Privacy-Preserving Hierarchical Federated Learning via CKKS Homomorphic Encryption

Ton That Tam Dinh, Manh Cuong Ho, Ayalneh Bitew Wondmagegn, Dongwook Won, Juyoung Kim, and Sungrae Cho*

School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea

Abstract

Hierarchical federated learning (HFL) reduces cloud-facing traffic by aggregating client updates at intermediate edge servers, but plaintext aggregation exposes model updates to honest-but-curious infrastructure. This paper studies a client–edge–cloud HFL architecture protected by the Cheon–Kim–Kim–Song (CKKS) approximate homomorphic-encryption scheme and evaluated under wireless impairments. Clients encrypt packed model updates, edge servers aggregate ciphertexts, and the cloud completes the second aggregation stage without accessing plaintext updates. A trusted key holder alone decrypts the global update. The wireless model accounts for heterogeneous bandwidth, fading, packet loss, retransmissions, and deadline-based client admission. Experiments on Fashion-MNIST use 20 clients, four edge servers, Dirichlet non-IID partitioning with concentration $\alpha = 0.5$, and 100 global rounds. Four variants are compared: plaintext HFL, CKKS-HFL, wireless plaintext HFL, and wireless CKKS-HFL. Under reliable links, plaintext and CKKS curves nearly overlap and exceed 91% final accuracy, while CKKS relative aggregation error remains below approximately 6×10^{-10} in the ideal-link case. Wireless plaintext HFL remains close to the reliable-link baseline because almost all clients are received each round. In contrast, wireless CKKS-HFL admits only about one to seven clients per round because encrypted updates are much larger, causing noisier convergence and lower final accuracy of approximately 89%. HFL nevertheless reduces encrypted edge-to-cloud traffic from roughly 670 MB at the client tier to about 470 MB in the ideal CKKS case, with larger reductions when wireless admission limits participation. These results show that CKKS numerical error is negligible, whereas ciphertext-induced communication pressure is the dominant practical bottleneck.

Keywords: Hierarchical federated learning, homomorphic encryption, CKKS, secure aggregation, edge computing, privacy-preserving machine learning, TenSEAL

1. Introduction

Federated learning (FL) trains a shared model without centralizing raw data [1]. HFL extends FL by inserting edge aggregators between clients and the cloud, reducing cloud-facing communication and matching the structure of mobile-edge networks [2]. This hierarchy, however, creates an additional privacy exposure: an edge server can inspect individual client updates and the cloud can inspect regional aggregates.

CKKS supports approximate arithmetic over encrypted real-valued vectors [5]. Since weighted FL aggregation mainly requires additions and multiplications by public scalar weights, CKKS can protect model updates while allowing both edge and cloud servers to aggregate directly in the ciphertext domain. Yet encryption expands the payload. In a wireless system, this expansion interacts with rate variation, fading, packet errors, retransmissions, and deadlines. The central systems question is therefore not only whether CKKS preserves model accuracy, but whether encrypted updates can be delivered reliably enough for learning.

This work converts the prior system-design study into a 100-round empirical evaluation. Its contributions are: 1) an end-to-end CKKS-secured two-tier HFL protocol; 2) a wireless admission model that couples ciphertext size with heterogeneous links; 3) a controlled four-way comparison separating encryption and wireless effects; 4) a joint analysis of accuracy, loss, aggregation er-

*Corresponding author

Email address: srcho@cau.ac.kr (and Sungrae Cho)

ror, successful clients, and tier-specific traffic; and 5) a systems-level interpretation that identifies ciphertext-induced client dropout, rather than CKKS approximation, as the primary deployment bottleneck.

2. Related Work

2.1. Federated Learning and Hierarchical Aggregation

Federated learning enables multiple clients to collaboratively train a global model without transferring their raw datasets to a central server. FedAvg remains the most widely adopted aggregation algorithm, in which participating clients perform several local stochastic-gradient-descent steps and transmit their updated model parameters to a server. The server then computes a sample-weighted average according to the amount of local training data held by each client [1, 8]. Although FedAvg reduces direct data exposure, its conventional single-server architecture can create a communication bottleneck when a large number of geographically distributed clients simultaneously upload high-dimensional model updates.

Hierarchical federated learning addresses this limitation by introducing intermediate edge aggregators between the clients and the cloud [2]. Clients first send their local updates to nearby edge servers, which compute regional aggregates. The cloud subsequently combines only the edge-level aggregates to update the global model. When the same sample-count weights are used at both stages, hierarchical aggregation is mathematically equivalent to centralized FedAvg under exact arithmetic and complete client participation. However, HFL reduces the number of cloud-facing transmissions from the number of participating clients to the number of edge servers. This property makes HFL particularly suitable for mobile-edge computing, Internet-of-Things deployments, and future wireless networks in which client-to-cloud connections may be costly or unreliable.

Despite this communication advantage, conventional HFL introduces an additional privacy risk. In plaintext HFL, edge servers receive individual client updates and may attempt to infer information about the corresponding local training data. The cloud may similarly inspect regional edge aggregates. Transport-layer encryption protects updates only while they are in transit and does not prevent an edge or cloud aggregator from accessing plaintext parameters after reception. Therefore, privacy-preserving HFL requires aggregation mechanisms that protect updates during computation, rather than merely during transmission.

2.2. Federated Learning under Non-IID Data

Most theoretical analyses of distributed optimization initially assume that client data are independently and identically distributed. In practice, however, data collected by mobile devices, sensors, hospitals, vehicles, or regional edge domains are usually heterogeneous. Differences in user behavior, sensing environments, hardware, geography, and application context can produce highly imbalanced class distributions and unequal local dataset sizes.

Non-IID data can cause local optimization directions to deviate substantially from the global objective, thereby slowing convergence, increasing round-to-round oscillation, and reducing final accuracy [3, 4]. These effects may become more pronounced in HFL because clients assigned to the same edge server can share similar geographical or contextual characteristics. As a result, edge-level aggregates may themselves be statistically biased before they are combined at the cloud.

Dirichlet allocation is commonly used to emulate label-distribution heterogeneity in federated-learning experiments. A concentration parameter α controls the degree of non-IID behavior: a small value produces highly client-specific class distributions, whereas a large value approaches an IID partition. The present work adopts Dirichlet partitioning to evaluate whether encrypted and wireless-aware hierarchical aggregation remains stable when client updates are statistically heterogeneous.

Wireless failures can further amplify non-IID effects. If clients with poor channels, limited bandwidth, or large encrypted payloads repeatedly miss aggregation deadlines, the successfully received updates may no longer represent the original client population. Consequently, communication failures can create participation bias in addition to the statistical heterogeneity already present in the local datasets. This interaction motivates evaluating privacy protection, wireless reliability, and learning convergence jointly.

2.3. Differential Privacy for Federated and Hierarchical Learning

Differential privacy protects individual training samples by clipping client updates and injecting calibrated random noise before or during aggregation [9, 10]. Its main advantage is that it provides a formal statistical privacy guarantee that can be quantified using a privacy budget. Differential privacy can also be combined with secure aggregation so that the server observes only a noisy aggregate rather than individual updates.

However, differential privacy introduces a direct privacy–utility trade-off. Stronger privacy generally requires larger noise variance, which can reduce model accuracy or slow convergence, especially under highly non-IID data. Privacy accounting also becomes more complicated over many global rounds because the privacy loss accumulates across repeated participation. In an HFL architecture, the location at which noise is added is important. Client-side noise protects updates from edge servers but can accumulate across many clients, whereas edge-level noise requires trust in the edge aggregator. Differential privacy therefore provides statistical protection but does not by itself allow an untrusted edge or cloud server to compute exact weighted aggregates over hidden updates.

2.4. Secure Multiparty Computation and Secure Aggregation

Secure multiparty computation and secret-sharing-based secure aggregation allow multiple parties to jointly compute an aggregate without revealing their individual inputs. In federated learning, each client can divide its update into shares that are distributed among multiple servers or combined through pairwise masks. Under suitable non-collusion assumptions, the aggregator learns only the final sum.

These approaches generally preserve model accuracy because they operate over exact or quantized values and do not require privacy noise. Nevertheless, they often introduce additional communication rounds, pairwise key establishment, mask generation, dropout-recovery procedures, or coordination among several non-colluding entities. Client dropout is especially challenging because the masks or shares associated with an unavailable client must be reconstructed or removed before aggregation can be completed.

The communication requirements of SMPC can be significant in wireless environments. A protocol that is efficient over stable wired links may become difficult to execute when clients experience fading, packet loss, heterogeneous bandwidth, or strict communication deadlines. Hierarchical SMPC can reduce some cloud-facing traffic, but it still requires careful coordination between clients, edge servers, and possibly multiple computation parties.

2.5. Homomorphic Encryption for Federated Aggregation

Homomorphic encryption enables arithmetic operations to be performed directly over ciphertexts. In an encrypted federated-learning system, clients encrypt their model updates before transmission, and an untrusted server computes the aggregate without decrypting the individual contributions. Only an authorized key holder decrypts the final result.

The CKKS scheme is particularly suitable for neural-network parameters because it supports approximate arithmetic over real- or complex-valued vectors and provides SIMD-style packing of many model coordinates into a single ciphertext [5]. Federated averaging mainly requires ciphertext additions and multiplications by known plaintext weights, resulting in a relatively shallow homomorphic circuit. Consequently, CKKS-based aggregation does not generally require bootstrapping when appropriate encryption parameters are selected.

Unlike exact cryptosystems, CKKS introduces a small approximation error due to encoding, rescaling, and finite ciphertext precision. This error must be evaluated against the corresponding plaintext aggregate. In addition, the polynomial modulus degree, coefficient-modulus chain, encoding scale, and number of packed slots determine the trade-off among security, numerical precision, ciphertext size, and computational cost.

Most CKKS-based FL studies focus primarily on encryption time, decryption time, homomorphic-aggregation latency, ciphertext expansion, or numerical error under reliable communication. However, ciphertext expansion also changes the wireless behavior of the learning system. Larger payloads require longer transmission times, increase the probability of deadline violation, and may require more packet retransmissions. Therefore, cryptographic overhead can indirectly affect model convergence by reducing the number and statistical diversity of successfully participating clients.

2.6. Privacy-Preserving Hierarchical Federated Learning

Privacy-preserving HFL must protect updates at both aggregation tiers. Protecting only the edge-to-cloud link is insufficient because the edge server still observes individual client updates. Similarly, encrypting client-to-edge traffic with conventional transport security does not prevent the edge server from decrypting the updates before aggregation.

In the CKKS-based HFL architecture considered in this paper, clients encrypt packed, sample-weighted model updates. Each edge server sums the corresponding ciphertext blocks from its successfully received clients and forwards only the encrypted edge aggregate to the cloud. The cloud then performs a second ciphertext-domain aggregation. Neither tier accesses the plaintext model updates, and only a trusted

Table 1
Conceptual comparison of privacy-preserving hierarchical federated-learning approaches.

Method	Protection during aggregation	Potential accuracy impact	Principal overhead	Main trust or security assumption
Plaintext HFL	No protection after update reception	None from the privacy mechanism	Low computation and communication cost	Edge and cloud aggregators are trusted not to inspect updates
DP-HFL	Statistical protection through clipping and calibrated noise	Possible degradation due to privacy noise, particularly under strong non-IID data	Noise calibration, clipping, and multi-round privacy accounting	Correct implementation of the privacy mechanism; the guarantee depends on the selected privacy budget
SMPC-HFL	Individual updates are hidden through secret sharing or masking	Usually negligible, except for quantization or finite-field encoding	Additional messages, coordination, key establishment, and dropout recovery	Required number of computation parties do not collude
CKKS-based flat FL	Client updates remain encrypted during server-side aggregation	Small approximation error from CKKS encoding and arithmetic	Encryption, ciphertext expansion, and homomorphic aggregation over all clients	Secret key remains unavailable to the aggregation server
Proposed CKKS-HFL	Client, edge-level, and cloud-level updates remain encrypted during both aggregation stages	Small CKKS approximation error; wireless dropout may indirectly affect convergence	Client encryption, two-stage ciphertext aggregation, and enlarged wireless payloads	Trusted key holder or threshold decryptor; edge and cloud follow the protocol but may be honest-but-curious

or threshold-based decryptor obtains the final global aggregate.

Hierarchical aggregation provides a particularly important benefit for homomorphically encrypted FL. Although the client-to-edge payload remains enlarged by CKKS encryption, the cloud receives encrypted updates from only the edge servers rather than from every client. Thus, HFL does not eliminate ciphertext expansion at the wireless access tier, but it can substantially reduce encrypted edge-to-cloud traffic and the number of ciphertext additions performed at the cloud.

Unlike approaches that evaluate cryptographic computation independently of networking conditions, this work studies the coupled effects of CKKS ciphertext size, heterogeneous bandwidth, fading-dependent transmission rates, packet loss, retransmissions, and communication deadlines. The comparison includes plaintext HFL, CKKS-HFL over ideal links, wireless plaintext HFL, and wireless CKKS-HFL. This design makes it possible to distinguish numerical approximation error from participation loss caused by encrypted communication overhead.

Table 1 highlights that the privacy mechanisms address different adversarial and systems assumptions. Differential privacy provides a statistical guarantee but may reduce utility, whereas SMPC avoids approximation noise at the cost of interactive communication and non-collusion assumptions. CKKS allows non-interactive aggregation by honest-but-curious edge and cloud servers, but introduces ciphertext expansion and bounded numerical error. The proposed approach adopts CKKS because the HFL aggregation circuit is dominated by additions and public scalar weighting, while explicitly evaluating how encrypted payload expansion affects wireless client availability and learning convergence.

3. System Model

Figure 1 illustrates the overall architecture of our proposed system model, outlining the primary components and the structural flow of information between them. As depicted in the diagram, the framework is organized into distinct layers to efficiently handle data processing, communication protocols, and system constraints. This visual layout serves as the foundational blueprint for the technical mechanisms detailed throughout the remainder of this section, providing a clear roadmap of how individual subsystems interact to achieve the target performance metrics.

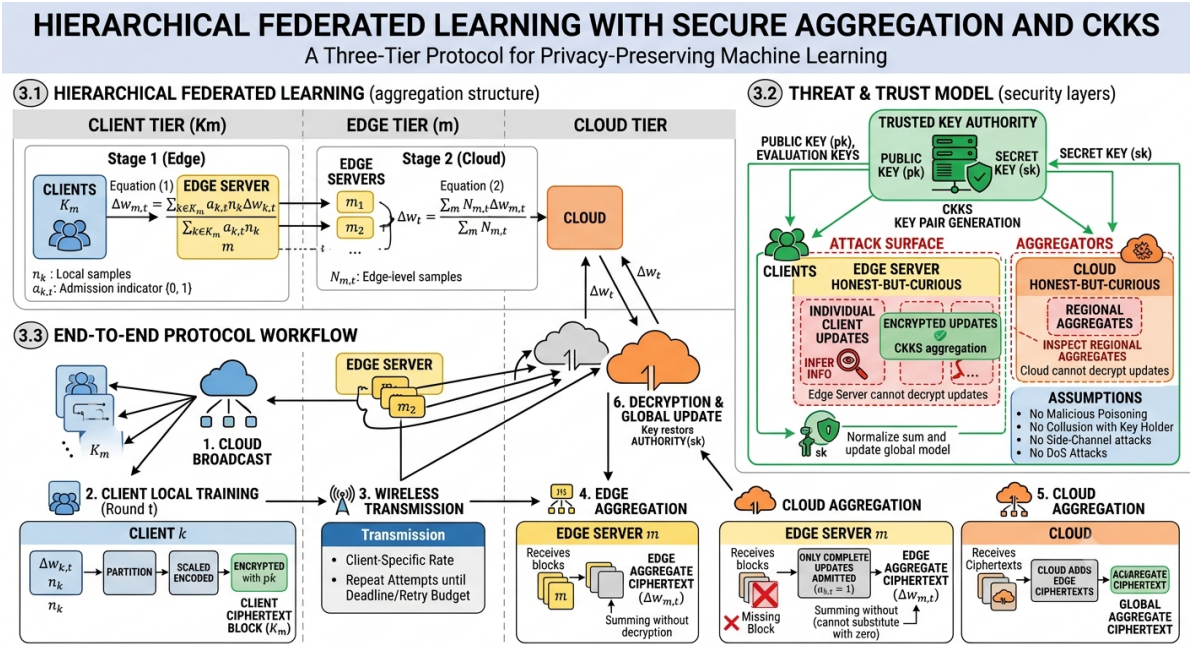

Figure 1. System Model

3.1. Hierarchical Federated Learning

Let $\mathcal{K}_m$ be the clients associated with edge server m, and let n_k denote the local sample count. Client k produces update $\Delta\mathbf{w}_{k,t}$. The plaintext two-stage aggregate is

$$\Delta\mathbf{w}_{m,t} = \frac{\sum_{k\in\mathcal{K}_m} a_{k,t} n_k \Delta\mathbf{w}_{k,t}}{\sum_{k\in\mathcal{K}_m} a_{k,t} n_k}, \tag{1}$$

$$\Delta\mathbf{w}_t = \frac{\sum_m N_{m,t} \Delta\mathbf{w}_{m,t}}{\sum_m N_{m,t}}, \tag{2}$$

where $a_{k,t} \in \{0, 1\}$ indicates whether the complete update is admitted and $N_{m,t} = \sum_{k\in\mathcal{K}_m} a_{k,t} n_k$.

3.2. Threat and Trust Model

Clients, edge servers, and the cloud are assumed to follow the aggregation protocol but may be honest-but-curious. An edge server may attempt to infer information from individual client updates, while the cloud may inspect regional aggregates. A trusted key authority generates the CKKS public/secret key pair and evaluation keys. The public key and evaluation material are distributed to clients and aggregators, whereas the secret key is retained only by the authority. Consequently, neither edge servers nor the cloud can decrypt client or edge-level updates. The present scope excludes malicious update poisoning, collusion with the key holder, side-channel attacks, and denial-of-service behavior.

3.3. End-to-End Protocol Workflow

At the beginning of round t, the cloud broadcasts the current global model through the edge tier. Each selected client trains locally and forms a model difference $\Delta\mathbf{w}_{k,t}$. In plaintext modes, the update is serialized directly. In CKKS modes, the vector is partitioned into slot-compatible blocks, multiplied by the public sample count n_k, encoded, and encrypted. The wireless module then computes a client-specific rate and repeatedly attempts transmission until either all blocks are delivered or the retry/deadline budget is exhausted. Edge servers aggregate only complete updates. The cloud adds the edge ciphertexts, after which the trusted key holder decrypts and normalizes the sum before the global model is updated. This complete-update rule is essential because a missing ciphertext block cannot be replaced with zeros without distorting a subset of model coordinates.

3.4. CKKS Aggregation

The Cheon–Kim–Kim–Song (CKKS) scheme is an approximate homomorphic-encryption scheme designed for arithmetic over real- or complex-valued vectors. This property makes CKKS suitable for federated learning because model parameters and model updates are represented by floating-point values, and aggregation mainly requires vector addition and multiplication by public scalar weights.

Let $\mathbf{x} \in \mathbb{R}^d$ denote a real-valued model-update block. Before encryption, $\mathbf{x}$ is mapped into the CKKS plaintext space by the encoding function

$$\mathrm{Encode}_\Delta : \mathbb{R}^d \to \mathcal{P}, \tag{3}$$

where $\mathcal{P}$ denotes the plaintext space and Δ is the encoding scale. The scale controls the numerical precision with which the real-valued entries are represented.

The public-key encryption operation is then written as

$$\mathrm{Enc}_{pk} : \mathcal{P} \to \mathcal{C}, \tag{4}$$

where pk is the CKKS public key and C is the ciphertext space. Thus, the encrypted representation of $\mathbf{x}$ is

$$\mathbf{c} = \mathrm{Enc}_{pk}\left(\mathrm{Encode}_\Delta(\mathbf{x})\right). \tag{5}$$

The notation Enc_{pk} therefore refers only to public-key encryption of an already encoded CKKS plaintext. In some parts of the manuscript, the encoding and encryption steps are written compactly as a single operation for readability, but they remain conceptually distinct.

At the receiving side, ciphertext recovery consists of two steps. First, the secret-key decryption function

$$\mathrm{Dec}_{sk} : \mathcal{C} \to \mathcal{P} \tag{6}$$

maps the ciphertext back to the CKKS plaintext space, where sk is the secret key. Second, the CKKS decoding function

$$\mathrm{Decode} : \mathcal{P} \to \mathbb{R}^d \tag{7}$$

recovers an approximate real-valued vector.

Accordingly, CKKS satisfies the following approximate-correctness relation:

$$\mathrm{Decode}\Big(\mathrm{Dec}_{sk}\Big(\mathrm{Enc}_{pk}(\mathrm{Encode}_\Delta(\mathbf{x}))\Big)\Big) = \mathbf{x} + \boldsymbol{\varepsilon}_{\mathrm{CKKS}}, \tag{8}$$

where $\boldsymbol{\varepsilon}_{\mathrm{CKKS}}$ denotes the bounded approximation error introduced by finite-precision encoding, polynomial arithmetic, scaling, and homomorphic operations.

Unlike exact homomorphic-encryption schemes, CKKS does not generally recover $\mathbf{x}$ bit-for-bit. Instead, it produces a numerically close approximation. The magnitude of $\boldsymbol{\varepsilon}_{\mathrm{CKKS}}$ depends on the polynomial modulus degree, coefficient-modulus chain, encoding scale, circuit depth, and number of rescaling operations. In the proposed HFL framework, the encrypted circuit is shallow because aggregation requires only ciphertext additions and, when needed, ciphertext–plaintext multiplications by public sample weights. Therefore, the CKKS error remains small and bootstrapping is not required under the selected parameterization.

The update is divided into blocks that fit the CKKS slot capacity. Client k encrypts the weighted qth block as

$$\mathbf{c}_{k,t}^{(q)} = \mathrm{Enc}_{pk}\Big(\mathrm{Encode}_\Delta\left(n_k \Delta \mathbf{w}_{k,t}^{(q)}\right)\Big). \tag{9}$$

Edge m computes $\mathbf{c}_{m,t}^{(q)} = \sum_{k\in\mathcal{K}_m} a_{k,t}\mathbf{c}_{k,t}^{(q)}$, and the cloud computes $\mathbf{c}_t^{(q)} = \sum_m \mathbf{c}_{m,t}^{(q)}$. A trusted key holder decrypts, decodes, and normalizes by the total admitted sample count. The relative aggregation error is

$$e_t^{\mathrm{rel}} = \frac{\|\widehat{\Delta\mathbf{w}}_t - \Delta\mathbf{w}_t^{\mathrm{plain}}\|_2}{\|\Delta\mathbf{w}_t^{\mathrm{plain}}\|_2}. \tag{10}$$

3.5. Wireless Admission

For client bandwidth B_k, transmit power P_k, large-scale gain g_k, fading $h_{k,t}$, and noise density N_0, the rate is

$$R_{k,t} = B_k \log_2\left(1 + \frac{P_k g_k |h_{k,t}|^2}{N_0 B_k}\right). \tag{11}$$

With payload S_k, the one-attempt transmission time is $T_{k,t}^{\mathrm{tx}} = S_k / R_{k,t}$. Retransmissions multiply this delay. A client is admitted only if every block arrives intact before the deadline:

$$a_{k,t} = \mathbb{I}\Big[z_{k,t} = 1,\ T_{k,t}^{\mathrm{cmp}} + \widetilde{T}_{k,t}^{\mathrm{tx}} \le T_{\max}\Big]. \tag{12}$$

Partial encrypted updates are discarded because zero-filling missing ciphertext blocks would bias selected coordinates.

3.6. Computation and Communication Cost

Let d be the number of model parameters, L the number of usable CKKS slots per ciphertext, and $Q = \lceil d/L \rceil$ the number of ciphertext blocks per update. Plaintext HFL communicates $O(d)$ numeric values per participating client. CKKS-HFL communicates Q serialized ciphertexts, so its payload is approximately QS_{ct} bytes, where S_{ct} is the serialized ciphertext size. Client-side cryptographic work is $O(Q)$ encryptions. Since all clients use an identical packing layout, edge and cloud aggregation requires only blockwise ciphertext addition and no rotations. For K_t admitted clients and M_t active edges, the aggregation work is $O(K_t Q)$ additions at the edge tier and $O(M_t Q)$ additions at the cloud tier.

The hierarchy reduces cloud-facing objects from $K_t Q$ in flat encrypted FL to at most $M_t Q$ in HFL. It does not, however, remove the client-to-edge expansion. Thus, the architecture primarily protects the cloud link and central aggregator from per-client encrypted traffic, while the access link remains the critical capacity bottleneck. This distinction is directly reflected by the traffic results in Section V.

Table 2
Compared Variants

Variant	CKKS	Wireless impairments
Plaintext HFL	No	No
CKKS-HFL	Yes	No
Wireless plaintext HFL	No	Yes
Wireless CKKS-HFL	Yes	Yes

Table 3
Experimental Configuration

Parameter	Value
Dataset / task	Fashion-MNIST classification
Model size	421,642 parameters
Clients / edge servers	20 / 4
Global rounds / local epochs	100 / 1
Batch size	64
Optimizer	SGD, momentum 0.9
Learning rate / weight decay	0.01 / 5×10^{-4}
Data partition	Dirichlet, $\alpha = 0.5$
Compared modes	Plain, CKKS, wireless plain, wireless CKKS
Primary metrics	Accuracy, loss, clients, error, traffic

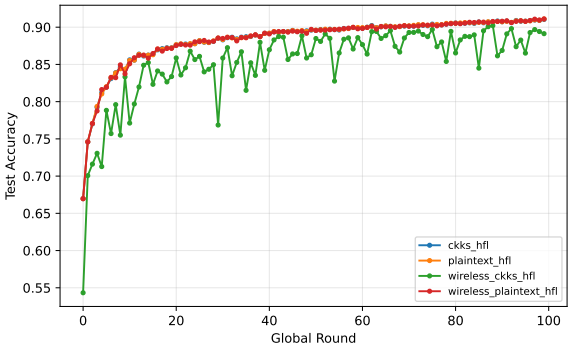

Figure 2. Test accuracy over 100 global rounds.

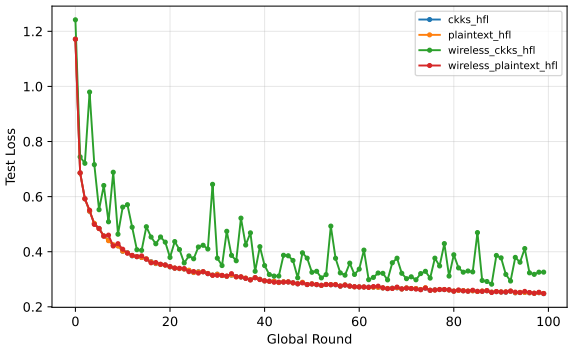

Figure 3. Test loss over 100 global rounds.

4. Experimental Methodology

The study uses Fashion-MNIST, a CNN with 421,642 trainable parameters, 20 clients, four edges, one local epoch per round, batch size 64, SGD with momentum 0.9, learning rate 0.01, weight decay 5×10^{-4}, Dirichlet $\alpha = 0.5$, and 100 global rounds. The four evaluated variants are summarized in Table 2, while Table 3 lists the common learning configuration.

The reported figures provide round-wise test accuracy, test loss, successful clients, relative aggregation error, client-to-edge traffic, and edge-to-cloud traffic. Because the archive contains figures rather than raw CSV logs, numerical values below are read from the plotted trajectories and are therefore reported approximately.

5. Results and Discussion

5.1. Learning Accuracy and Loss

Fig. 2 shows that plaintext HFL and CKKS-HFL almost perfectly overlap over 100 rounds. Both rise from roughly 67% to slightly above 91%, demonstrating that CKKS approximation does not materially disturb optimization under reliable links. Wireless plaintext HFL follows the same trajectory and finishes at a comparable level because 18–20 clients are usually received per round.

Wireless CKKS-HFL behaves differently. Its first-round accuracy is near 54%, then it climbs rapidly but exhibits persistent fluctuations between approximately 83% and 90%. It finishes near 89%, about two percentage points below the reliable-link variants. The corresponding loss curve in Fig. 3 is noisier, with repeated spikes up to about 0.65, whereas the other variants decrease smoothly toward approximately 0.25.

5.2. Successful Clients

Fig. 4 explains the convergence gap. Both ideal-link variants receive all 20 clients. Wireless plaintext HFL usually receives 19–20 clients and only occasionally falls to 17–18. Wireless CKKS-HFL receives roughly one to seven clients, with most rounds clustered around three to six. Thus, the dominant degradation

Table 4
Summary of the Experimental Findings

Variant	Final accuracy	Successful clients/round	Relative aggregation error	Main communication behavior
Plaintext HFL	$> 91\%$	20	0	Lowest payload; ideal delivery
CKKS-HFL	$> 91\%$	20	$< 6 \times 10^{-10}$	High access traffic; reduced cloud traffic
Wireless plaintext HFL	$\approx 91\%$	Usually 18–20	0	Mild losses; convergence preserved
Wireless CKKS-HFL	$\approx 89\%$	About 1–7	Peaks near 1.4×10^{-9}	Retransmissions and deadline misses dominate

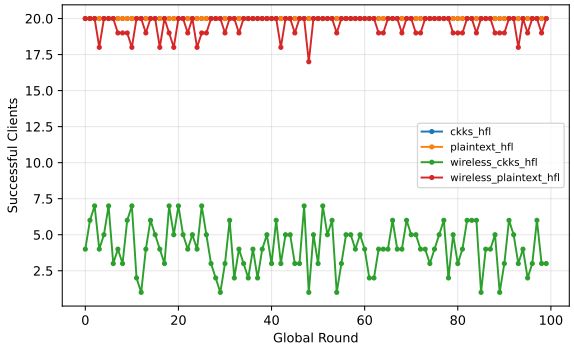

Figure 4. Number of complete client updates admitted per round.

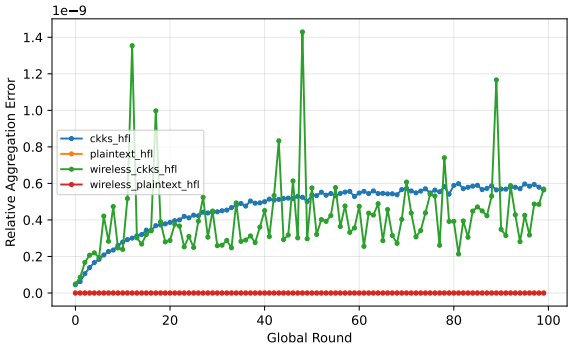

Figure 5. Relative error between CKKS and plaintext aggregates.

is not CKKS numerical error but participation sparsity induced by ciphertext expansion and deadline failures. Under non-IID data, this sparse and time-varying subset also changes the class mixture of each aggregate.

5.3. *CKKS Approximation Error*

Fig. 5 reports extremely small relative error. Ideal-link CKKS-HFL rises gradually and remains below about 6×10^{-10}. Wireless CKKS-HFL is more variable, with rare peaks near 1.4×10^{-9}, but remains far below a level that would explain the observed accuracy gap. Plaintext variants are zero by definition. These results confirm that encrypted arithmetic is numerically faithful for this shallow aggregation circuit.

5.4. *Communication Traffic*

The client-to-edge traffic in Fig. 6 is approximately 670,000 KB per round for ideal CKKS-HFL, compared with about 32,000–38,000 KB for plaintext transmission. Wireless CKKS-HFL varies around 670,000–810,000 KB because retransmissions increase attempted traffic. Wireless plaintext HFL remains close to the plaintext baseline.

At the cloud tier, Fig. 7 shows the advantage of hierarchy. Ideal CKKS-HFL sends approximately 470,000 KB from edges to the cloud, lower than the client-tier encrypted traffic because only four edge aggregates are forwarded. Wireless CKKS-HFL typically sends about 70,000–240,000 KB, reflecting the smaller number of edges with admitted client updates. Plaintext edge-to-cloud traffic is approximately 20,000–25,000 KB. Hence, HFL does not remove the expensive encrypted client uplink, but it substantially limits cloud-facing encrypted traffic.

5.5. *Security, Reliability, and Fairness Trade-offs*

The secure aggregation mechanism protects update confidentiality during both aggregation stages, but confidentiality alone does not guarantee robust learning. A client whose ciphertexts arrive late is excluded even if its local update is statistically valuable. If bandwidth, distance, or fading conditions correlate with geography or user population, repeated deadline misses can create a persistent representation bias. In that case, the global objective is optimized over a communication-selected population rather than the intended client population. This issue is particularly important under non-IID data because each missed client may remove classes or operating conditions that are rare elsewhere.

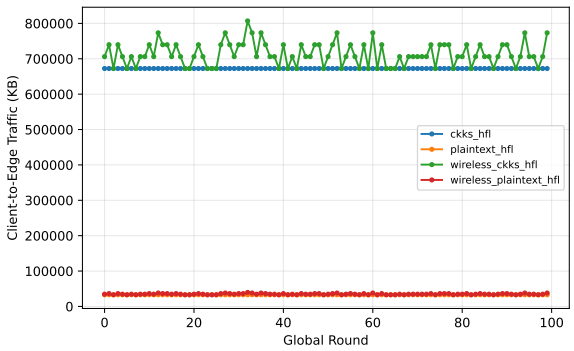

Figure 6. Client-to-edge traffic per global round.

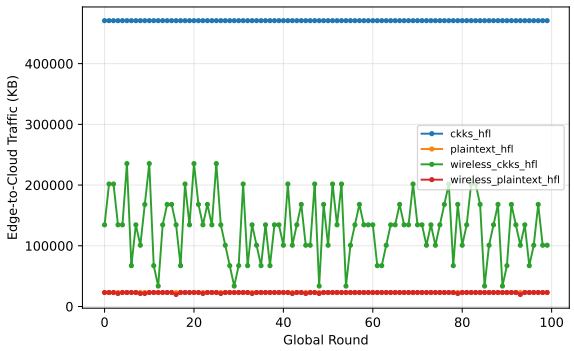

Figure 7. Edge-to-cloud traffic per global round.

A practical scheduler should therefore track not only instantaneous channel quality but also historical participation. One option is to assign a deficit counter to each client and increase its priority whenever it is excluded. Another is to optimize a joint score combining expected delivery probability, sample count, update novelty, and participation age. Such policies can trade a modest increase in communication delay for improved long-term fairness. The present results motivate this direction because wireless CKKS-HFL repeatedly aggregates from only a small subset of the 20 available clients.

Reliability mechanisms also require care. Increasing the retry limit improves the chance of complete delivery, but it raises attempted traffic and may still violate the global round deadline. Extending the deadline increases participation but reduces the control frequency of the learning system. Layer-wise deadlines could provide a middle ground: high-value or low-dimensional layers may be required for admission, while less influential layers could be deferred or transmitted less frequently. However, such partial-update designs would require a new aggregation rule and are not equivalent to the complete-update policy evaluated here.

6. Conclusion

This paper presented an empirical study of CKKS-secured wireless-aware HFL. CKKS-HFL matches plaintext HFL under reliable links and maintains relative aggregation error below approximately 10^{-9}. Wireless plaintext HFL also remains close to the ideal baseline because nearly all clients succeed. The main challenge appears when CKKS and wireless constraints are combined: only a small fraction of encrypted updates meet the deadline, producing noisy convergence and a modest final-accuracy loss. HFL still reduces cloud-facing encrypted traffic through edge aggregation. The results therefore identify communication-aware secure aggregation, rather than CKKS arithmetic precision, as the primary research bottleneck.

Acknowledgment

This work was supported in part by the IITP (Institute of Information & Communications Technology Planning & Evaluation) - ITRC (Information Technology Research Center) (IITP-2026-RS-2023-00258639, 50%) grant funded by the Korea government (MSIT), in part by the NRF (National Research Foundation of Korea) (No. RS-2023-00209125) grant funded by the Korea government (MSIT), and in part by the Chung-Ang University Young Scientist Scholarship in 2025.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[2] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE ICC*, 2020, pp. 1–6.

[3] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE ICDE*, 2022, pp. 965–978.

[4] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[5] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. ASIACRYPT*, 2017, pp. 409–437.

[6] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," arXiv:1708.07747, 2017.

[7] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," TR 38.901, v17.0.0, 2022.

[8] K. Li, S. Hu, B. Wu, S. Zou, W. Ni, and F. Dressler, "Undermining federated learning accuracy in EdgeIoT via variational graph auto-encoders," in *Proc. IEEE Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2025, pp. 716–721.

[9] L. Qi, B. Wu, F. Wang, K. Li, W. Ni, and A. Jamalipour, "Differentially private energy sharing among smart grid-powered base stations," in *Proc. IEEE 100th Veh. Technol. Conf. (VTC2024-Fall)*, 2024, pp. 1–6.

[10] L. Qi, Z. Huang, Y. Chen, B. Ma, and B. Wu, "Clustered and differentially private federated load forecasting for EV charging networks," in *Proc. Int. Conf. Smart Grid Sustain. Energy (SGSE)*, 2025, pp. 153–157.