—a compromise well-suited to machine learning workloads where approximate gradients and model parameters are inherently tolerated by optimization algorithms.

The CKKS Homomorphic Encryption Scheme

The Cheon-Kim-Kim-Song (CKKS)[14] scheme represents a paradigm shift in homomorphic encryption design, prioritizing practical efficiency for real-number computations over exact arithmetic precision. This scheme specifically addresses the computational requirements of applications like machine learning, signal processing, and statistical analysis where approximate results suffice.



CKKS Scheme Overview

CKKS is a lattice-based FHE scheme. The above figure gives a high-level overview of the CKKS scheme. In CKKS, the plaintext and ciphertext spaces include elements of the polynomial ring$R_q = \mathbb{Z}_q[x]/f(x)$, where q is an integer called the coefficient modulus and $f(x)$ is a polynomial known as the polynomial modulus. Elements of R_q polynomials with integer coefficients are bounded by q. The prevailing selection for the function $f(x)$ in scholarly works is $f(x) = x^N + 1$, where N (referred to as the ring dimension or polynomial modulus degree) is an integer that is a power of 2. The message $m \in \mathbb{C}^{N/2}$ is firstly encoded into a plaintext polynomial $p(X) \in R_q$ and then encrypted into two ciphertext polynomials $(c_0(X), c_1(X)) \in R_q^2$. Within the ciphertext domain, CKKS can perform homomorphic addition, multiplication, and rotation operations.

To utilize CKKS effectively, especially in the context of training models using federated learning, it is essential to carefully select cryptographic parameters that balance security, computational cost, and precision. These parameters influence the scheme's performance and ability to support operations like addition and multiplication across multiple layers of a neural network.

Polynomial Modulus Degree (N): This parameter determines the ring dimension and directly affects both the security level and the number of slots for SIMD operations. Typically, it N is a power of two, and commonly used values are $2^{13} = 8192$, $2^{14} = 16384$, or $2^{15} = 32768$. Larger N increases computational and memory costs but allow more noise budget and higher security.

Coefficient Modulus (q): The ciphertext modulus q is a product of primes (usually 60-bit, 30-bit, or 40-bit primes) and defines the total noise budget. Larger q allows for deeper computations but increases ciphertext size and processing time. It is selected based on the required multiplicative depth$d$, since each multiplication consumes noise. The modulus chain should be long enough to support d levels with rescaling.

Scaling Factor (Δ): This determines the precision of encoded real/complex numbers. A typical value is 2^{40} or2^{50}, balancing approximation error with numerical stability. After each multiplication, a rescaling step divides Δ to manage ciphertext growth.

Security Level: All parameter choices must meet at least 128-bit post-quantum security. This is typically ensured by adhering to NIST recommendations for parameter sets, verified using tools such as the LWE estimator.

Packing (Slots): CKKS allows encoding of multiple values into one ciphertext using SIMD. For polynomial modulus degree$N$, the scheme supports up to $N/2$ complex slots, enabling efficient vectorized operations during federated learning.

Chapter 4

Workflow of the System

In this section we present our privacy-preserving framework integrated with CKKS homomorphic encryption and secret sharing. We consider a horizontal-federated learning scenario implemented within a client-server architecture. The model training takes place in an iterative way. The training process ends after a sufficient number of iterations or when the training effect of the model meets the expectations. Learning data comes from various clients. A central server first initializes a model and sends it to selected clients. In each iteration, each client uses its own private data to train that model, encrypts their trained model or gradients using homomorphic encryption, and then uploads the ciphertext to the central (aggregation) server. The server aggregates the ciphertext information uploaded by the clients. Later, the participants download the aggregated ciphertext information from the server, decrypt the ciphertext using threshold decryption, and obtain the aggregated model.

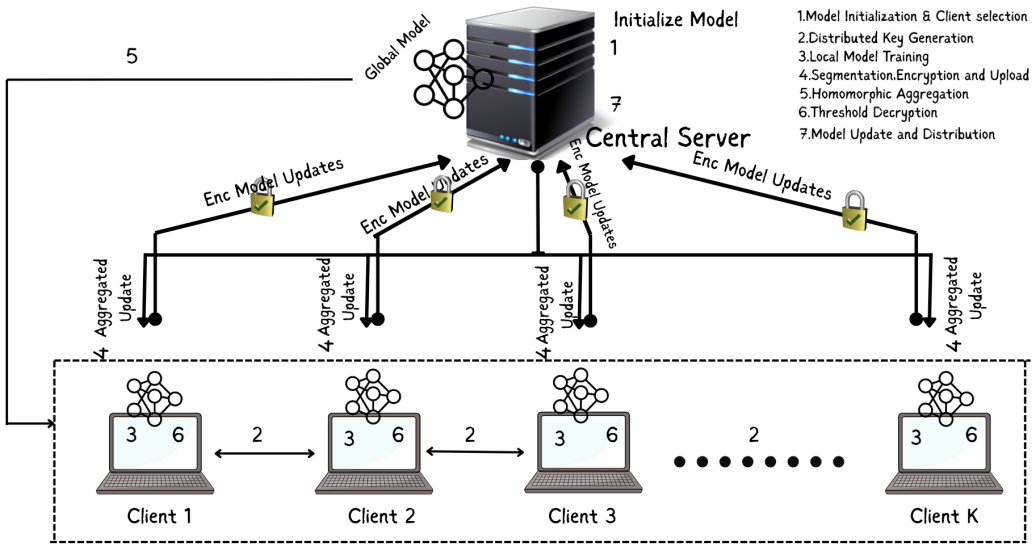


Figure 4.1: End-to-End Workflow of Federated Learning with Homomorphic Encryption and Threshold Decryption

Before implementing encryption, each participant needs to generate the associated key. The specific process is as follows:A common random polynomial a is sampled uniformly from the ring R_Q and distributed to all M parties. Each participating party P_i will first generate its private secret share s_i and a corresponding public component b_i, which is then broadcast to a combiner. The combiner calculates the final shared public key PK by summing these components and

makes it publicly known. Each party can then use this public key to perform homomorphic encryption on their model updates.

After the server has aggregated the encrypted updates, a set of participating parties must collaborate to decrypt it. To do this, each party computes a partial decryption using their secret share and sends it to the combiner, which then aggregates these parts to reconstruct the final plaintext result.

Algorithm 1: CKKS Encryption Context Setup

Require:
- Security parameter λ
- Polynomial modulus degree N (e.g., 2^{13}, 2^{14}, 2^{15})
- Coefficient modulus $\mathcal{Q} = (q_0, q_1, \ldots, q_t)$

Ensure: CKKS encryption context `ctx`

1: Define polynomial modulus: $\phi(X) = X^N + 1$
2: Choose coefficient modulus: $\mathcal{Q} = (q_0, q_1, \ldots, q_t)$
3: Set scaling factor: $\Delta = 2^{40}$ (adjustable for precision)
4: Generate keys:
- Public key `Pk`
- Secret key `Sk`
- Evaluation key `evk` (including relinearization and rotation keys)

In distributed settings, `Pk` and `evk` are generated via collaborative DKG (see Algorithm 3).

5: **return** `ctx` $= (N, \mathcal{Q}, \Delta, \texttt{Pk}, \texttt{Sk}, \texttt{evk})$

4.0.1 Clients

In our framework, the clients are the entities that both provide data for training and benefit from the final trained model. The role of each participant is governed by a strict cryptographic protocol to ensure data privacy and distributed trust throughout the federated learning process. A complete round of training involves the following steps executed by each participant:

The process begins with the client initializing a local model, $\mathcal{W}'$, using the current global model parameters, $\mathcal{W}$, received from the central server. The client then trains this model for a predetermined number of local epochs, $\mathcal{E}$, using its own dataset, $\mathcal{D}$.Finally, the client returns the updated local model parameters $\mathcal{W}'$, which are then ready for the encryption and aggregation phase.

Algorithm 2: Local Client Training

Require:
- Local client dataset $\mathcal{D}$
- Current global model parameters $\mathcal{W}$
- Number of local epochs $\mathcal{E}$

Ensure: Updated local model parameters $\mathcal{W}'$

Initialize local model: $\mathcal{W}' \leftarrow \mathcal{W}$
for epoch $e = 1$ to $\mathcal{E}$ **do**
  for each mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
    Compute loss: $\mathcal{L}(\mathcal{W}', \mathcal{B})$
    Compute gradient: $\nabla\mathcal{L}(\mathcal{W}', \mathcal{B})$
    Update weights: $\mathcal{W}' \leftarrow \mathcal{W}' - \eta\nabla\mathcal{L}(\mathcal{W}', \mathcal{B})$ *// η: learning rate*
  end for
end for
return $\mathcal{W}'$

Each participant P_i collaborates in a Distributed Key Generation (DKG) protocol. Instead of interacting with a key server, each party generates a private secret share s_i, and a public component (b_i, a). b_i is broadcast to a combiner. This process results in a shared public key, PK, which is used for encryption by all participants. Each participant securely retains its own secret share, s_i, for the decryption phase.

Algorithm 3: Distributed Key Generation (DKG) for CKKS

Require:
- CKKS encryption context `ctx` (includes N, $\mathcal{Q}$)
- Number of parties M
- Decryption threshold T (where $1 < T \leq M$)

Ensure:
- Shared public key $PK = (b, a)$ in $R_{\mathcal{Q}} = \mathbb{Z}_{\mathcal{Q}}[X]/(X^N + 1)$
- Each party holds a secret share $s_i \in R = \mathbb{Z}[X]/(X^N + 1)$

1: A common random polynomial a is sampled uniformly from the ring $R_{\mathcal{Q}}$ and distributed to all M parties.
2: Initialize:

$$b_{\text{agg}} \leftarrow 0 \mod \mathcal{Q}$$

3: **for** each party P_i, $i = 1$ to M **do**
4: Generate secret share $s_i \in R$ with small coefficients (e.g., ternary or Gaussian)
5: Sample error polynomial $e_i \in R$ with small coefficients
6: Compute $b_i = (-a \cdot s_i + e_i) \mod \mathcal{Q}$
7: Broadcast b_i to all other parties or a designated combiner
8: **end for**
9: Aggregation by each P_j or the combiner:

$$b = \sum_{i=1}^{M} b_i \mod \mathcal{Q}$$

10: Form shared public key $PK = (b, a)$
11: **return**
- Public output: $PK = (b, a)$
- Private output (per party): secret key share s_i

Note: The full secret key $sk = \sum_{i=1}^{M} s_i \mod \phi(X)$ *is never reconstructed explicitly.*

After a client computes its local model updates, denoted as $\Delta\mathcal{W}$, this algorithm takes over. It processes each parameter tensor of the model update individually. First, it flattens the tensor into a one-dimensional vector. Since the CKKS homomorphic encryption scheme can only encrypt a vector up to a certain length (determined by the polynomial modulus degree N), the algorithm segments the flattened vector into smaller chunks. If a final segment is smaller than the maximum capacity, it is padded with zeros.

Each of these segments is then encoded into a plaintext polynomial and subsequently encrypted using the shared public key, PK. The final output is a dictionary where each key corresponds to a model parameter, and its value is a list of ciphertexts, with each ciphertext containing a segment of the original parameter update. This ensures that the entire model update is securely transmitted without exposing any plaintext data.

Algorithm 4: Encryption and Segmentation

Require:
- CKKS context `ctx` (includes N, scale Δ)
- Model updates $\Delta\mathcal{W}$ (collection of weight/gradient tensors)
- Shared public key PK

Ensure: Encrypted and segmented updates: $\texttt{Enc}(\Delta\mathcal{W})$ — a dictionary mapping each parameter to a list of ciphertexts

Initialize: $\texttt{Enc}(\Delta\mathcal{W}) \leftarrow \{\}$
for each tensor $\Delta\mathcal{W}[i]$ in $\Delta\mathcal{W}$ **do**
  Flatten $\Delta\mathcal{W}[i]$ into 1D vector v
  Let $L \leftarrow \text{length}(v)$
  Compute segments: $n \leftarrow \lceil \frac{L}{N/2} \rceil$
  Initialize: $\texttt{Enc}(\Delta\mathcal{W}[i]) \leftarrow [\,]$
  for $j = 0$ to $n - 1$ **do**
    Extract segment: $v_j \leftarrow v[j \cdot \frac{N}{2} : \min((j+1) \cdot \frac{N}{2}, L)]$
    if $\text{length}(v_j) < \frac{N}{2}$ **then**
      Pad v_j with zeros to length $\frac{N}{2}$
    end if
    Encode v_j into plaintext polynomial $p_j(X) \in \mathbb{Z}[X]/(X^N + 1)$ with scale Δ
    Encrypt: $\texttt{Enc}_{v_j} \leftarrow \texttt{Encrypt_CKKS}(PK, p_j(X))$
    Append $\texttt{Enc}_{v_j}$ to $\texttt{Enc}(\Delta\mathcal{W}[i])$
  end for
  Store metadata (e.g., original shape, length L) for reconstruction
end for
return $\texttt{Enc}(\Delta\mathcal{W})$

After the server performs homomorphic aggregation and sends back the final encrypted result, the participants must collaborate to decrypt it. A threshold T of participants is required for this step. Each participating client uses its private secret share s_k to compute a *partial decryption* of the aggregated ciphertext. This partial decryption is then sent to a designated Combiner. The Combiner aggregates these partial decryptions, and only if the number of participants meets the threshold T, can it successfully reconstruct the plaintext model update.

4.0.2 Server

In our framework, a central server acts as the Homomorphic Aggregation Server and a coordinator for the cryptographic protocols, rather than having a separate key server. The server's primary responsibility is to orchestrate the federated learning process without accessing any sensitive plaintext data, including model updates or private keys. The server's functions are defined by the following workflows:

The server's main task during each training round is to securely aggregate the encrypted model updates received from the participating clients. Upon receiving encrypted updates from m clients, the server performs homomorphic addition directly on the ciphertexts to compute the encrypted sum of all model updates. Subsequently, it performs a homomorphic multiplication with an encrypted scalar $\left(\frac{1}{m}\right)$ to compute the average. This entire process, detailed in Algorithm 5, ensures that the server produces an encrypted aggregated model update without ever decrypting or learning the individual contributions from any client.

Algorithm 5: Homomorphic Aggregation

Require:
- Number of clients: m
- Encrypted model updates: $\texttt{Enc}(\Delta\mathcal{W}_1), \ldots, \texttt{Enc}(\Delta\mathcal{W}_m)$
- CKKS evaluation key: `evk` (for homomorphic operations)

Ensure: Aggregated encrypted update: $\texttt{Enc}(\Delta\mathcal{W})$

Initialize: $\texttt{Enc}(\Delta\mathcal{W}) \leftarrow \{\}$
for each parameter index i (i.e., model layer) **do**
  for each segment index j **do**
    Initialize sum: $\text{Sum}_{i,j} \leftarrow \texttt{Enc}(\Delta\mathcal{W}_1[i][j])$
    for $k = 2$ to m **do**
      $\text{Sum}_{i,j} \leftarrow \texttt{Add_CKKS}(\text{Sum}_{i,j}, \texttt{Enc}(\Delta\mathcal{W}_k[i][j]))$
    end for
    Encode scalar: $\text{Plain}_{1/m} \leftarrow \texttt{Encode_Plain}\left(\frac{1}{m}\right)$
    Compute average:

$$\text{Avg}_{i,j} \leftarrow \texttt{Multiply_CKKS}\left(\text{Sum}_{i,j}, \text{Plain}_{1/m}\right)$$

    Append $\text{Avg}_{i,j}$ to $\texttt{Enc}(\Delta\mathcal{W}[i])$
  end for
end for
return $\texttt{Enc}(\Delta\mathcal{W})$

After aggregation, the server initiates the Threshold Decryption protocol (Algorithm 6). It distributes the final encrypted result to a set of participating clients (at least a threshold of T). The server then acts as a *Combiner*, receiving *partial decryption* components from each participating client. The server's role is limited to collecting these partial decryptions and, if a sufficient number are received, combining them to reconstruct the final plaintext model update. The server cannot decrypt the result on its own and requires the active, successful cooperation of the client quorum. This design removes the need for a trusted key server and distributes the decryption capability, thereby preventing the server from being a single point of failure.

Algorithm 6: Threshold Decryption

Require: • Encrypted ciphertext: $\text{Enc}(x) = (c_0(X), c_1(X)) \in \left(\mathbb{Z}_Q[X]/(X^N+1)\right)^2$
• A set $\mathcal{K}$ of T participating parties, each with secret share $s_k(X)$
Ensure: Decrypted plaintext polynomial: pt

1: **for** each party P_k where $k \in \mathcal{K}$ **do**
2: Compute partial decryption share:

$$d_k(X) \leftarrow (c_1(X) \cdot s_k(X)) \bmod Q$$

3: Transmit partial share $d_k(X)$ to the combiner
4: **end for**
5: **Combiner operations:**

1. Aggregate shares and add public component:

$$D_{\text{total}}(X) \leftarrow \left(c_0(X) + \sum_{k \in \mathcal{K}} d_k(X)\right) \bmod Q$$

2. Perform post-processing:

- Rescale: $D_{\text{rescaled}}(X) \leftarrow \text{Rescale}(D_{\text{total}}(X))$
- Decode: $\text{pt} \leftarrow \text{Decode}(D_{\text{rescaled}}(X))$

6: **return** pt

4.0.3 Security Analysis

The security of the proposed framework is analyzed based on its resilience to key threats inherent in federated learning. Our design provides robust, multi-layered cryptographic guarantees that address vulnerabilities at the levels of data transmission, server processing, and inter-participant privacy.

Protection Against Gradient Leakage Attacks: The framework provides provable security against gradient leakage attacks. By homomorphically encrypting all model updates before transmission using the CKKS scheme (Algorithm 4), the system ensures that raw gradient values are never exposed. This directly thwarts attacks like Deep Leakage from Gradients (DLG) and its variants, which fundamentally rely on access to plaintext gradients to reconstruct training data.

Protection Against an Honest-but-Curious Server: Our system is explicitly designed to operate under an "honest-but-curious" server model. The aggregation server performs all computations directly on ciphertexts (Algorithm 5) , allowing it to correctly execute the Federated Averaging algorithm without ever accessing the plaintext model updates. This design choice cryptographically prevents the central server from inferring sensitive information from individual or aggregated gradients, thus mitigating risks of data misuse.

Inter-Participant Privacy: The privacy of each participant's contribution is maintained throughout the aggregation phase. Since each client's model update is individually encrypted before being sent to the server, no other participating client can inspect the raw contributions of its peers during the server-side aggregation. While all participants receive the final decrypted global model to begin the next round, the homomorphic averaging process ensures that indi-

vidual data signatures are mathematically blended, preserving privacy among the collaborating parties.

Distributed Trust through Threshold Security: A cornerstone of the framework's security is the distribution of trust, which prevents any single point of failure. This is achieved through two core protocols:

The `Distributed Key Generation (DKG)` protocol (Algorithm 2) ensures that no single entity, including the server, ever possesses the complete master secret key.

The `Threshold Decryption protocol` (Algorithm 6) requires a quorum of T participants to collaboratively decrypt the aggregated model update. This mechanism prevents unilateral decryption by the server and ensures the system remains secure even if a limited number of participants are malicious or unavailable.

4.0.4 Computational Optimization

The integration of Fully Homomorphic Encryption, particularly the CKKS scheme, into federated learning introduces substantial computational overhead compared to plaintext operations. To ensure the practical viability of our framework, we employed several key optimization strategies aimed at balancing security, precision, and performance.

Parameter Selection for the CKKS Context: The choice of cryptographic parameters is the most critical factor in performance tuning. We carefully selected the `Polynomial Modulus Degree (N)`, the `Coefficient Modulus (q)`, and the `Scaling Factor (Δ)` to provide a 128-bit security level while minimizing computational load. A smaller `N` and a shorter modulus chain `q` lead to faster operations, but they reduce the noise budget and the number of permissible homomorphic operations. Our selected parameters represent a deliberate trade-off to support our specific neural network architectures efficiently.

SIMD Operations through Packing: The CKKS scheme allows for multiple plaintext values to be "packed" into a single ciphertext, enabling Single Instruction, Multiple Data (SIMD) operations. By encoding up to $N/2$ values in one ciphertext, we can perform vectorized additions and multiplications. This technique is crucial for our framework, as it allows us to process entire layers or segments of the model's weight vectors in parallel, dramatically reducing the number of homomorphic operations required during the aggregation phase.

Model Architecture Considerations: The complexity of the neural network, specifically its multiplicative depth, directly impacts the required FHE parameters and, consequently, performance. While not the primary focus of this work, the choice of a model architecture that is "FHE-friendly" (i.e., has a lower multiplicative depth) is a known optimization strategy. Our custom CNN for CIFAR-10 was specifically engineered to handle the image data while being computationally feasible within the CKKS framework.

These optimization techniques are essential for mitigating the inherent performance costs of cryptographic privacy enforcement, making the framework more practical for real-world deployment scenarios. The impact of these choices on performance and overhead is empirically evaluated in Section 5.6.

Chapter 5

Experimental Framework

This section details the experimental methodology used to evaluate our CKKS-based federated learning framework, including dataset selection, implementation details, baseline comparisons, and evaluation metrics.

5.0.1 Experimental Datasets

To rigorously evaluate the performance, security, and utility of our framework, we selected two canonical benchmark datasets, each serving a distinct purpose in our analysis.

Our primary experiments were conducted using the CIFAR-10 dataset. As a widely recognized benchmark for image classification, CIFAR-10 presents a suitable level of complexity for evaluating the framework's capabilities. The dataset consists of 60,000 32x32 pixel color images distributed across 10 classes, with a standard partition of 50,000 images for training and 10,000 for testing.

To validate the framework's fundamental correctness and establish a performance baseline on a less complex task, we also utilized the MNIST dataset. This dataset of handwritten digits comprises 70,000 28x28 pixel grayscale images, also across 10 classes, with a standard split of 60,000 training and 10,000 test images.

5.0.2 Model Architecture

For the main evaluation on the CIFAR-10 dataset, we developed a custom Convolutional Neural Network (CNN). This architecture is specifically engineered to handle the complexity of multi-channel image data while maintaining a structure that is feasible for the intensive computations required by the CKKS homomorphic encryption scheme.

To validate the framework's correctness on a foundational task, we employed the classic LeNet-5 architecture for experiments on the MNIST dataset. This serves two purposes: first, to confirm that our secure protocols function correctly with a well-understood, standard model; and second, to provide a performance baseline in a less computationally demanding environment.

5.0.3 FHE Context Selection

A critical aspect of implementing a framework based on homomorphic encryption is the careful selection of the cryptographic parameters that define the FHE context. These parameters govern the trade-offs between security, computational performance, and the precision of the encrypted arithmetic. Before conducting our federated learning experiments, we determined a set of parameters for the Cheon-Kim-Kim-Song (CKKS) scheme that provides a standard 128-bit security level while maintaining sufficient precision for neural network training and minimizing computational overhead.

Table 5.1: Modified LeNet-5 CIFAR-10 Parameter Breakdown

Network Neural Layer	**Parameters**	**Bytes Consumed**
conv1.weight	4,800	19,200
conv1.bias	64	256
conv2.weight	204,800	819,200
conv2.bias	128	512
conv3.weight	819,200	3,276,800
conv3.bias	256	1,024
fc1.weight	524,288	2,097,152
fc1.bias	2,048	8,192
fc2.weight	2,097,152	8,388,608
fc2.bias	1,024	4,096
fc3.weight	10,240	40,960
fc3.bias	10	40
Total	3,664,010	14,656,040

Table 5.2: LeNet-5 MNIST Parameter Breakdown

Network Neural Layer	**Parameters**	**Bytes Consumed**
conv1.weight	150	600
conv1.bias	6	24
conv2.weight	2,400	9,600
conv2.bias	16	64
conv3.weight	48,000	192,000
conv3.bias	120	480
fc1.weight	10,080	40,320
fc1.bias	84	336
fc2.weight	840	3,360
fc2.bias	10	40
Total	61,706	246,824

Table 5.3: Model Architecture Comparison

Model	**Parameters**	**Size (MB)**	**Memory (MB)**
LeNet-5 (MNIST)	61,706	0.24	0.26
Modified LeNet-5 (CIFAR-10)	3,664,010	13.98	14.30

Drawing from established best practices for secure and efficient homomorphic encryption, we configured our `tenseal` FHE context with the following settings for the experiments:

- **Scheme**: CKKS
- **Polynomial Modulus Degree (n)**: 2^{13} (8192)
- **Scale (Δ)**: 2^{40}
- **Coefficient Modulus (q_i sizes)**: [60, 40, 40, 60]

The polynomial modulus degree of 8192 provides a strong foundation for 128-bit security while allowing for efficient SIMD operations. The coefficient modulus chain and the scaling factor of 2^{40} were selected to ensure sufficient precision for the neural network's parameters throughout the encrypted aggregation process, preventing significant accuracy loss due to approximation errors.

5.0.4 Data Partitioning for Experiments

To evaluate our framework under diverse conditions, we partitioned the experimental datasets using the IID and non-IID distribution models previously described in Section 3.1.

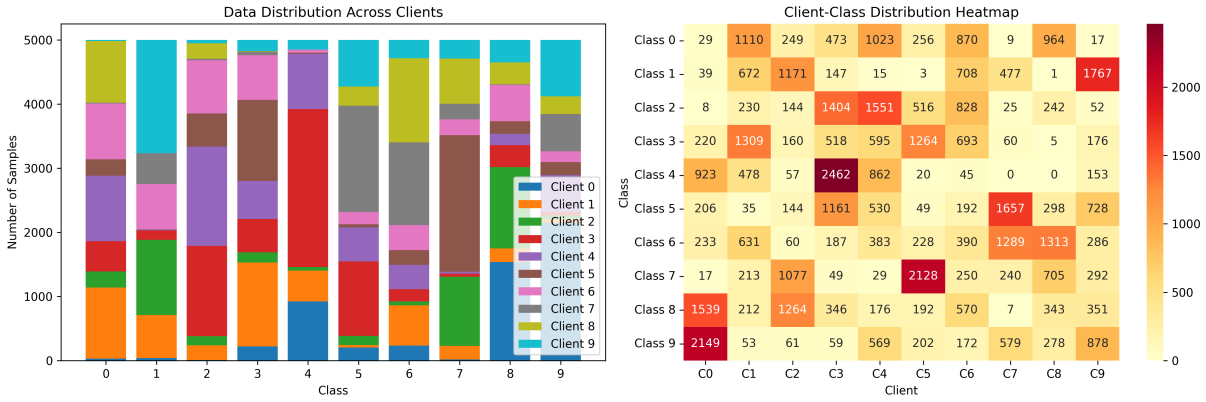


Figure 5.1: Visualization of a Non-IID Data Partition for the CIFAR-10 Dataset

A visualization of an exemplary non-IID partition for the CIFAR-10 dataset across 5 clients is presented in Figure 5.1, clearly illustrating the class imbalance among the participants.

5.0.5 Performance and Overhead Analysis

The performance of the proposed privacy-preserving framework was evaluated over 30 communication rounds. Figure 5.2 presents the key results, assessing the model's learning effectiveness, training efficiency, and the computational overhead introduced by the cryptographic protocols.

Model Convergence and Accuracy: The top row of Figure 5.2 illustrates the model's learning progress. The test accuracy (top-left) exhibits rapid and successful convergence, increasing from approximately 84% to over 98% within the first five communication rounds. Following this initial learning phase, the accuracy stabilizes and remains consistently high, plateauing at approximately 98.5%. Similarly, the test loss (top-right) drops sharply from nearly 0.5 to a stable low value of approximately 0.05 during the same initial five rounds. This demonstrates that the integration of Fully Homomorphic Encryption does not hinder the model's ability to learn and converge effectively, achieving a high degree of accuracy.
Computational and Cryptographic Overhead: The bottom row of Figure 5.2 analyzes the computational costs associated with the framework. The per-round training

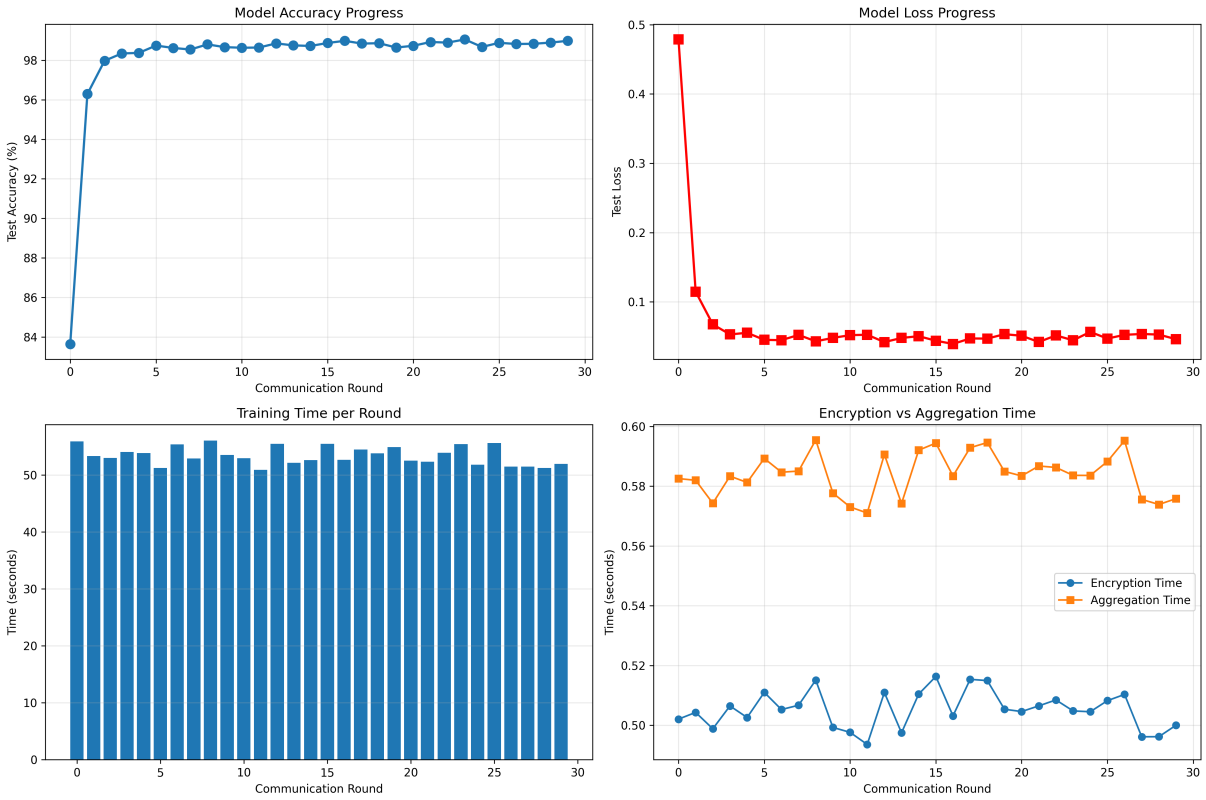


Figure 5.2: Performance analysis of the proposed framework over 30 communication rounds. (Top-left) Test accuracy progression. (Top-right) Test loss convergence. (Bottom-left) Per-round local training time. (Bottom-right) Time overhead of cryptographic operations (encryption and aggregation).

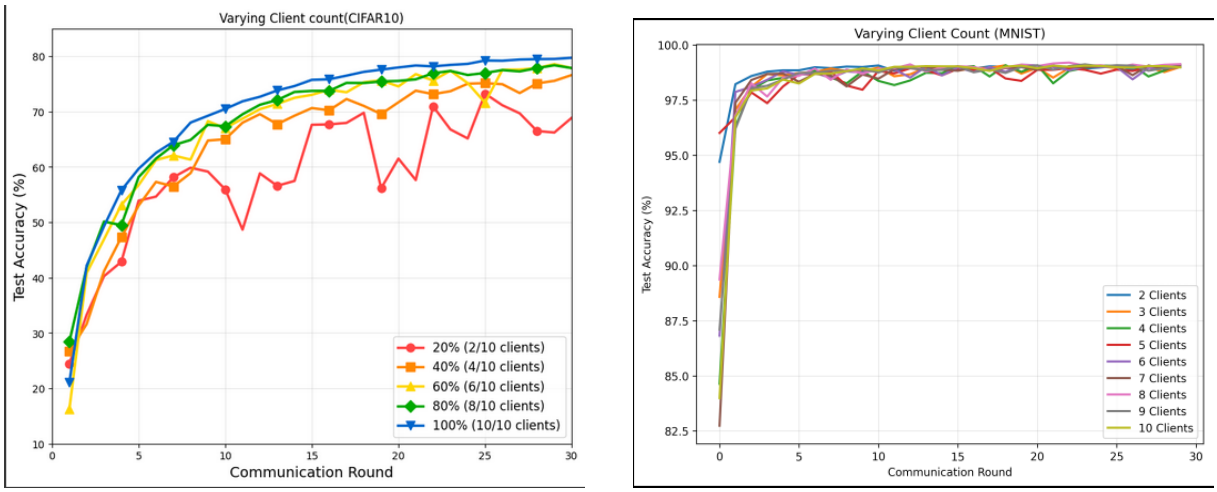
time (bottom-left) represents the local computation performed by each client. This phase is the most time-consuming, averaging a consistent 52-54 seconds per round. Of particular importance is the cryptographic overhead, detailed in the bottom-right panel. This plot isolates the time required for homomorphic encryption of the model updates by the client and the subsequent homomorphic aggregation on the server. The aggregation time consistently averages around 0.58 seconds, while encryption takes approximately 0.51 seconds per round.

5.0.6 Client Participation Impact on Model Performance

To investigate this relationship, we conducted an empirical analysis to quantify how varying the client participation rate affects the global model's convergence speed and final accuracy. This analysis is essential for understanding the trade-offs between system-wide resource utilization and the speed of learning.

In this set of experiments, the total number of clients in the federation was held constant at 10, with all experiments conducted on the non-IID partitioned CIFAR-10 dataset. The Fully Homomorphic Encryption (FHE) component of our framework was active to ensure all operations were performed securely.

The primary variable in this analysis was the number of clients selected to participate in each communication round. We varied this number as a percentage of the total client pool (e.g., from 20% to 100% participation), which corresponds to collecting a different number of client parameter matrices for aggregation in each round. Performance was measured by the test accuracy of the global model after each round of training.



(a) CIFAR-10 dataset results

(b) MNIST dataset results

Figure 5.3: Test accuracy vs communication rounds for different client participation rates in FHE-enabled federated learning on CIFAR-10 and MNIST datasets

As demonstrated in Figure 5.3, the impact of client participation rate on model convergence exhibits distinct patterns across the two datasets. The CIFAR-10 results (Figure 5.3a) show more pronounced sensitivity to participation rates due to the dataset's higher complexity, while the MNIST results (Figure 5.3b) demonstrate more stable convergence across different participation levels.

Table 5.4: Final recorded accuracies for Figure 3 and 4

Percent of Client Selection	**MNIST Test Accuracy**	**CIFAR Test Accuracy**
20% Client	98.43%	68.86%
40% Client	98.55%	76.59%
60% Client	98.75%	77.86%
80% Client	98.76%	77.86%
100% Client	98.77%	79.71%

Table 5.4 provides a quantitative comparison between the two datasets, highlighting the performance gap that reflects the inherent complexity difference between CIFAR-10 and MNIST. The results demonstrate that while both datasets benefit from increased client participation, CIFAR-10 shows more substantial improvements, indicating its greater sensitivity to federated learning dynamics under homomorphic encryption.

Chapter 6

Summary of Contributions

In this work, we successfully developed a privacy-preserving federated learning system that synergistically integrates Homomorphic Encryption (HE) with Secret Sharing. The core contribution is a comprehensive architecture that ensures the confidentiality of model parameters throughout the training lifecycle. The main components and achievements of this framework include:

- **A Secure, Decentralized Architecture**: We designed a system centered around clients and a coordinating aggregation server, eliminating the need for a trusted key server. Security is rooted in two key cryptographic protocols:

  `A Distributed Key Generation (DKG)` protocol that allows clients to collaboratively create a public key, ensuring no single entity ever holds the master secret key.

  `A Threshold Decryption protocol` that requires a quorum of clients to combine their secret shares to decrypt the final aggregated model, preventing unilateral decryption by the server.

- **End-to-End Privacy**: By leveraging the Cheon-Kim-Kim-Song (CKKS) homomorphic encryption scheme, our framework allows the server to perform the Federated Averaging algorithm directly on encrypted data. This protects against gradient leakage attacks and ensures confidentiality even from an "honest-but-curious" server.

- **Empirical Validation**: We validated the framework's performance on the CIFAR-10 and MNIST datasets using custom and standard CNN architectures. Our experimental results demonstrate that the framework can achieve high model accuracy, comparable to non-private federated learning, while introducing a manageable level of computational overhead.

Chapter 7

conclusion

In conclusion, this thesis makes a significant contribution to the field of privacy-preserving machine learning by presenting a functional and secure federated learning framework. By leveraging homomorphic encryption and threshold cryptography, our work demonstrates that it is possible to build collaborative intelligence systems that are both powerful and private, aligning with the growing societal and regulatory demand for trustworthy AI.

7.1 Limitations

While this research successfully demonstrates the viability of the proposed framework, it is important to acknowledge certain limitations that offer avenues for future work:

- **Computational Overhead**: The use of homomorphic encryption, while providing strong security, inherently introduces significant computational and communication overhead compared to plaintext operations. The time required for encryption and aggregation remains a practical challenge for real-time applications or resource-constrained devices.
- **Scalability**: Our experiments were conducted with a moderate number of clients. The performance, network latency, and coordination challenges of the DKG and threshold decryption protocols in a large-scale setting with hundreds or thousands of clients have not been fully explored.
- **Malicious Adversary Model**: The security analysis primarily assumes an "honest-but-curious" server and honest clients. The framework's robustness against malicious adversaries who may attempt to disrupt the protocol or submit poisoned data has not been evaluated.

7.2 Future Research Directions

The findings and limitations of this work open up several promising directions for future research:

- **Performance Optimization**: Future work could focus on reducing the computational overhead by exploring hardware acceleration for homomorphic operations, optimizing cryptographic parameters, or investigating more efficient secret sharing schemes.
- **Large-Scale Deployment and Analysis**: A crucial next step is to evaluate the framework's scalability by simulating its performance in a large-scale federated network. This would involve analyzing the impact of network latency and developing more efficient protocols for managing a large number of client keys and shares.

- **Robustness Against Malicious Adversaries**: The security model could be extended to defend against malicious clients. This could involve integrating verifiable computation techniques to ensure clients perform training correctly or developing robust aggregation methods that can identify and discard malicious updates.
- **Hybrid Privacy Models**: Research into hybrid models that combine the cryptographic guarantees of our framework with the formal privacy definitions of differential privacy could offer even more comprehensive protection. For example, applying differential privacy to the decrypted global model before its distribution could protect against inference attacks on the final output.

Bibliography

[1] Harrison Kurunathan, Hailong Huang, Kai Li, Wei Ni, Ekram Hossain (2022). Machine Learning-Aided Operations and Communications of Unmanned Aerial Vehicles: A Contemporary Survey. https://doi.org/10.48550/arXiv.2211.04324.

[2] Al-Huthaifi, R.; Li, T.; Huang, W.; Gu, J.; Li, C. (2023). Federated learning in smart cities: Privacy and security survey. *Information Sciences*, 632, 833–857. https://doi.org/10.1016/j.ins.2023.01.065.

[3] Rahman, M.M.; Kundu, D.; Suha, S.A.; Siddiqi, U.R.; Dey, S.K. (2022). Hospital patients' length of stay prediction: A federated learning approach. *Journal of King Saud University - Computer and Information Sciences*, 34(8), 7874–7884. https://doi.org/10.1016/j.jksuci.2022.02.014.

[4] Kurunathan, H.; Huang, H.; Li, K.; Ni, W.; Hossain, E. (2024). Machine Learning-Aided Operations and Communications of Unmanned Aerial Vehicles: A Contemporary Survey. *IEEE Communications Surveys & Tutorials*, 26, 496–533. https://doi.org/10.1109/COMST.2024.3354761.

[5] Microsoft (2023). Microsoft SEAL (release 4.1). Microsoft Research, Redmond, WA. https://github.com/Microsoft/SEAL

[6] Zhao, B., Mopuri, K. R., & Bilen, H. (2020). iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*.

[7] Muhammad Firdaus, Harashta Tatimma Larasati, Kyung Hyune-Rhee. (2025). Blockchain-based federated learning with homomorphic encryption for privacy-preserving healthcare data sharing. *Published by Elsevier* https://doi.org/10.1016/j.iot.2025.101579.

[8] Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4, 169–180.

[9] Gentry, C. (2009). A fully homomorphic encryption scheme. Ph.D. Thesis, Stanford University, Stanford, CA, USA.

[10] Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3), 13.

[11] Gentry, C., Halevi, S., & Smart, N. P. (2012). Better bootstrapping in fully homomorphic encryption. *International Workshop on Public Key Cryptography*, 1–16. Springer.

[12] Gentry, C., Sahai, A., & Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Advances in Cryptology—CRYPTO 2013*, 75–92. Springer.

[13] Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. *Advances in Cryptology—EUROCRYPT 2010*, 24–43. Springer.

[14] Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. *Advances in Cryptology—ASIACRYPT 2017*, 409–437. Springer.

[15] Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2019). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys*, 51(4), 79.

[16] Zhang, X., Deng, H., Wu, R., Ren, J., & Ren, Y. (2024). PQSF: Post-quantum secure privacy-preserving federated learning. *Scientific Reports*, **14**, Article number: 74377.
DOI: 10.1038/s41598-024-74377-6, PMID: 39384909, PMCID: PMC11464711.

[17] Fu, F., Xue, H., Cheng, Y., Tao, Y., & Cui, B. (2022). BlindFL: Vertical Federated Machine Learning without Peeking into Your Data. *arXiv preprint* arXiv:2206.07975.
DOI: 10.48550/arXiv.2206.07975

[18] Zhu, L., Liu, Z., & Han, S. (2019). Deep Leakage from Gradients. *arXiv preprint* arXiv:1906.08935.
DOI: 10.48550/arXiv.1906.08935

[19] Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. *Advances in Cryptology—EUROCRYPT 2010*, 24–43. Springer.

[20] Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3), 13.

[21] Gentry, C., Sahai, A., & Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Advances in Cryptology—CRYPTO 2013*, 75–92. Springer.

[22] Firdaus, M., Larasati, H. T., & Hyune-Rhee, K. (2025). Blockchain-based federated learning with homomorphic encryption for privacy-preserving healthcare data sharing. *Internet of Things*, 101579.
DOI: 10.1016/j.iot.2025.101579

Experiments were conducted on Lightning AI Studios and Google Colab platforms using NVIDIA GPUs (T4/A10G/V100) for neural network training and multi-core CPUs for homomorphic encryption operations, as FHE libraries currently lack GPU support.