

Article

V-MHESA: A Verifiable Masking and Homomorphic Encryption-Combined Secure Aggregation Strategy for Privacy-Preserving Federated Learning

Soyoung Park ^{*}  and Jeonghee Chi 

Department of Computer Science & Engineering, Konkuk University, Seoul 05029, Republic of Korea; jhchi@konkuk.ac.kr

^{*} Correspondence: soyoungpark@konkuk.ac.kr

Abstract

In federated learning, secure aggregation is essential to protect the confidentiality of local model updates, ensuring that the server can access only the aggregated result without exposing individual contributions. However, conventional secure aggregation schemes lack mechanisms that allow participating nodes to verify whether the aggregation has been performed correctly, thereby raising concerns about the integrity of the global model. To address this limitation, we propose V-MHESA (Verifiable Masking-and-Homomorphic Encryption-combined Secure Aggregation), an enhanced protocol extending our previous MHESA scheme. V-MHESA incorporates verification tokens and shared-key management to simultaneously ensure verifiability, confidentiality, and authentication. Each node generates masked updates using its own mask, the server's secret, and a node-only shared random nonce, ensuring that only the server can compute a blinded global update while the actual global model remains accessible solely to the nodes. Verification tokens corresponding to randomly selected model parameters enable nodes to efficiently verify the correctness of the aggregated model with minimal communication overhead. Moreover, the protocol achieves inherent authentication of the server and legitimate nodes and remains robust under node dropout scenarios. The confidentiality of local updates and the unforgeability of verification tokens are analyzed under the honest-but-curious threat model, and experimental evaluations on the MNIST dataset demonstrate that V-MHESA achieves accuracy comparable to prior MHESA while introducing only negligible computational and communication overhead.

Keywords: federated learning; secure aggregation; masking; homomorphic encryption; verifiability; dropout resilience

MSC: 94A60; 68T07; 68M14; 68P27



Academic Editor: Hui Cui

Received: 3 October 2025

Revised: 6 November 2025

Accepted: 14 November 2025

Published: 17 November 2025

Citation: Park, S.; Chi, J. V-MHESA: A Verifiable Masking and Homomorphic Encryption-Combined Secure Aggregation Strategy for Privacy-Preserving Federated Learning. *Mathematics* **2025**, *13*, 3687. <https://doi.org/10.3390/math13223687>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Federated learning (FL) [1] enables multiple distributed nodes to collaboratively train a global model without sharing their raw data, thereby preserving privacy and reducing data transfer. A central challenge in FL lies in protecting local model parameters during aggregation, as individual updates can leak sensitive information through inference or inverse attacks [2–4]. Secure aggregation protocols address this issue by allowing the server to compute only the aggregated result of local updates without accessing any individual updates. Over the years, various secure aggregation mechanisms have been proposed,

leveraging techniques such as secure multiparty computation [5,6], differential privacy [7,8], masking [9–16], and homomorphic encryption [17–29].

Masking-based schemes, which are among the most widely applied approaches for privacy preservation in the federated learning, achieve efficiency by having nodes add random masks to their local updates, which are designed to cancel out during aggregation. Although conceptually simple, these schemes require pairwise mask sharing among all nodes, which imposes communication and management overhead that scales with the number of nodes. Additionally, dropout events necessitate extra procedures to recover missing masks, further complicating the protocol.

Homomorphic encryption (HE)-based schemes protect local updates through encryption and enable direct aggregation over ciphertexts. However, they incur high computational overhead and often require the use of a shared secret key for encryption by all nodes, creating a single point of failure. Multi-key HE [24] mitigates this issue by allowing each node to use a distinct key, but the final decryption requires complex collaborative operations among all participants.

To address these limitations, our previous work introduced MHESA (Masking-and-Homomorphic-Encryption-Combined Secure Aggregation) [30], a hybrid scheme that integrates the simplicity of masking with the robustness of homomorphic encryption. In MHESA, local parameters are individually masked, and the masks are encrypted using the CKKS homomorphic encryption scheme [29]. This design eliminates the need for inter-node mask sharing, allows nodes to use distinct keys for homomorphic encryption, and enables automatic decryption of aggregated masks at the server without additional joint operations.

While MHESA effectively guarantees the confidentiality of local updates, it lacks mechanisms to verify the correctness and integrity of the aggregated results. Communication errors, adversarial tampering, or server misbehavior can lead to incorrect aggregation, thereby undermining the accuracy of the global model. Since the correctness of aggregation directly affects the overall performance of federated learning, verifiability becomes indispensable.

In this paper, we extend MHESA by proposing V-MHESA (Verifiable Masking-and-Homomorphic-Encryption-Combined Secure Aggregation), a novel protocol that ensures both confidentiality and verifiability. The proposed scheme provides the following features:

- **Correctness and integrity verification:** In each learning round, nodes generate their local updates in a verifiable form and additionally produce verification tokens corresponding to randomly selected local parameter values. These tokens enable lightweight yet reliable validation of the global update. The server simply aggregates both the local updates and the verification tokens to obtain the global model update and a global verification token. Using this global information, each node can verify whether its own local parameters have been accurately reflected in the global model update, thereby ensuring both correctness and integrity.
- **Server authentication:** Local updates are generated using a pre-shared server secret, ensuring that only the legitimate server can produce valid global model updates.
- **Authenticated aggregation group and efficient shared-key management:** During local update generation, each node incorporates an additional random nonce that is shared exclusively among the nodes. This design ensures that only the nodes—rather than the server—can ultimately recover the true global model parameters. The initial random nonce is distributed once during the setup phase, encrypted under each node's public key. In subsequent learning rounds, every node can independently refresh the nonce without requiring any additional exchange or synchronization, thereby eliminating both computational and communication overhead for nonce sharing. To form the aggregation group of legitimate nodes, the server provides a fresh round nonce in

each learning round. Nodes respond by generating a simple authentication code over the round nonce and their identifiers, enabling the server to authenticate participants. Consequently, only nodes that return valid authentication codes are admitted into the aggregation group, effectively preventing unauthorized nodes from contributing to the learning process

- **Robustness against dropouts:** The proposed scheme preserves both confidentiality and verifiability even when some nodes drop out or submit delayed updates.

We analyze the security of V-MHESA under the honest-but-curious threat model and evaluate its performance on the MNIST dataset [31]. The experimental results demonstrate that the proposed protocol achieves strong privacy protection, reliable integrity verification, and robustness against dropouts, while incurring only negligible computational and communication overhead.

The remainder of this paper is organized as follows: Section 2 reviews the related work on secure and verifiable aggregation in federated learning. Section 3 presents the proposed V-MHESA model, describing its algorithms and the federated learning protocol in detail. Section 4 analyzes the security of the scheme, focusing on the confidentiality of local updates and the unforgeability of global updates, and evaluates its performance in terms of learning accuracy, aggregation correctness, and computational efficiency. Section 5 discusses the proposed model in depth, and Section 6 concludes the paper.

2. Related Work

Federated learning has inspired a wide range of secure aggregation techniques aimed at preserving the confidentiality of client updates. Broadly, these studies can be categorized into masking-based schemes, homomorphic encryption-based schemes, and more recent verifiable aggregation approaches

Bonawitz et al. [9,10] introduced an additive masking protocol in which clients conceal their local updates using pairwise masks that cancel during aggregation, ensuring that the server only learns the sum of all updates. The protocol was later extended with Shamir's secret sharing to tolerate client dropouts. To further reduce the overhead of pairwise exchanges, So et al. [15] designed turbo-aggregation, which performs circular aggregation within subgroups to localize communication. Similarly, Kim et al. [16] proposed a group-based masking protocol that clusters nodes by processing latency and location, thereby reducing dropout sensitivity while providing a mechanism for public verification of mask integrity without relying on Shamir's scheme.

Another line of research employs homomorphic encryption (HE) to enable aggregation over ciphertexts. Aono et al. [21] and Fang–Qian [22] proposed HE-based federated learning schemes in which local model parameters are encrypted with a common key, allowing the server to compute encrypted aggregates. However, this single-key design requires all participants to share the same secret key. Park–Lim [23] addressed this limitation by allowing each node to use its own key pair, enabling the server to update the global model within a distributed cryptosystem, although the design relies on an additional trusted computation provider for decryption. Liu et al. [24] proposed a multi-key FHE (MKFHE)-based protocol that allows nodes to encrypt under distinct keys while still supporting aggregation. Their construction is round-efficient and supports dynamic participation by generating refresh keys for newly joining nodes. To reduce payload size, Jin et al. [25] selectively encrypted only privacy-sensitive parameters, thereby mitigating ciphertext expansion, albeit still requiring a common HE key. Park et al. [30] introduced MHESA, a secure aggregation scheme that integrates masking with multi-key homomorphic encryption. By combining the simplicity and efficiency of masking-based aggregation with the strong confidentiality

of homomorphic encryption, MHESA achieves secure aggregation while mitigating the limitations of both approaches

While the above techniques provide privacy, they do not inherently guarantee that the server aggregates updates honestly. Recent works therefore emphasize verifiability—enabling clients to validate that the global update truly reflects their submitted inputs. Xu et al. [32] proposed VerifyNet, which integrates secure aggregation with cryptographic commitments, and Guo et al. [33] introduced VeriFL, which leverages homomorphic hashes (HH) to allow fast and communication-efficient verification. To improve practicality under dropouts, Buyukates et al. developed LightVeriFL, which employs lightweight homomorphic hashing [34]. Behnia et al. [35] presented e-SeaFL, reducing the aggregation to a single round with authenticated vector commitments (VC). Peng et al. [36] designed a communication-efficient protocol that combines single-mask encryption with lightweight hash-based checks. More recently, Zhou et al. [37] proposed GVSA, which allows group-level verification, while Xu et al. [38] and Li et al. [39] developed efficient and lightweight verifiable secure aggregation protocols (LVSA) that remain robust under dropout and adversarial manipulation. Table 1 compares the proposed V-MHESA with the previously reported verifiable schemes, highlighting its key technical distinctions and improvements.

Table 1. Comparison of the characteristics of V-MHESA and existing verifiable federated learning schemes.

Characteristic	V-MHESA	VerifyNet [32]	VeriFL [33]	LightVeriFL [34]	e-SeaFL [35]	Peng et al. [36]	GVSA [37]	Xu et al. [38]	LVSA [39]
Components	FL server, Node	TA, Cloud server, User	Aggregation server, Client	Central server, User	Aggregation server, User, Assisting nodes (AN)	TA, Central server, User	Central server, Group aggregation server, User	Server, Compute nodes, Client	Aggregation server, User, Assisting nodes
Local update generation	Masking + HE	Masking	Masking	Masking	Masking	Masking	Masking	Masking	Masking
Mask generation method	Non-interactive single mask	Pairwise mask	Pairwise mask	Pairwise mask	Single mask computed with all assisting nodes	Single mask + HSS	Single mask + HSS	Single mask + HSS	Non-interactive single mask
Verification mechanism	Masking-based verification token	HHF + Bilinear Pairing	HHF + Equivocal Commitment (Amortized Verification)	EC-based HHF + a variation of the Pedersen commitment	HHF + APVC (Authenticated Pedersen VC)	HHF	Verification Tag using the Hadamard product operation	Homomorphic secret sharing + Masking-based verification MAC	HHF + Inner Product calculation
Additional data size for verification	Selected gradient set	$O(w)$	$O(w)$	$O(w)$	$O(w)$	$O(w)$	$O(w)$	$O(w)$	$O(w)$
Node collaboration during each learning (aggregation) phase	X	O (User provides all secret shares for the shared key and self-mask of each user)	O (Client provides all secret shares for the self-mask of each client, its decommitment strings, and decommitment)	O (User exchanges its encoded mask and commitment with other users)	O (Assisting node provides the sum of the round secrets for users)	O (Node provides the sum of secret shares received)	O (Node provides the sum of secret shares received within its group)	O (Client provides the sum of secret shares for masking; Compute node provides the sum of secret shares for verification)	O (User calculates the inner product of its local gradient)
Server Auth.	O	X	X	X	X	X	X	X	X
Node Auth.	O (Hash-based authentication code)	X	X	X	O (Digital signature)	X	O (Digital signature)	X	X

$|w|$: the size of local model parameter (gradient) set.

Compared with prior approaches, the proposed V-MHESA requires neither a trusted third party nor assistance from other nodes to verify the correctness of the global model. It operates without requiring node collaboration—i.e., no additional communication between nodes or between nodes and the server—during the learning phase. Moreover, it eliminates the need for complex cryptographic primitives such as homomorphic hash functions (HHF) or homomorphic secret sharing (HSS). Instead, each node generates lightweight verification tokens for only a small subset of model parameters using a simple masking technique. The federated server simply aggregates these tokens, while each node verifies the correctness and integrity of the global model through basic arithmetic operations. This design results in negligible computational overhead for verification. Furthermore, the proposed scheme simultaneously provides server authentication, node authentication, and robustness against dropout scenarios.

3. V-MHESA-Based Federated Learning Model

3.1. System Overview

The proposed federated learning model consists of a single federated server (denoted as FS) and a set of N static nodes (each denoted as U_i). The nodes hold heterogeneous local datasets of varying sizes that follow a non-IID distribution, yet each node possesses sufficient computational resources to perform local training. All communication and interaction occur exclusively between FS and each node through standard Internet connections. The federated learning process comprises two phases: (i) the *Setup* phase, during which the cryptographic parameters required for secure aggregation are generated and distributed, and (ii) the *Learning* phase, where local parameters are collected and the global model parameters are updated. The *Setup* phase is executed only once at the beginning of the learning process, whereas the *Learning* phase is iteratively repeated until the global model converges.

At the t -th *Learning* phase, FS and nodes update the global and local model parameters as follows: Given the global parameter w^{t-1} generated by FS in round $t - 1$, each node U_i computes its local parameter

$$w_i^t := w^{t-1} - \eta \nabla F_i(w^{t-1}), \quad (1)$$

where $\nabla F_i(w^{t-1})$ is the average gradient on its local dataset using the current model w^{t-1} and a fixed learning rate η . If the size of dataset held by U_i is denoted by x_i , then the total size across all nodes is $x = \sum_i x_i$. Each node transmits its data-size-weighted local parameter $\frac{x_i}{x} w_i^t$ to FS . FS aggregates these weighted parameter to update the global model as

$$w^t = \sum_{i=1}^N \frac{x_i}{x} w_i^t, \quad (2)$$

and then shares the result with all nodes.

However, if the local parameter w_i^t is directly exposed during aggregation, an adversary could deduce sensitive information about the local datasets via inverse attacks [2–4]. Therefore, a secure aggregation protocol is required to ensure that w_i^t remains confidential while still allowing FS to correctly compute the global update. Building upon our prior MHESA scheme, this paper introduces an enhanced protocol, V-MHESA, which satisfies stricter security requirements. The system operates under the following threat model.

- **Honest-but-curious model:** FS and nodes securely maintain their private information and do not collude with other parties. They strictly follow the prescribed protocol; however, they may attempt to infer additional information from legitimately obtained data.

Under this model, the proposed aggregation protocol is designed to satisfy the following security requirements:

- Privacy of local parameters: Local datasets and parameters held by each node remain confidential and are not exposed to any other nodes, *FS*, or any external entities.
- Correctness verification of global parameters: Each node can verify that its own local parameters have been accurately incorporated into the global update.
- Integrity verification of global parameters: Each node can verify that the global parameters have not been altered or tampered with during transmission, ensuring that the received values match those generated by *FS*.
- Mutual authentication of server and nodes: *FS* can authenticate that local parameters were generated by registered nodes, while the nodes can authenticate that the global parameters were genuinely produced by *FS*.
- Robustness against dropouts: Even if some nodes drop out during the aggregation phase due to communication failures, the remaining nodes can continue the aggregation process, and *FS* can still compute a correct global update. Furthermore, if delayed updates from dropout nodes arrive after aggregation, the privacy of all nodes' local parameters—including those of the dropout nodes—remain protected.

To realize secure aggregation, the protocol employs multiple cryptographic primitives: Elliptic Curve Cryptography (ECC) for public-key encryption and decryption, SHA-256 for hashing, and the CKKS homomorphic encryption (CKKS-HE) scheme for privacy-preserving aggregation.

Figure 1 illustrates the core aggregation workflow of the proposed V-MHESA model. During each *Learning* phase, every node U_i generates its masked local update D_i by masking the local parameter w_i with a mask M_i , it then produces the ciphertext E_i of M_i using CKKS-HE and a verification token T_i . The resulting local update $LU_i = \langle D_i, E_i, T_i \rangle$ is transmitted to *FS*. Upon receiving all LU_i s, *FS* simply aggregates them and, using its secret S , derives the global model parameter GW and the global verification token GT . The global update $GU = \langle GW, GT \rangle$ is then distributed back to all participating nodes. Each node updates its local model using GW if and only if GU passes the proposed verification process. Conversely, if a modified or forged global update (e.g., GU_m due to transmission error or GU_f from a malicious actor) is received, the verification process fails. The node then notifies the verification failure to *FS* and requests retransmission of a valid GU . Through this mechanism, V-MHESA ensures that every node updates its local model only with a verified and correct global model parameter, thereby preserving the integrity and correctness of the overall federated learning process.

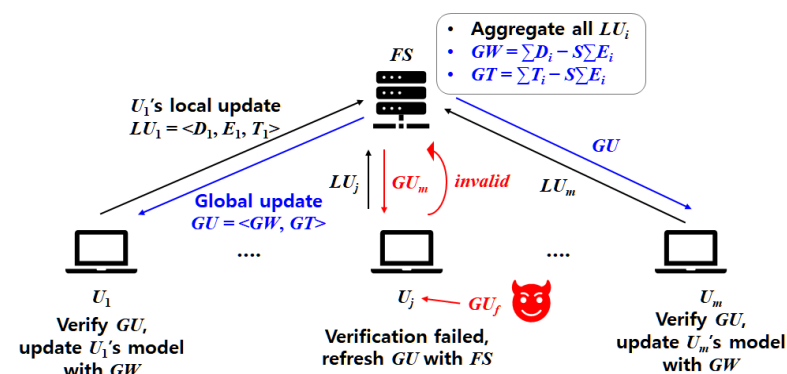


Figure 1. The core aggregation workflow of V-MHESA.

3.2. MHESA: Masking and Homomorphic Encryption–Combined Secure Aggregation

We first briefly describe the baseline aggregation protocol, MHESA, which serves as the foundation of our proposed model. In MHESA, each node U_i masks its local parameter w_i with a random value M_i chosen by the node, thereby ensuring confidentiality. To enable automatic mask removal during aggregation at FS, each mask M_i is encrypted using CKKS-HE scheme and transmitted together with the masked parameter. Importantly, this encryption process embeds a partial decryption share generated by each node. As a result, when the encrypted masks are aggregated, they are automatically decrypted into the sum of the original masks. It should be emphasized that although the aggregated masks can be recovered, the individual masks remain undiscoverable. FS thus computes the decrypted sum of the encrypted masks and subtracts it from the sum of the masked local parameters, thereby obtaining the average of the local parameters.

The MHESA protocol consists of the following six algorithms: *KeyGen()*, *GroupKeyGen()*, *Ecd()*, *Dcd()*, *LocUpGen()* and *GlobUpGen()*. Among these, the *Ecd()* and *Dcd()* algorithms are identical to those defined in CKKS-HE scheme. For clarity, we reformulate the two encryption and decryption algorithms, *Enc()* and *Dec()*, as *LocUpGen()* and *GlobUpGen()*, respectively. The technical details of each algorithm are provided as follows:

For a positive integer M , $\Phi(M)$ is the M -th cyclotomic polynomial of degree $n = \phi(M)$. $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ is a power-of-two degree cyclotomic ring, and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ be the residue ring of $\mathcal{R}$ modulo q . For $Q = q^2$, a polynomial $A(x) \in \mathcal{R}_Q$ is defined as $A(x) = \sum_{0 \leq i < n} a_i \cdot X^i$ with the vector of its coefficients $(a_0, \dots, a_{n-1})$ in $\mathbb{Z}_Q^n$. The coefficient vector $(a_0, \dots, a_{n-1})$ is denoted as A . For a real $\sigma > 0$, $D(\sigma^2)$ represents a distribution over $\mathcal{R}$, sampling n coefficients independently from the discrete Gaussian distribution with variance σ^2 . For a positive integer h , $HWT(h)$ is the set of signed binary vectors in $\{-1, 0, 1\}^n$ whose Hamming weight is exactly h . For a real $0 \leq \rho \leq 1$, the distribution $ZO(\rho)$ draws each entry in the vector from $\{-1, 0, 1\}^n$, assigning a probability of $\rho/2$ for both -1 and $+1$, and a probability of $1 - \rho$ for 0 .

- *KeyGen*(n, h, q, σ, A, Q) generates a public–private key pair $\langle pk_i, sk_i \rangle$ and the commitment c_i for node U_i . It samples $s_i \leftarrow HWT(h)$, $v_i \leftarrow ZO(0.5)$ and $e_i, e_{0i} \leftarrow D(\sigma^2)$. It sets the secret key $sk_i = (1, s_i)$ and public key $pk_i = -A \cdot sk_i + e_i \pmod{Q}$. Then, it sets a commitment c_i for v_i such as $c_i = A \cdot v_i + e_{0i} \pmod{Q}$. It outputs $\langle pk_i, sk_i, c_i \rangle$.
- *GroupKeyGen*(PK, C, U_T) generates a group public key $PK_T = \sum_{u_j \in U_T} pk_j$, a group commitment $C_T = \sum_{u_j \in U_T} c_j$, and the total size of datasets $X_T = \sum_{u_i \in U_T} x_i$, for a given node set $U_T \subseteq U$, where $PK = \{pk_i\}$, $C = \{c_i\}$ and $X = \{x_i\}$ for all U_i in U . It outputs $\langle PK_T, C_T, X_T \rangle$.
- *Ecd*($z; \Delta$) generates an integral plaintext polynomial $m(X)$ for a given $(n/2)$ -dimensional vector $z = (z_j)_{j \in T} \in \mathbb{Z}[i]^{n/2}$. It calculates $m(X) = \mu^{-1} \left(\left[\Delta \cdot \pi^{-1}(z) \right]_{\mu(R)} \right) \in \mathcal{R}$, where $\Delta \geq 1$ represents a scaling factor, and π is a natural projection defined by $(z_j)_{j \in Z_M^*} \mapsto (z_j)_{j \in T}$ for a multiplicative subgroup T of Z_M^* satisfying $Z_M^*/T = \{\pm 1\}$. μ is a canonical embedding map from integral polynomial to elements in the complex field $\mathbb{C}^n$. It computes a polynomial whose evaluations at the complex primitive roots of unity in the extension field $\mathbb{C}$ correspond to the given vector of complex numbers.
- *Dcd*($m; \Delta$) returns the vector $z = \pi \circ \mu \left(\Delta^{-1} \cdot m \right)$ for an input polynomial m in $\mathcal{R}$, i.e., $z_j = \Delta^{-1} \cdot m \left(\zeta_M^j \right)$ for $j \in T$.
- *LocUpGen*($w_i, x_i, X_T, PK_T, C_T, sk_i, v_i$) outputs a secure local update $LU_i = \langle D_i, E_i \rangle$ for a set of local parameters w_i . It chooses a real random mask $M_i \in \mathbb{R}$ and generates a masked local parameter $D_i = \frac{x_i}{X_T} \cdot w_i + M_i$. To generate a ciphertext E_i for M_i , it samples $e_{1i} \leftarrow D(\sigma^2)$, generates a plaintext polynomial $m_i(x) = Ecd(M_i; \Delta)$, and then creates $E_i = v_i \cdot PK_T + m_i + e_{1i} + C_T \cdot sk_i \pmod{Q}$.

- $GlobUpGen(LU_T, U_T)$ outputs a global model update w for given $LU_T = \{LU_i\}$ for all U_i in U_T . It calculates $D = \sum_{i \in LU_T} D_i$ and $E = \sum_{i \in LU_T} E_i \pmod{q}$. Here, $E = \sum (m_i + e')$ by CKKS-HE algorithm. It computes $E' = Dcd(E; \Delta) = \sum M_i$. Finally, it generates $w = D - E' = \sum_{U_i \in U_T} \frac{x_i}{X_T} \cdot w_i$.

Each node generates its own key pair $\langle pk_i, sk_i \rangle$ using the $KeyGen()$ algorithm. FS then generates a group public key $\langle PK_T, C_T \rangle$, which is used to encrypt the masking values of the nodes in the t -th *Learning* round, as specified in $GroupKeyGen()$. This group key is constructed from the public keys of all nodes participating in the corresponding *Learning* round. The local parameter w_i of node U_i is masked with M_i , and M_i is subsequently encrypted as defined in $LocUpGen()$. The ciphertext E_i of M_i includes both the encryption of M_i under the group key $\langle PK_T, C_T \rangle$ and a partial decryption share generated using the node's secret key sk_i . Since E_i is encrypted with the group key, decryption would normally require the combination of all partial decryption shares corresponding to the secret keys of the nodes whose public keys contributed to the group key generation. In the proposed scheme, however, each ciphertext already embeds the partial decryption share of its corresponding node. Consequently, when all ciphertexts from participating nodes are aggregated, decryption is automatically completed, revealing only the sum of all masking values. Importantly, each ciphertext E_i contains only the partial decryption share of its corresponding node. Since the partial decryption shares of E_i from other nodes are not given, the individual mask values remain hidden, while only the sum of the masks is recovered. Finally, FS can obtain the global model parameter w using the $GlobUpGen()$ algorithm.

Based on the above algorithms, the MHESA-based federated learning is performed as follows:

At the *Setup* stage, FS collaboratively establishes system parameters and keys with each node.

1. FS generates and publishes system parameters $\langle n, h, q, \sigma, A, Q \rangle$ to all nodes in U .
2. Each U_i generates its key pair and commitment $\langle pk_i, sk_i, c_i \rangle \leftarrow KeyGen(n, h, q, \sigma, A, Q)$ and sends $\langle pk_i, c_i, x_i \rangle$ to FS .
3. FS sets $PK = \{pk_i\}$, $C = \{c_i\}$ and $X = \{x_i\}$ for all U_i in U .

At the t -th *Learning* iteration ($t > 0$), FS interacts with each U_i as follows:

1. FS broadcasts a new iteration start message to all nodes in U .
2. All available nodes respond with their x_i s.
3. FS sets the t -th node group U_T for all replied nodes and generates group parameters $\langle PK_T, C_T, X_T \rangle \leftarrow GroupKeyGen(PK, C, U_T)$, and then broadcasts it to all nodes in U_T .
4. Each U_i generates t -th local updates $\langle D_i^t, E_i^t \rangle \leftarrow LocUpGen(w_i^t, x_i, X_T, PK_T, C_T, sk_i, v_i)$ for the t -th local parameter w_i^t and sends it to FS . (The local learning process to obtain w_i^t from w^{t-1} with its local dataset is omitted.)
5. FS generates the t -th global model update $w^t \leftarrow GlobUpGen(D_T, E_T, U_T)$ and shares it with all nodes in U .

3.3. Notations

Before providing the detailed description of the proposed V-MHESA protocol, we summarize the main parameters and functions used throughout the paper in Table 2. These notations are consistently used in the subsequent sections for clarity and precision.

Table 2. Notations.

Notation	Description
FS	A federated learning server.
U_i, U	The i – th node U_i in the set U of all nodes.
U_R	A representative node that generates the initial random nonce hR^0 , shared only among nodes in U .
S	The master secret of FS , shared with all nodes.
ID_i	The identifier of U_i , randomly generated by FS .
K_i^+, K_i^-	The public/private key pair of U_i for ECC
pk_i, sk_i, c_i	The public/private key pair and commitment of U_i for CKKS-HE
x_i, X	The dataset size of U_i ; X denotes the sum of all x_i across nodes in U
U_T	The aggregation group of participating nodes in the t -th round.
X_T	The sum of all x_i s of nodes in U_T .
PK_T, C_T	The t – th group public key and group commitment for all nodes in U_T .
$hR^t = PRG(hR^{t-1})$	The t – th hash – chained pseudo – random value derived from the initial input hR^0 .
RN^t	The t -th round nonce generated by FS .
R^t, R_1^t, R_2^t	The t – th random nonces shared only among the nodes, where R^t is generated from both hR^t and RN^t , and R_1^t, R_2^t are derived from R^t .
idx^t	The t -th set of parameter indexes randomly selected by FS for verification.
AC_i	An authentication code of U_i to prove its legitimacy for joining the aggregation.
M_i^t	The t – th random masking value of U_i .
w_i^t	The t – th local model parameter of U_i .
w^t	The t -th global model parameter.
$LU_i^t = \langle D_i^t, E_i^t, T_i^t \rangle$	The t – th local update of U_i , where D_i^t is the masked local parameter, E_i^t is the ciphertext of M_i^t , and T_i^t is the verification token corresponding to w_i^t .
$GU^t = \langle GW^t, GT^t \rangle$	The t – th global update generated by FS , where GW^t is the global model update for w^t , and GT^t is the global verification token for w^t .
$PKE(K^+, M), PKD(K^-, C)$	Public – key encryption and decryption of a message M and its ciphertext C using the key pair $\langle K^+, K^- \rangle$; ECIES (Elliptic Curve Integrated Encryption System) is used in the proposed model.
$DS(K^-, M), VS(K^+, Sig, M)$	Digital signature generation and verification of a message M and its signature Sig using the key pair $\langle K^+, K^- \rangle$; ECDSA is used in the proposed model.
$H(M)$	A hash function for a message M ; SHA-256 hash is used in our model
$PRG(M)$	A pseudo-random generator that produces a random integer for input M .
$PKParams = \langle p, a, b, G, n \rangle$	The ECC domain parameters: p is a prime modulus, a and b are curve coefficients, G is a base (generator) point, and n is the order of G .
$HEParams = \langle n, h, q, \sigma, A, Q \rangle$	The CKKS-HE domain parameters, as described in Section 3.2.

3.4. V-MHESA: A Verifiable MHESA Scheme

In this section, we present V-MHESA, an enhanced version of MHESA that provides verification of global model parameters. Unlike the original MHESA, V-MHESA requires nodes to validate the correctness and integrity of the global parameters at the end of each *Learning* round, and only upon successful verification do they proceed to the next round. To achieve this, V-MHESA introduces three key mechanisms: (1) generation of a fresh random shared nonce in every round, known exclusively to the nodes; (2) construction of local updates in a verifiable form; and (3) a verification procedure that allows nodes to confirm the correctness and integrity of the global update produced by FS .

3.4.1. Algorithms for V-MHESA

In the new model, the original key generation algorithm $KeyGen()$, the local update generation algorithm $LocUpGen()$, and the global update generation algorithm $GlobUpGen()$

are replaced by $VKeyGen()$, $VLocUpGen()$, and $VGlobUpGen()$, respectively. In addition, a new algorithm $Verf()$, is introduced to verify the validity of the global update. All other algorithms remain unchanged from the original protocol as described in Section 3.2. The detailed specifications of the newly developed algorithms are described as follows:

- $VKeyGen(PKParams, HEPParams)$ generates two pairs of public–private keys $\langle K_i^+, K_i^- \rangle$ and $\langle pk_i, sk_i \rangle$, and a commitment c_i of a node U_i . $\langle K_i^+, K_i^- \rangle$ is used for ECC and $\langle pk_i, sk_i \rangle$ is used for CKKS-HE. $\langle K_i^+, K_i^- \rangle$ are generated by ECC standard (the detailed algorithm is omitted). The algorithm to generate $\langle pk_i, sk_i \rangle$ and c_i is identical to the algorithm in $KeyGen()$ described in Section 3.2. It outputs $\langle K_i^+, K_i^-, pk_i, sk_i, c_i \rangle$.
- $VLocUpGen(w_i, R, x_i, X_T, PK_T, C_T, sk_i, v_i, S, idx)$ outputs a secure local update $LU_i = \langle D_i, E_i, T_i \rangle$ for a local parameter w_i . Here, R is a random nonce shared among all nodes in U_T and S is the master secret of FS . It chooses a real random mask $M_i \in \mathbb{R}$ and generates a masked local parameter set as follows:

$$D_i = R \cdot \frac{x_i}{X_T} \cdot w_i + S \cdot M_i. \quad (3)$$

To generate a ciphertext E_i of M_i , it samples $e_{1i} \leftarrow D(\sigma^2)$, generates a plaintext polynomial $m_i(x) = Ecd(M_i; \Delta)$, and then creates

$$E_i = v_i \cdot PK_T + m_i + e_{1i} + C_T \cdot sk_i \pmod{Q}. \quad (4)$$

A verification token T_i is additionally created. Using R , it sets $R_1 = PRG(R)$ and $R_2 = PRG(R_1)$, and generates T_i as follows:

$$T_i = R_1 \cdot \frac{x_i}{X_T} \cdot w_i[idx] + R_2 + S \cdot M_i, \quad (5)$$

where $w_i[idx]$ is a set of selected weight parameter values according to a given index set idx .

- $VGlobUpGen(LU_T, U_T)$ outputs a global update $GU = \langle GW, GT \rangle$ for the global model parameters w , given the set of local updates $LU_T = \{LU_i\}$ from all nodes $U_i \in U_T$. It computes $D = \sum_{i \in LU_T} D_i$ and $E = \sum_{i \in LU_T} E_i \pmod{q}$. According to the CKKS-HE, $E = \sum(m_i + e')$. Then, it obtains $E' = Dcd(E; \Delta) = \sum M_i$ by decoding E with scaling factor Δ . It also computes $T = \sum_{i \in LU_T} T_i$. Finally, the algorithm generates the global model update GW and the corresponding verification token GT for the global model parameter w as follows:

$$GW = D - S \cdot E' = \left(R \sum_{i \in U_T} \frac{x_i}{X_T} w_i + S \sum_{i \in U_T} M_i \right) - S \sum_{i \in U_T} M_i = R \sum_{i \in U_T} \frac{x_i}{X_T} w_i = R \cdot w, \quad (6)$$

$$\begin{aligned} GT = T - S \cdot E' &= \left(R_1 \sum_{i \in U_T} \frac{x_i}{X_T} w_i[idx] + \sum_{i \in U_T} R_2 + S \sum_{i \in U_T} M_i \right) - S \sum_{i \in U_T} M_i \\ &= R_1 \sum_{i \in U_T} \frac{x_i}{X_T} w_i[idx] + \sum_{i \in U_T} R_2 = R_1 \cdot w[idx] + N_T \cdot R_2, N_T = |U_T|. \end{aligned} \quad (7)$$

- $Verf(GU, R, idx)$ verifies the validity of w and outputs either true or false. Given $GU = \langle GW, GT \rangle$, it computes $R_1 = PRG(R)$ and $R_2 = PRG(R_1)$, and then calculates $w' = GW/R$ and $T' = (GT - N_T \cdot R_2)/R_1$. Finally, it verifies whether the following equation holds:

$$\|w'[idx] - T'\| < \varepsilon \quad (8)$$

If the condition satisfies, it outputs true, otherwise, it outputs false.

The $VKeyGen()$ algorithm generates both a homomorphic encryption key pair and a classical public–private key pair. The homomorphic encryption keys are used to encrypt the private masking values employed during local update generation, while the classical key pair is used only in the *Setup* phase to distribute an initial random seed hR^0 among nodes. The seed is subsequently used to derive the random nonce R in each *Learning* round.

The $VLocUpGen()$ algorithm constructs a verifiable local update $LU_i = \langle D_i, E_i, T_i \rangle$ for each node U_i based on its local model parameter w_i . The key difference from the MHESA scheme is that D_i is generated not only using the node-chosen masking value M_i , but also the random nonce R —shared exclusively among the nodes—and the secret S held by FS . This ensures that only FS , which holds S , can aggregate a valid global update, while only the nodes, which share R , can recover actual global parameter $w = \sum w_i$, which remains hidden from FS . In addition to D_i , each node also generates a verification token T_i corresponding to a selected subset of model parameters. Similarly to D_i , the token T_i is constructed using a simple masking technique; however, it additionally incorporates two nonces, R_1 and R_2 , which are derived from R . Consequently, no party without knowledge of R can generate a valid token. Moreover, T_i is constructed only for a small subset of parameters indexed by values randomly chosen by FS in each *Learning* round. This design significantly reduces computational and communication overhead by limiting the number of verification tokens that need to be additionally generated and transmitted.

The $VGlobUpGen()$ algorithm aggregates all local updates LU_i collected from the participating nodes in a given *Learning* round to produce the global update $GU = \langle GW, GT \rangle$. For the global model parameter w , the aggregated values are defined as $GW = R \cdot w$ and $GT = R_1 \cdot w[idx] + N_T \cdot R_2$. Here, only FS , which knows S , can compute $S \cdot \sum E_i = S \cdot \sum M_i$, and subsequently generate a valid value $R \sum w_i$. This ensures that no party, other than FS and legitimate nodes, can impersonate FS or fabricate $R \sum w_i$.

Each node executes the $Verf()$ algorithm to verify both the correctness and integrity of the global parameter w obtained from $GU = \langle GW, GT \rangle$. Integrity ensures that GW has not been tampered with during transmission, while correctness ensures that GW accurately represents the aggregation of all legitimate local updates provided by the participating nodes. Using the shared nonce R , each node derives w from GW and reconstructs the subvector $w[idx]$ corresponding to the randomly selected indices idx . The node then compares this value with $w'[idx] = (GT - N_T \cdot R_2) / R_1$. If the difference satisfies $\|w[idx] - w'[idx]\| < \epsilon$, for a predefined threshold ϵ , the global model w is considered valid. Although verification is performed only on a subset of parameters, the index set idx is randomly selected in each *Learning* round. Hence, if the selected subset satisfies the verification condition, the entire global model w is regarded as valid.

If the verification equation holds, it implies that both GW and GT are consistent values derived from the same global model w , thereby confirming that these values were not altered during transmission (integrity is ensured).

Furthermore, the pair $\langle GW, GT \rangle$ passes the verification procedure if and only if all legitimate local updates LU_i have been correctly collected and aggregated.

(1) When all $LU_i = \langle D_i, E_i, T_i \rangle$ are valid, FS homomorphically aggregate E_i to recover the sum of masks (using its secret S), and removes the combined mask term from $\sum D_i$ to obtain the blinded aggregate $GW = R \cdot w$. Simultaneously, FS computes the aggregated token GT for the selected indices. Because nodes know R and its derived random values R_1 and R_2 , they can recover w and the subvector $w[idx]$ from GW , and compare these values with GT after removing R_1 and R_2 . Since all values originate from correctly aggregated LU_i , the reconstructed $w[idx]$ and the value derived from GT coincide, and the verification succeeds.

(2) If forged updates $LU_f = \langle D_f, E_f, T_f \rangle$ that do not correctly incorporate R and S are used, the resulting $\langle GW_f, GT_f \rangle$ produced by FS fails to cancel the masks properly. Because the tokens do not embed the correct R_1 and R_2 , the two values $w_f[idx]$ and $w_{f'}[idx]$ derived from GW_f and GT_f , respectively, do not match, and the verification fails.

Therefore, when the verification equation holds, it guarantees that GW was produced as the aggregation of all legitimate local updates provided by the participating nodes. The unforgeability of constructing a valid pair $\langle GW_f, GT_f \rangle$ without knowledge of R is formally analyzed in Section 4.1.

3.4.2. Federated Learning Using V-MHESA

Based on the algorithms proposed above, FS and each node collaboratively perform federated learning as follows:

At the *Setup* phase, FS collaboratively establishes system parameters and keys with each node.

1. FS generates system parameters $\langle PKParams, HEPParams \rangle$, its public key pair $\langle K^+, K^- \rangle$, its master secret S , and the initial global model update $GW^0 = w^0$, and then publishes public parameters $\langle PKParams, HEPParams, K^+, GW^0 \rangle$ to all nodes in U . Here, w^0 is randomly generated by FS .
2. Each U_i generates keys and commitment $\langle K_i^+, K_i^-, pk_i, sk_i, c_i \rangle \leftarrow VKeyGen(PKParams, HEPParams)$ and sends only public parameters $\langle K_i^+, pk_i, c_i, x_i \rangle$ to FS .
3. FS sets $K = \{K_i^+\}$, $PK = \{pk_i\}$, $C = \{c_i\}$, $X = \{x_i\}$, for all U_i in U , and creates a random identifier ID_i for each U_i . Then, FS generates $CS_i = PKE(K_i^+, S)$ and $CID_i = PKE(K_i^+, ID_i)$. FS selects a representative node U_R , which can be selected randomly or sequentially among nodes. FS sends K to U_R .
4. U_R chooses a random seed hR^0 , which can be used to generate a random nonce at every single *Learning* round, and generates $CR_i = PKE(K_i^+, hR^0)$ and $Sig = DS(K_R^-, H(hR^0))$. U_R sends back $\langle CR = \{CR_i\}, Sig \rangle$ to FS .
5. FS sends $\langle K, CR_i, Sig, CS_i, CID_i \rangle$ to each U_i .
6. Each U_i obtains $S = PKD(K_i^-, CS_i)$, $hR^0 = PKD(K_i^-, CR_i)$, and $ID_i = PKD(K_i^-, CID_i)$, and determines if $VS(K_R^+, Sig, hR^0)$ holds, send “valid” to FS .

If all nodes agree to “valid”, the *Setup* phase completes, otherwise, starts again from step 4.

Once the *Setup* phase is completed, each U_i carries out the initial local learning with w^0 to obtain w_i^1 . Then, FS and nodes repeat the subsequent *Learning* phases until the federated learning process is concluded. At the t -th *Learning* iteration ($t > 0$), FS interacts with each U_i as follows:

1. FS randomly generates a weight index set idx^t and a round nonce RN^t and broadcasts the t -th “ROUND START” message along with $\langle idx^t, RN^t \rangle$ to all nodes in U .
2. Each U_i generates its authentication code $AC_i = H(ID_i \parallel RN^t)$ and responds $\langle AC_i, x_i \rangle$ to FS .
3. For each AC_i , FS verifies whether $H(ID_i \parallel RN^t) = AC_i$ holds. FS then forms the t -th node group U_T consisting of all nodes with valid authentication codes. (If an invalid authentication code is detected, FS may re-request a new authentication code from the corresponding node.) Once U_T is determined, FS generates the group parameters $\langle PK_T, C_T, X_T \rangle \leftarrow GroupKeyGen(PK, C, U_T)$, and broadcasts them to all nodes in U_T .
4. Each U_i generates the t -th random nonces $hR^t = PRG(hR^{t-1})$ and $R^t = PRG(hR^t \parallel RN^t)$, and then constructs its local updates $LU_i^t = \langle D_i^t, E_i^t, T_i^t \rangle \leftarrow VLocUpGen(w_i^t, R^t, x_i, X_T, PK_T, C_T, sk_i, v_i, S, idx^t)$ for the t -th local parameter w_i^t . Each U_i sends LU_i^t to FS .

5. FS generates the t -th global update $GU^t \leftarrow VGlobUpGen(LU_T, U_T)$, and distributes it to all nodes in U_T .
6. Each U_i verifies the validity of GU^t by performing $Verf(GU^t, R^t, idx^t)$. If the verification succeeds, it responds with “valid”; otherwise, it responds with “invalid” to FS .
7. If all nodes respond with “valid,” FS broadcasts a “ROUND TERMINATION” message to all nodes in U , after which each node performs its local learning step using w^t to obtain the next local parameter w_i^{t+1} . If more than half of the nodes respond with “invalid,” FS restarts the t -th iteration with newly generated idx^t and RN^t , beginning from Step 1, and broadcasts a “RESTART” message to all nodes in U . Otherwise, if only a minority of nodes return “invalid,” FS retransmits GU^t to those of nodes and repeats Step 6.

In V-MHESA, all nodes must share a fresh random nonce R^t in every *Learning* round. To achieve this, during the *Setup* phase, a designated node U_R generates an initial random seed hR^0 and securely distributes it to all other nodes. The initial generator U_R may be arbitrarily chosen by FS , selected as the node with the highest computational capability, or rotated among the nodes across successive rounds. The initial seed hR^0 is encrypted under the public key K_i^+ of each node, ensuring that only the intended recipients can decrypt it, while FS remains unaware of its value. Furthermore, hR^0 is accompanied by a digital signature from U_R , allowing each node to verify the authenticity of the decrypted value and preventing any tampering by FS . After the *Setup* phase, each node independently updates the nonce by repeatedly hashing it across rounds, e.g., $hR^{t+1} = PRG(hR^t)$, thereby generating a new $R^{t+1} = PRG(hR^{t+1} \parallel RN^{t+1})$ in every *Learning* phase. This property enables nodes that missed previous rounds to rejoin the protocol later while still generating a valid R^t .

For node authentication, FS assigns a random identifier ID_i to each node U_i , which is transmitted in encrypted form under the node’s public key K_i^+ . Thus, only the designated node can decrypt and retrieve its identifier. At the beginning of each *Learning* round, FS constructs the aggregation group of participating nodes and uses these identifiers to verify that each joining node is legitimately registered.

The *Learning* phase is enhanced with three additional steps: (i) node authentication, (ii) random nonce renewal, and (iii) verification of the global update. At the start of t -th round ($t > 0$), FS selects a random index set idx for verification and generates a round nonce RN^t , both of which are broadcast to all nodes. Each node computes its authentication code $AC_i = H(ID_i \parallel RN^t)$ using ID_i and RN^t , and sends it back to FS . FS verifies each response by recomputing the same hash with the corresponding ID_i and RN^t . Through this lightweight check, FS can confirm both the legitimacy of each responding node and its correct reception of RN^t . This mechanism prevents unregistered or malicious external nodes from impersonating legitimate participants and submitting invalid updates. FS then forms the active node group U_T using only those nodes that returned valid authentication codes and generates the corresponding group keys, which are distributed to the group members.

Using the refreshed R^t , each node constructs and transmits its local update LU_i^t to FS . FS aggregates all received local updates to produce the global update GU^t . Each node then verifies the validity of GU^t , only if the verification succeeds do the nodes update their local models with the new global parameter w^t . If any node reports an invalid result, the protocol restarts from Step 1 by default. However, since invalidity may arise from transmission errors rather than an incorrect GU^t , in practice the restart is triggered only when more than half of the nodes report invalid results. If fewer than half respond with invalid results, FS retransmits GU^t exclusively to those nodes for re-verification.

3.4.3. Dropout Robustness

During the aggregation of local updates, *FS* may encounter dropouts. *FS* waits for node submissions within a predefined timeout period: any local update not received within this period is treated as a dropout. If a node drops out and its local update is not delivered to *FS*, the sum of the masking values chosen by the nodes cannot be correctly reconstructed. As a result, the masking values cannot be removed, and a valid global update cannot be generated.

To address this, *FS* restarts the *Learning* process from Step 1 with the remaining nodes, excluding the dropout nodes. Importantly, a new set of random values must be used for the re-aggregation. For example, suppose a dropout occurs at Step 5 of the t -th *Learning* round. In that case, *FS* distributes a new pair $\langle idx^{t'}, RN^{t'} \rangle$ to the remaining nodes. *FS* also generates fresh group parameters $\langle PK_T', C_T', X_T' \rangle$ corresponding to the updated node group U_T' . Each node also generates a new nonce $R^{t'}$ and masking value $M^{t'}$, constructs a new local update $LU_i^{t'}$. Finally, *FS* aggregates these inputs to compute the revised global update $GU^{t'}$.

When a dropout occurs, two cases must be considered. The first is when dropped-out data are completely lost and never delivered to *FS*. The second is when the data are not fully lost but instead arrive at *FS* with significantly delay. The essential distinction between these two cases is whether *FS* ultimately obtains the local update LU_d^t from U_d identified as a dropout.

The second case poses a potential risk: *FS* may gain access to (i) all local updates LU_i^t from the original node group U_T , (ii) the new local updates $LU_i^{t'}$ from the reduced node group U_T' (excluding the dropout), and (iii) the corresponding global updates $GU^t = VGlobUpGen(LU_T, U_T)$ and $GU^{t'} = VGlobUpGen(LU_T', U_T')$. In the original MHESA model, since *FS* could compute the actual global model parameter w^t , the occurrence of a single dropout node U_d led to a privacy risk: by comparing the global parameter w^t including U_d with the parameter $w^{t'}$ excluding U_d , *FS* could derive the local parameter w_d^t of U_d as $w_d^t = w^t - w^{t'}$. To mitigate this, the original scheme required that, whenever a single dropout occurred, one additional node be excluded from the *Learning* round to prevent parameter exposure.

In contrast, the proposed V-MHESA scheme avoids this vulnerability. Instead of computing the true global parameter w^t , *FS* generates only the masked global update $GU^t = \langle GW^t, GT^t \rangle$, where $GW^t = R^t \cdot w^t$. Moreover, during re-aggregation caused by dropouts, R^t is refreshed. As a result, *FS* gains no additional information about either the final model parameter or any individual local parameters. Even if a single dropout node later submits its delayed update, *FS* cannot infer its local information. Thus, unlike the original MHESA scheme, V-MHESA does not require the exclusion of any additional nodes and provides a more practical and secure approach to handling dropout scenarios.

4. Analysis

4.1. Security

First, we analyze the security of the proposed model under the honest-and-curious threat model. We focus on the primary security properties: (i) the confidentiality of local parameters, ensuring that no party—including *FS*—can learn any individual node's local model parameters, and (ii) the unforgeability of the global update, guaranteeing that an adversary cannot construct a counterfeit pair $\langle GW_f, GT_f \rangle$ that passes the verification procedure without possessing the required secret R .

We consider a federated learning system consisting of a single *FS* and at least three participating nodes. Each node communicates solely with *FS*. Both *FS* and the nodes are assumed to follow the protocol honestly while keeping their private keys secure;

however, they may attempt to infer additional information from legitimately obtained data (honest-but-curious). Local updates are transmitted exclusively to FS and are never shared among nodes. External adversaries may eavesdrop on all communications over the public network but cannot compromise secret keys or cryptographic primitives under standard hardness assumptions.

4.1.1. Confidentiality of Local Update

In the proposed scheme, each node's original local parameter w_i is masked using a random nonce R , which is shared only among the nodes, and a randomly chosen mask M_i . M_i is encrypted separately using CKKS-HE scheme (assumed IND-CPA secure).

In V-MHESA, FS 's secret S is additionally embedded in the construction of each masked local parameter D_i . This design ensures that only FS can remove the aggregated masking terms from $\sum D_i$ to obtain the blinded value $GW = R \cdot w$, while the true global parameter w remains hidden. Moreover, no external entity can infer a valid GW from the transmitted local updates before FS publishes the global update pair $\langle GW, GT \rangle$ to the nodes.

Intuitively, the leakage set $\langle D_i, E_i, GW \rangle$ should reveal no non-negligible information about w_i , since w_i is multiplicatively blinded by the node-only nonce R , and the additive mask M_i is protected by IND-CPA secure homomorphic encryption. We exclude $\langle T_i, GT \rangle$ from the leakage set, as these are used solely for verification and are generated in the same randomized manner as D_i and GW . In particular, T_i incorporates additional random values R_1 and R_2 derived from the shared nonce R ; therefore, $\langle T_i, GT \rangle$ provides no further distinguishing advantage beyond what is already possible from $\langle D_i, E_i, GW \rangle$.

We thus analyze confidentiality with respect to three adversarial perspectives:

- External eavesdropper ($\mathcal{A}_{ext}$): This adversary passively observes all messages transmitted over the network, i.e., $\{\langle D_i, E_i \rangle\}_i$ and GW , but does not know R , S or any M_i .
- Federated server ($\mathcal{A}_{fs}$) (honest-but-curious): The federated server possesses S and receives all $\langle D_i, E_i \rangle$ values. It computes and stores GW , yet it still does not know R and any individual M_i .
- Malicious node ($\mathcal{A}_m$): In our threat model, each node is assumed to be honest-but-curious; thus, a node knows only the published GW and cannot access $\langle D_j, E_j \rangle$ of other nodes U_j . However, to strengthen the confidentiality argument, we also consider a hypothetical malicious node that, in addition to knowing R and S , can observe all $\langle D_i, E_i \rangle$ exchanged over the network, yet it still does not know any individual M_i . We show that even under this strongest assumption, the leakage set $\langle D_i, E_i, GW \rangle$ does not reveal any non-negligible information about w_i .

To prove confidentiality, we define the confidentiality game **ConfGame** with the leakage set $Leak = \langle D_i, E_i, GW \rangle$ as follows:

Definition 1. (ConfGame).

1. *Setup*: The challenger generates the system parameters and provides the server secret S to the simulator used in reduction. The secret S is shared between the nodes and FS but is not used to mask w_i ; rather, it serves only for server authentication, ensuring that only the legitimate FS can produce a valid GW . The node-shared nonce R and each node's mask M_i remain hidden from the adversary.
2. *Challenge queries*: The adversary $\mathcal{A}$ selects any two candidate local parameters w^0, w^1 that share the same dimension and domain. The challenger flips a random bit $b \leftarrow \{0, 1\}$, sets $w_i = w^b$, and executes the honest V-MHESA generation to obtain $\langle D_i, E_i, GW \rangle$ along with all other honest nodes' outputs for that round. The challenger then returns the leakage set $Leak = \langle D_i, E_i, GW \rangle$ to $\mathcal{A}$.

3. *Guess*: $\mathcal{A}$ outputs a guess b' . Its advantage is $\text{Adv}_{\mathcal{A}} = \left| \Pr[b = b'] - \frac{1}{2} \right|$.

V-MHESA is said to ensure confidentiality of w_i under leakage $\langle D_i, E_i, GW \rangle$ if every PPT adversary $\mathcal{A} \in \{\mathcal{A}_{\text{ext}}, \mathcal{A}_{fs}, \mathcal{A}_m\}$ has negligible advantage.

Theorem 1. Assume CKKS-HE is IND-CPA secure. Then, for any PPT adversary $\mathcal{A}$ (either $\mathcal{A}_{\text{ext}}$, $\mathcal{A}_{fs}$ or $\mathcal{A}_m$), the advantage in ConfGame is negligible in the security parameter. In particular, the leakage set $\langle D_i, E_i, GW \rangle$ is computationally indistinguishable when $w_i = w^0$ versus $w_i = w^1$, up to the public CKKS approximation tolerance ϵ .

Proof. We establish indistinguishability using a standard hybrid-game argument.

Hybrid H0 (Real World): The challenge executes V-MHESA honestly with $w_i = w^b$. The adversary receives $\text{Leak} = \{D_i, E_i, GW\}$ and all public values consistent with that round. Recall that $D_i = R w_i + S M_i$, $E_i = \text{Enc}(M_i)$, and $GW = R \sum_j w_j$.

Hybrid H1 (Mask-Ciphertext Replacement): Replace each ciphertext $E_i = \text{Enc}_{\text{CKKS}}(M_i)$ with an encryption $E'_i = \text{Enc}_{\text{CKKS}}(\tilde{M}_i)$ of a freshly sampled random mask $\tilde{M}_i$. By the IND-CPA security of CKKS-HE, $\mathcal{A}$ cannot distinguish between H0 and H1 except with negligible probability.

Hybrid H2 (Mask Re-sampling): The additional term $S \cdot M_i$ within D_i acts as a high-entropy randomizer independent of w_i from the adversary's perspective (the value M_i is hidden from CKKS from any $\mathcal{A} \in \{\mathcal{A}_{\text{ext}}, \mathcal{A}_{fs}, \mathcal{A}_m\}$, even for $\mathcal{A}_{fs}$ only the sum $\sum_i M_i$ matters for removal). Replace every M_i in D_i with a freshly sampled $\tilde{M}_i$ drawn from the same distribution as in the real protocol, keeping E'_i consistent with $\tilde{M}_i$ as in H1. Because R multiplicatively blinds w_i and the additive term $S \cdot \tilde{M}_i$ statistically mask any residual correlation, the joint distribution of $\langle D'_i, E'_i \rangle$ remains indistinguishable from the real one, up to negligible CKKS approximation error.

Hybrid H3 (Simulation): The simulator, given S from the challenger, generates the leakage set $\langle \tilde{D}_i, E'_i, GW \rangle$ as follows: It samples encryption key sk_i and generates $E'_i = \text{Enc}_{\text{CKKS}}(\tilde{M}_i)$. It randomly samples $\tilde{D}_1, \dots, \tilde{D}_{n-1}$ matching the real-protocol numerical scale and tolerance of CKKS outputs. Then, it fixes $\tilde{D}_n = GW + S \sum_{i=1}^n \tilde{M}_i - \sum_{i=1}^{n-1} \tilde{D}_i$ so that the global sum satisfies $\sum_{i=1}^n \tilde{D}_i = GW + S \cdot \sum_{i=1}^n \tilde{M}_i$.

Because both $\mathcal{A}_{fs}$ and $\mathcal{A}_m$ know S , they can verify that $\sum_i D_i - S \cdot \sum_i E'_i = GW$ holds exactly (up to ϵ). Hence, the simulated transcript is perfectly consistent with an honest FS computation.

In H3, the joint distribution of $\{\tilde{D}_i, E'_i, GW\}$ is computationally indistinguishable from that of H0, while being independent of whether $w_i = w^0$ or $w_i = w^1$.

If there exists a PPT adversary $\mathcal{A}$ that could distinguish the real transcript $(D_1, \dots, D_n)$ from the simulated tuple $(\tilde{D}_1, \dots, \tilde{D}_n)$ with non-negligible advantage, then, $\mathcal{A}$ would necessarily break at least one of the following assumptions: (1) IND-CPA security of CKKS-HE (between H0 and H1), (2) statistical hiding of mask re-sampling (between H1 and H2), and (3) indistinguishability under linear constraint (between H2 and H3).

Even the strongest adversary $\mathcal{A}_m$ knowing both R and S , can compute $R \cdot w^0$ and $R \cdot w^1$ for its chosen w^0, w^1 , but it still observes only $D_i = R \cdot w^b + S \cdot M_i$. To distinguish D_i from $\tilde{D}_i$ with non-negligible advantage, it would need to recover M_i from E_i , contradicting the IND-CPA security of CKKS-HE. Since the statistical distributions of D_i and $\tilde{D}_i$ coincide up to negligible approximation error, any distinguisher $\mathcal{A}$ leads to a contradiction. Therefore, the adversary's advantage in **ConfGame** is negligible. $\square$

Corollary 1. *If at least three participating nodes exist, the confidentiality of local parameters is preserved against $\mathcal{A}_m$.*

Proof. Unlike $\mathcal{A}_{fs}$ and $\mathcal{A}_{ext}$, which have no knowledge of R , $\mathcal{A}_m$ knows both R and S . When only two nodes participate in the aggregation— $\mathcal{A}_m$ and another node U_j — $\mathcal{A}_m$ can derive the local parameter of U_j as follows, since $\mathcal{A}_m$ can observe the local update $LU_j = \langle D_j, E_j, T_j \rangle$. Let the local update of $\mathcal{A}_m$ be $LU_m = \langle D_m, E_m, T_m \rangle$. Then $\mathcal{A}_m$ can compute the total mask $M_{sum} = M_j + M_m = E_j + E_m$, and consequently obtain $M_j = M_{sum} - M_m$. Since $\mathcal{A}_m$ knows both R and S , it can recover w_j of U_j as $w_j = (D_j - S \cdot M_j) / R$. Therefore, if $\mathcal{A}_m$ is one of only two participating nodes, it can infer the local parameters of the other node, violating confidentiality. To ensure the confidentiality of local updates even against $\mathcal{A}_m$, there must exist at least two honest (non-colluding) nodes other than $\mathcal{A}_m$. When at least three nodes participate in the aggregation, $\mathcal{A}_m$ can only obtain the sum of the other nodes' masking values $\sum_{j \neq m} M_j$, without learning any individual M_j . Consequently, the confidentiality of each node's local parameters is preserved. $\square$

4.1.2. Unforgeability of Global Update

In this section, we analyze the unforgeability of the global update pair $\langle GW, GT \rangle$. An adversary who does not know the node-shared nonce R can pass the verification if and only if it can construct a forged pair $\langle GW_f, GT_f \rangle$ that satisfies the verification relation used by the nodes.

Recall that nodes accept $\langle GW, GT \rangle$ if, after removing the known values R_1 and R_2 , the values derived from GW and GT coincide for the selected index set. Equivalently, denoting by $w[idx]$ the subvector at the chosen indexes, the verification condition can be written as

$$\frac{GW[idx]}{R} = \frac{GT - N_T \cdot R_2}{R_1},$$

or, rearranged,

$$GW[idx] = (GT - N_T \cdot R_2) \cdot R_3, \text{ where } R_3 = R / R_1$$

Therefore, any forged pair $\langle GW_f, GT_f \rangle$ that passes verification must satisfy

$$GW_f[idx] = (GT_f - N_T \cdot R_2) R_3.$$

If an adversary does not know R , it cannot compute R_1 , R_2 and R_3 . Under the usual assumption that the pseudo random generator behaves as a random oracle, the adversary's only option is to guess the values of R_2 and R_3 . Let m denote the bit-length of each of R_2 and R_3 (i.e., $|R_2| = |R_3| = m$). The probability that an adversary correctly guesses R_2 and R_3 is therefore at most 2^{-2m} . Consequently, the probability that the adversary can produce a valid forged pair $\langle GW_f, GT_f \rangle$ without knowledge of R is negligible and bounded by

$$\Pr[\langle GW_f, GT_f \rangle] \leq 2^{-2m}.$$

4.2. Experimental Evaluation

In this section, we evaluate the learning accuracy and computational efficiency of the proposed V-MHESA-based federated learning model using the MNIST dataset. We analyze performance under varying number of participating nodes, considering both IID and Non-IID data distribution across the nodes.

For learning accuracy, we compare the classification performance of V-MHESA with two baselines: (i) a conventional centralized training model and (ii) our previous MHESA scheme. This comparison allows us to examine the introduction of verification mechanisms in V-MHESA impacts model accuracy. In addition, we quantify the discrepancy between the global model parameters obtained through the direct aggregation of weighted local parameters and those produced by V-MHESA, thereby evaluating the effectiveness of the proposed scheme.

For computational efficiency, we analyze the execution time required for each step of the proposed protocol on both the node side and the server side. Furthermore, we assess the additional computational overhead introduced by the verification mechanism, providing a detailed evaluation of its practical feasibility.

4.2.1. Experimental Environment

For the experiments, we employed two systems implementing the federated server and clients. The server system is equipped with an Intel(R) Core(TM) i7-12700K CPU, Intel(R) UHD Graphics 770 GPU, NVIDIA GeForce RTX 3080 GPU, and 32 GB of RAM. The client system is configured with an Intel(R) Core(TM) i9-7920X CPU, two NVIDIA GeForce GTX 1080Ti GPUs, and 32 GB of RAM. On the client system, nodes were implemented as independent threads, thereby ensuring identical computing power and communication conditions across nodes. Both systems were connected within the same local network with a bandwidth of 1 Gbps; hence, the analysis on actual communication efficiency including communication latency is not explicitly considered in our analysis.

For local training at each node, we employed a two-layer CNN model with 5×5 convolution layers (the first with sixteen channels, the second with thirty-two channels, each followed with 2×2 max pooling). ReLU activation functions were applied after each convolution, and a final softmax layer was used for classification. Local learning at the client side was implemented using the PyTorch 2.9.1 framework, while the federated learning operations on both sides were implemented in C++ with open-source CKKS libraries. The MNIST dataset, comprising 60,000 training images and 10,000 test images, was used in the experiments. In the IID setting, data samples were evenly distributed across the nodes, whereas in the non-IID setting, the dataset size allocated to each node varied, resulting in heterogeneous local training times and update latencies, thereby effectively simulating heterogeneous nodes.

The server and nodes executed 100 rounds of the *Learning* phase (i.e., global model updates) to complete one federated learning process. To achieve a 128-bit security level for CKKS-HE, we set the degree of the ring polynomial $n = 2^{16}$, the decryption modulus to $q = 800$ bits, and the encryption modulus to $Q = 1600$ bits. For each experimental setting, the reported results represent the average of five independent runs. Table 3 summarizes the experimental parameters and corresponding values used in our evaluation.

Table 3. Experimental parameters and values.

Parameters	Values
The number of nodes (N)	1, 10, 20, 50, 100
The number of rounds for federated learning	100
The number of weight parameters	21,840
The degree n of a ring polynomial	2^{16} (= 65,536)
The size of q as the modulus for decryption (bits)	800
The size of Q as the modulus for encryption (bits)	1600

Table 4 shows the size of data allocated to nodes according to the number of nodes in both distributions.

Table 4. The size of data allocated to nodes.

Number of Nodes	Data Size per Node		
	IID Distribution	Non-IID Distribution	
		Minimum	Maximum
1	60,000	60,000	
10	6000	400	10,200
20	3000	800	6000
50	1200	100	2300
100	600	100	1200

4.2.2. Experimental Results

We first evaluate the learning accuracy of V-MHESA. The analysis is conducted from four perspectives: (1) accuracy across training rounds, (2) accuracy with varying number of nodes, (3) accuracy under different data distributions (IID vs. non-IID), and (4) error rate analysis of the global model parameters generated by the proposed scheme.

For the round-based accuracy comparison, we compare the accuracy of V-MHESA with that of a baseline model that aggregates raw local parameters, as well as with our previous MHESA scheme. For node-scaling analysis, we examine how accuracy changes as the number of participating nodes increases, and compare the results with both a single-node (centralize) learning model and the MHESA model. Regarding the data distribution analysis, in the IID setting, all nodes are assigned training datasets of equal size, whereas in the non-IID setting, nodes hold training datasets of varying sizes. This heterogeneity may lead to differences in aggregation accuracy. We therefore compare the performance of V-MHESA against MHESA under both IID and non-IID scenarios. Finally, we analyze the parameter error rate between the global model parameters obtained by direct aggregation of weighted local updates and those generated by V-MHESA, to assess the validity and utility of the proposed verification enhanced scheme.

Table 5 presents the comparison of learning accuracy across training rounds. The objective of this analysis is to examine the accuracy difference between the global model obtained through the aggregation of masked local updates and the global model generated by directly aggregating raw local parameters. To ensure fairness across all models, the experimental setting was unified by using 10 nodes with data distributed under the IID assumption, which yielded the highest accuracy among the tested configurations. The results show that the three models exhibit almost identical accuracy trends across rounds. This indicates that aggregating encrypted local parameters in the proposed scheme does not compromise the accuracy of global model.

Table 5. Average accuracy comparison of the V-MHESA model, the raw data aggregation model, and the MHESA model (for $N = 10$, IID-setting).

Round	Accuracy of Raw Data Aggregation (%)	Accuracy of MHESA (%)	Accuracy of V-MHESA (%)
10	91.27	91.28	91.27
20	94.32	94.33	94.18
40	96.36	96.36	96.37
60	97.09	97.08	97.06
80	97.43	97.42	97.50
100	97.74	97.75	97.74

Table 6 reports the learning accuracy of V-MHESA compared with a single centralized server model and the previous MHESA scheme under the IID setting, while Table 7 presents the corresponding results under the Non-IID setting. As shown in Table 6, the single centralized model, which processes all data collectively, achieves the highest accuracy. In contrast, the accuracy of both V-MHESA and MHESA decreases as the number of nodes increases, since each node receives a smaller portion of the training dataset. For example, in the IID setting, where data is evenly distributed among nodes, the single server model achieved 99.16% accuracy after 100 rounds, while V-MHESA reached 97.74% with 10 nodes but dropped to 90.55% when the number of nodes increased to 100. Nevertheless, the accuracy difference between V-MHESA and MHESA remains negligible across all cases.

Table 6. Average accuracy under IID distribution across different numbers of nodes.

Average Accuracy (%), IID									
<i>N</i>	<i>N</i> = 1	<i>N</i> = 10		<i>N</i> = 20		<i>N</i> = 50		<i>N</i> = 100	
Round		MHESA	V-MHESA	MHESA	V-MHESA	MHESA	V-MHESA	MHESA	V-MHESA
10	97.69	91.27	91.27	84.29	87.27	53.65	53.86	16.26	16.41
20	98.35	94.27	94.18	91.19	91.18	76.70	76.43	47.21	46.48
40	98.78	96.33	96.37	94.27	94.37	89.59	89.58	71.77	71.57
60	98.94	97.08	97.06	95.54	95.49	92.10	91.93	85.77	85.46
80	99.02	97.53	97.50	96.31	96.25	93.35	93.41	88.65	88.63
100	99.16	97.74	97.74	96.73	96.73	94.13	94.09	90.39	90.55

Table 7. Average accuracy under Non-IID distribution across different numbers of nodes.

Average Accuracy (%), Non-IID									
<i>N</i>	<i>N</i> = 1	<i>N</i> = 10		<i>N</i> = 20		<i>N</i> = 50		<i>N</i> = 100	
Round		MHESA	V-MHESA	MHESA	V-MHESA	MHESA	V-MHESA	MHESA	V-MHESA
10	97.69	87.96	83.54	87.47	87.27	66.98	62.17	35.55	26.60
20	98.35	92.15	89.27	91.19	92.12	86.15	85.30	66.09	69.57
40	98.78	94.72	92.78	95.05	94.92	91.27	91.40	84.70	85.83
60	98.94	95.68	94.23	96.21	96.05	93.06	93.06	88.9	89.12
80	99.02	96.28	95.18	96.67	96.61	94.13	94.08	90.74	90.66
100	99.16	96.7	95.72	97.05	97.02	94.93	94.72	91.79	91.80

Table 7 illustrates the results under the Non-IID setting, where the distribution of training data varies significantly across nodes. Here, V-MHESA achieved 95.72% accuracy with 10 nodes, slightly lower than the 97.74% accuracy under the IID setting. Interestingly, with 100 nodes, the accuracy was 91.80%, slightly higher than the 90.55% observed in the IID setting. This behavior occurs because, unlike the IID case where increasing the number of nodes inevitably reduces the dataset size per node, in the Non-IID case some nodes retain relatively larger datasets and thus produce stronger local models, which positively influence the aggregated global model. Similarly to the IID experiments, the accuracy gap between V-MHESA and MHESA in the Non-IID setting is negligible. Figure 2 shows that, as the number of nodes increases, the accuracy in the Non-IID setting slightly improves. Up to 20 nodes, the accuracy in the IID setting is marginally higher or nearly identical to that of the Non-IID case. However, when the number of nodes reaches 50 or more, the Non-IID setting outperforms the IID setting in terms of accuracy.

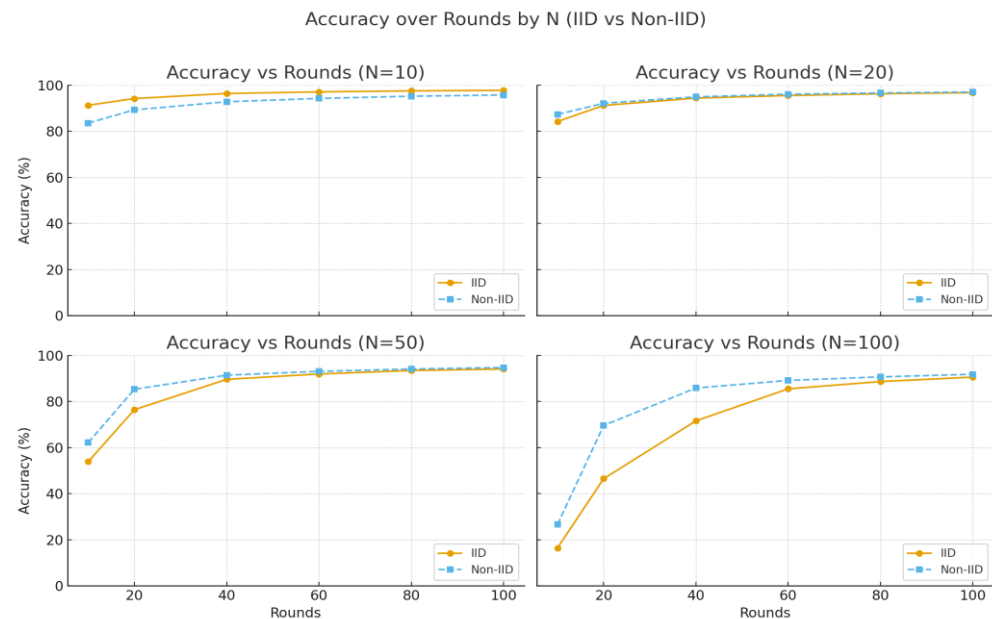


Figure 2. Accuracy trends with varying numbers of nodes under IID and Non-IID settings.

Overall, while the global model accuracy tends to decrease as the number of nodes increases—primarily due to the reduced training data size per node—the comparison between V-MHESA and MHESA demonstrates that the introduction of verification functionality in V-MHESA does not adversely affect learning accuracy.

In the proposed model, the generation of local updates incorporates not only the masking value selected by each node but also the server's secret S and the random nonce R shared exclusively among the nodes. To evaluate the accuracy of the global parameter reconstructed from masked local updates, we compared the global parameter w obtained by directly aggregating the raw local parameters without masking and the global parameter w' derived from aggregating the masked local parameters. The resulting error analysis between w and w' is presented in Figure 3. Furthermore, one of the core functionalities of V-MHESA is the verification of the global model. For the index set idx selected for verification, we compared the aggregated values of the actual local parameters, denoted as $w[idx]$, with the values $w'[idx]$ reconstructed by each node from the global verification token GT provided by FS . The error analysis between $w[idx]$ and $w'[idx]$ is presented in Figure 4.

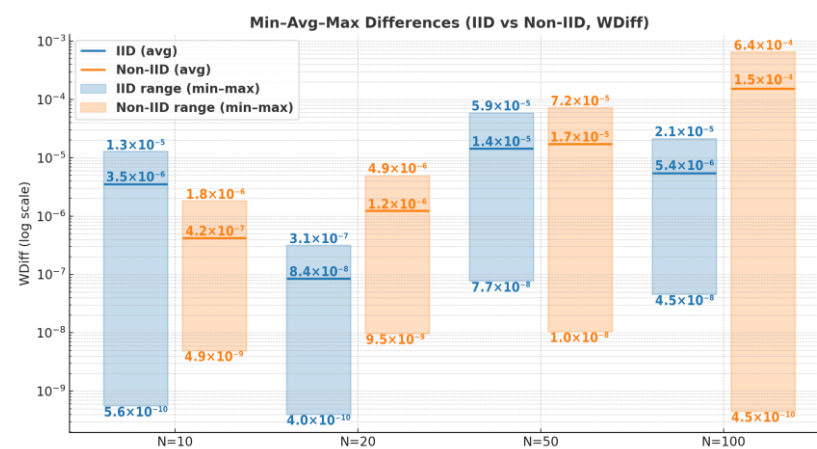


Figure 3. Error analysis (minimum, maximum, and average) between the global parameter w obtained from raw aggregation and w' obtained from masked aggregation in V-MHESA.

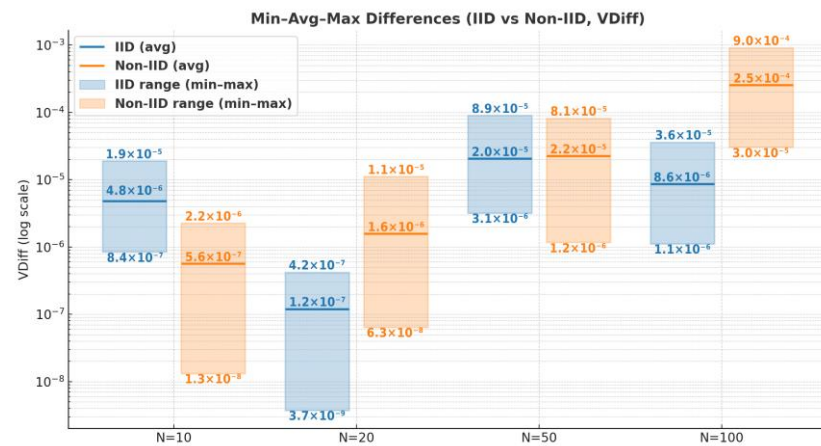


Figure 4. Error analysis (minimum, maximum, and average) between the aggregated local parameters $w[idx]$ and the reconstructed values $w'[idx]$ from the global verification token GT.

As shown in Figure 3, in the IID setting, the error of the global model parameter w' ranges from a minimum of 4.0×10^{-10} to a maximum of 5.9×10^{-5} , with an average error of 5.8×10^{-6} . In contrast, in the Non-IID setting, the error ranges from a minimum of 4.49×10^{-10} to a maximum of 6.44×10^{-4} , with an average error of 4.29×10^{-5} . The larger deviation observed in the Non-IID case is attributed to differences in the local dataset sizes across nodes, which lead to greater variations in the training rate (x_i/X) of local parameters and consequently larger error fluctuations. Nevertheless, in all cases, the maximum error remains below 6.5×10^{-4} , which does not significantly affect the accuracy of global model updates.

Figure 4 presents the error rate of $w'[idx]$ derived from the global verification token GT. Since this analysis reflects only the subset of parameters corresponding to the randomly selected indices idx , the error deviations are smaller than those in Figure 3, though the extent of variation depends on which parameters are selected. In the IID setting, the minimum error is 3.74×10^{-9} , the maximum error is 8.93×10^{-5} , and the average error is 8.47×10^{-6} . In the Non-IID setting, the minimum error is 1.3×10^{-8} , the maximum error is 8.98×10^{-4} , and the average error is 6.93×10^{-5} . Across all cases, the maximum error remains below 9.0×10^{-4} , indicating that the use of verification tokens does not impair the accuracy of the global model.

To evaluate the efficiency of V-MHESA, we analyze both the transmitted data size and the runtime overhead across the principal stages of the protocol. In the *Setup* phase, the key operations are (i) generating cryptographic keys for homomorphic-encryption-based aggregation between the server and the nodes, and (ii) distributing the initial random seed hR^0 among the nodes. Since V-MHESA inherits the homomorphic encryption framework of MHESA, the key generation procedure remains identical to that of MHESA. The latter operation, however, is newly introduced in V-MHESA: a randomly selected node U_R samples a random value and encrypts it under each peer's public key for secure distribution.

In the *Learning* phase, the essential operations are local-update generation at each node, server side aggregation, and per-round verification. Compared with MHESA, V-MHESA introduces two additional tasks: (i) the generation of a verification token during local update construction, and (ii) the verification of the global update on the node side. Our efficiency analysis therefore focuses on these extensions, quantifying the incremental communication incurred by transmitting verification tokens and additional ciphertexts, and the incremental computation required for token generation and for executing the verification procedure at both the nodes and the server.

As shown in Table 8, the sizes of the additional parameters introduced by the verification functionality in V-MHESA are relatively small. Although public-key ciphertexts

are added and their size grows proportionally with the number of nodes, this overhead occurs only once during the *Setup* phase and thus does not affect the overall efficiency of the protocol. In the repeatedly executed *Learning* phase, additional messages are generated and transmitted, including the random index set for verification, the round nonce, the corresponding authentication code, verification tokens, and the global verification token. While the communication size increases proportionally with the size of the random index set, the added overhead is negligible compared with the dominant payloads, namely the masked local parameters D_i (174, 720 B) and the ciphertexts of the mask values E_i (12.5 MB). Therefore, the use of verification tokens introduces only minimal communication overhead in practice.

Table 8. Data sizes of additional parameters introduced in V-MHESA.

Phase	Communication	Parameters	Description	Size (Byte)
Setup	$U_i \rightarrow FS$	K_i^+	Public key of U_i	32
	$FS \rightarrow U_i$	$K = \{K_i^+\}$	List of public keys	$32 \cdot N$
	$U_R \rightarrow FS$	$CR = \{CR_i\}$	Ciphertexts of hR^0	$56 \cdot N$
	$U_R \rightarrow FS, FS \rightarrow U_i$	Sig	Digital signature of U_R	72
	$FS \rightarrow U_i$	CR_i, CS_i, CID_i	Ciphertexts of hR^0 , S and ID_i	56
Learning	$FS \rightarrow U_i$	idx	Set of random indexes	$8 \cdot idx $
	$FS \rightarrow U_i$	RN^t	Round nonce	8
	$U_i \rightarrow FS$	AC_i	Authentication code	32
	$U_i \rightarrow FS$	T_i	Verification token	$8 \cdot idx $
	$FS \rightarrow U_i$	GT^t	Global verification token	$8 \cdot idx $

Table 9 compares the average execution times of the main operations in the V-MHESA model with those in the MHESA model under the Non-IID setting with 100 nodes. Since both models employ CKKS-HE, the homomorphic operations dominate the computational cost. In V-MHESA, the additional procedures required for generating verifiable local updates (including verification tokens), server authentication, and node authentication incur only negligible overhead compared with MHESA.

Table 9. The actual average time to perform main operations in V-MHESA ($N = 100$, Non-IID).

Phase	Actor	Operations	AVG Time (ms) of V-MHESA	AVG Time (ms) of MHESA
Setup	FS	Creating setup parameters including A and S	198.236	196.412
	U_i	Generating $\langle pk_i, c_i \rangle$ pair and K_i^+	529.098	496.73
	FS	Generating CS_i and CID_i for each U_i	0.1855	-
	U_R	Generating $CR = \{CR_i\}$ and SR for all U_i	8.539	-
	U_i	Decrypting each of CS_i , CID_i and CR_i	0.272	-
Learning	FS	Generating PK_T and C_T	626.5	624.1
	U_i	Generation AC_i	0.00092	-
	FS	Verifying AC_i for all U_i	0.043	-
	U_i	Generating $LU_i = \langle D_i, E_i, T_i \rangle$	274.785	259.759
	FS	Generating $GU = \langle GW, GT \rangle$	427.154	376.548
	U_i	Verification of GU	0.1887	-

During the *Setup* phase, the designated node U_R generates the initial random nonce hR^0 and encrypts it under the public keys of all other nodes. While the encryption cost scales linearly with the number of participating nodes, this operation is performed only once at initialization. Even in the 100-node setting, the total encryption time was approximately 8.54 ms, which is negligible. In the *Learning* phase, each node generates a masked local parameter D_i along with its verification token T_i . The additional cost of this operation is

minimal, with the total execution time remaining within 15 ms compared with the original local update generation in MHESA. Similarly, FS aggregates both D_i and T_i , but the total aggregation time differs by less than 50 ms from that in MHESA.

Overall, although V-MHESA introduces additional mechanisms for verifiability, server and node authentication, and dropout resilience, the associated overhead is extremely small. Therefore, these enhancements do not meaningfully impact the overall efficiency of the federated learning process.

Finally, we analyze the overall execution time per *Learning* round. During the *Learning* phase, the most time-consuming operations are (1) generating and distributing the group key and group commitment for all participating nodes, (2) collecting all local model updates from the nodes, (3) generating and broadcasting the global update from the aggregated local updates, and (4) verifying the validity of the global update and updating each node's local model with the new global parameters.

Figure 5 illustrates the execution times of these main operations according to the number of participating nodes. In the figure, “Round execution time by FS ” refers to the total execution time for steps (1)–(3), while “Total round execution time” represents the overall time required for steps (1)–(4), including the local model update at each node.

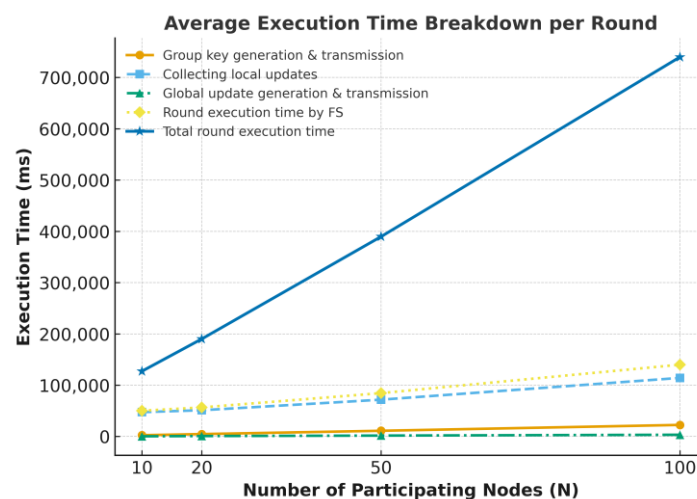


Figure 5. Average Execution time breakdown per round according to the number of participating nodes (N).

In step (1), the group key generation and transmission required 2466.25 ms when $N = 10$, 4644.08 ms when $N = 20$, 11,079.8 ms when $N = 50$, and 22,501.9 ms when $N = 100$. In step (2), the collection of local updates took 47,422.3 ms, 51,128.2 ms, 71,713.6 ms, and 114,191 ms for $N = 10, 20, 50$, and 100 , respectively. In step (3), the generation and transmission of the global update required 310.46 ms, 805.69 ms, 1691.64 ms, and 3303.94 ms, respectively. The average round execution time (steps (1)–(3)) therefore increased proportionally with the number of nodes, from 50,199.68 ms at $N = 10$ to 84,487.28 ms at $N = 50$ and 140,001.4 ms at $N = 100$. This proportional growth is natural since both computational and communication delays increase with the number of participating nodes.

When step (4) is included, the total execution time grows sharply with the number of nodes, mainly because the local model update at each node is the most time-consuming operation. In the experimental setup, all nodes were implemented as independent threads within a single client machine; thus, local model updates were executed sequentially rather than in parallel. As a result, the total execution time increased significantly—from 127,236 ms ($N = 10$) to 190,134 ms ($N = 20$), 389,664 ms ($N = 50$), and 739,829 ms ($N = 100$). However, in a real-world deployment, since nodes operate independently and perform

local model updates in parallel, the overall time would not increase proportionally with N , but rather approximate the round execution time plus the maximum local update time among all nodes.

Figure 6 presents the results of analyzing the average execution time under various dropout conditions. Dropout occurs during the local update collection phase. When certain nodes drop out, the system waits for the maximum timeout (150 s in the experiment, corresponding to the maximum time required to complete step (2) for $N = 100$), and then reinitiates the *Learning* phase from the beginning with the remaining nodes to complete the round. Figure 6a presents the round execution time for dropout ratios (d) of 10%, 30%, and 50%, whereas Figure 6b presents the corresponding total round execution time. As shown in Figure 6a, once dropout occurs, the overall execution time inevitably takes longer than in the non-dropout case, since it must repeat the aggregation process with the remaining nodes after the timeout. However, as the dropout ratio increases, the number of remaining nodes decreases, leading to a slight reduction in the re-aggregation time. For example, when $N = 100$, the round execution time was 304,909 ms for $d = 10\%$, 275,023.8 ms for $d = 30\%$, and 248,489.6 ms for $d = 50\%$.

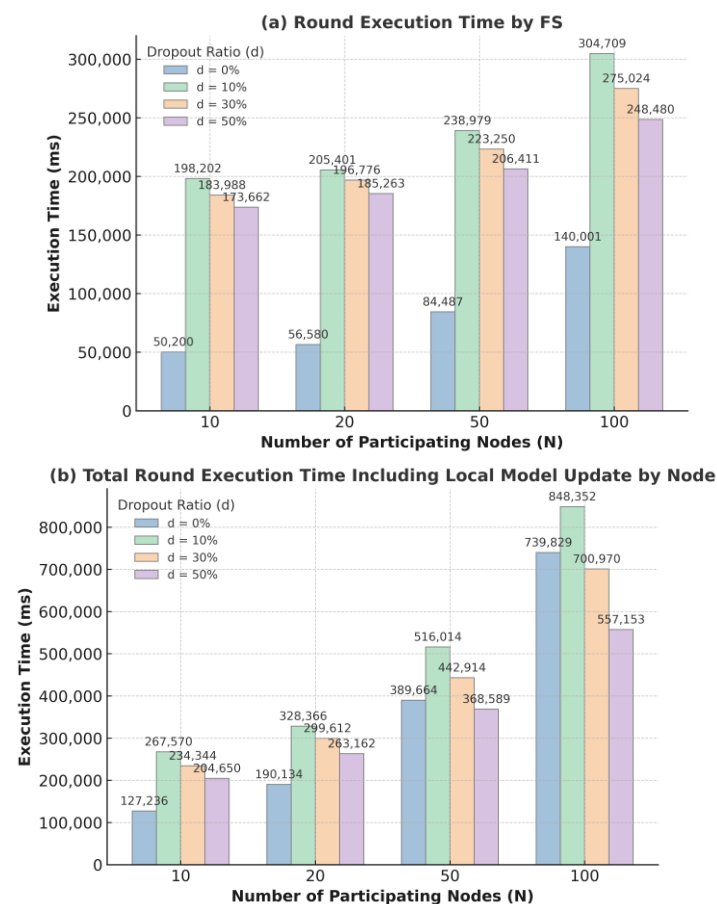


Figure 6. Average execution time per round under various dropout ratios (d) for different number of participating nodes (N).

Figure 6b includes the local model update by each node and shows that, in our experiments, the total round execution time decreases as the dropout ratio increases. This is because fewer active nodes lead to fewer local model updates to perform, thereby reducing the total execution time. For instance, when $N = 100$, the total execution time was 848,352.4 ms for $d = 10\%$, 700,970.4 ms for $d = 30\%$, and 557,153.4 ms for $d = 50\%$. However, in practical scenarios, since the nodes independently perform their local model updates in parallel, even under various dropout conditions, the total execution time is expected to

increase only marginally appropriately by the maximum local model update time added to the round execution time.

We conducted an experimental evaluation of the V-MHESA-based federated learning model with respect to learning accuracy, aggregation correctness, the overhead introduced by the additional verification mechanisms of V-MHESA, and the overall round execution time. The results demonstrate that incorporating verifiable local updates does not compromise learning accuracy—the proposed model achieves the same accuracy as the original MHESA scheme. The total increase in communication volume caused by the verification functionality remains within approximately 10 KB even with 100 participating nodes, while the additional computation cost is limited to only a few tens of milliseconds. Compared with MHESA, both the communication and computational overheads introduced by V-MHESA are therefore negligible. Although the overall round execution time increases with the number of participating nodes, the average round time of about 140 s for $N = 100$ indicates that the proposed protocol can be executed within a reasonable duration. Although real network communication delays were not included in the experimental setup, the protocol's intrinsic computational overhead remains modest, suggesting that, in practical deployments, total performance would be dominated primarily by network communication rather than by the internal operations of the protocol itself.

5. Discussion

This study presented V-MHESA, a verifiable secure aggregation protocol for federated learning that extends our previous MHESA scheme by incorporating lightweight verification and authentication mechanisms. Unlike conventional secure aggregation methods that solely protect local parameters, V-MHESA additionally guarantees the correctness and integrity of the aggregated global model.

A key innovation of V-MHESA lies in its multi-layered security design. In each *Learning* round, *FS* distributes a round nonce, and every node responds with a hash-based authentication code derived from its (secure) identifier (known only to *FS*) and the nonce. This procedure ensures that only legitimate, pre-registered nodes can participate in the aggregation, thereby providing robust node authentication.

During local update generation, each node masks its parameter w_i using (1) its own random mask M_i , (2) *FS*'s secret S , and (3) a shared random secret R known only to the participating nodes. This ensures that only *FS* can remove the aggregate masking term to obtain the blinded sum $GW = R \cdot w$, while only the nodes that share R can subsequently recover the actual global model w . Consequently, adversaries outside the legitimate *FS*-node group are prevented from joining the learning process, generating aggregation results, or generating forged updates.

The shared secret R is updated independently by the nodes in each learning round via hash chaining which eliminates any need for repeated communication or collaboration between nodes. This mechanism not only minimizes communication overhead but also strengthens the confidentiality of local updates by ensuring that a fresh random secret is used in every round.

To achieve verifiability, each node additionally generates a verification token for a small, randomly selected subset of parameters. This token-based approach requires neither a trusted third party nor complex cryptographic primitives such as homomorphic hash function or secret sharing. Instead, it relies on simple masking operations and lightweight arithmetic aggregation, resulting in negligible computational and communication cost.

From a security standpoint, V-MHESA ensures that all local parameters remain confidential under the honest-but-curious threat model. *FS* can obtain only the blinded aggregation $R \cdot w$, while the true global parameter w is recoverable solely by the nodes that

share R . Moreover, forging a valid global update pair $\langle GW, GT \rangle$ without knowledge of R is computationally infeasible, with success probability bounded by 2^{-2^m} , where m denotes the bit length of the pseudo-random output.

Experimental evaluations on the MNIST dataset confirmed that this verification mechanism does not affect model accuracy. The V-MHESA model achieved nearly identical accuracy to MHESA under both IID and non-IID settings. The additional communication incurred by transmitting verification tokens remained minimal compared with the overall size of masked local parameters and encrypted mask values.

The main limitation of the proposed model lies in the relatively large size of encrypted mask data. This is an inherent trade-off of using CKKS-HE, which provides 128-bit security but increases ciphertext size. Therefore, V-MHESA is particularly suitable for high-security federated learning environments—such as healthcare, finance, or other domains where data sensitivity is critical—and where static node participation, high computational capability, and sufficient network bandwidth are available. In contrast, in lightweight mobile environments with limited bandwidth, the communication overhead caused by large ciphertexts may be non-negligible. However, by appropriately tuning the security level—such as adjusting the polynomial degree used in CKKS-HE—it is possible to reduce the ciphertext size and thus the transmission cost. Future work will explore such adaptive configurations to make V-MHESA more broadly applicable across diverse federated learning environments with different security and resource requirements.

6. Conclusions

V-MHESA achieves a compelling balance between security, verifiability, and efficiency in federated learning. It introduces only marginal computational and communication overhead compared with MHESA, yet it significantly enhances the trustworthiness of the aggregation process by enabling every node to independently verify the correctness and integrity of the global model—without revealing any sensitive information.

By combining masking and homomorphic encryption with a lightweight verification mechanism, V-MHESA effectively overcomes one of the most critical limitations of existing secure aggregation schemes. Its simplicity, robustness, and negligible overhead make it particularly suitable for real-world, privacy-sensitive applications such as healthcare, finance, and IoT-based collaborative learning.

Future work will extend V-MHESA to more heterogeneous and large-scale federated environments, including asynchronous learning, dynamic node participation, and stronger adversarial models beyond the honest-but-curious assumption.

Author Contributions: Conceptualization, S.P.; methodology, S.P.; software, S.P.; validation, S.P. and J.C.; formal analysis, S.P.; data curation, J.C.; writing—original draft preparation, S.P.; writing—review and editing, S.P. and J.C.; supervision, S.P.; project administration, S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-2021R1F1A1063172).

Data Availability Statement: The original data presented in the study are openly available at <http://yann.lecun.com/exdb/mnist> (accessed on 3 October 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

- McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 9–11 May 2017; Volume 54.
- Zhu, L.; Liu, Z.; Han, S. Deep Leakage from Gradients. *arXiv* **2019**, arXiv:1906.08935. [\[CrossRef\]](#)
- Wang, Z.; Song, M.; Zhang, Z.; Song, Y.; Wang, Q.; Qi, H. Beyond Inferring Class Representatives: User-Level Privacy Leakage from Federated Learning. In Proceedings of the IEEE INFOCOM, Paris, France, 29 April–2 May 2019; pp. 2512–2520. [\[CrossRef\]](#)
- Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. Inverting Gradients—How Easy Is It to Break Privacy in Federated Learning? In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Online, 6–12 December 2020.
- Yao, A.C. Protocols for Secure Computations. In Proceedings of the 23rd IEEE Annual Symposium on Foundations of Computer Science (SFCS 1982), Chicago, IL, USA, 3–5 November 1982; pp. 160–164. [\[CrossRef\]](#)
- Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613. [\[CrossRef\]](#)
- Geyer, R.C.; Klein, T.; Nabi, M. Differentially Private Federated Learning: A Client Level Perspective. In Proceedings of the NIPS 2017 Workshop: Machine Learning on the Phone and Other Consumer Devices, Proceedings of the Conference on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 8 December 2017. [\[CrossRef\]](#)
- Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.S.; Poor, H.V. Federated Learning with Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [\[CrossRef\]](#)
- Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Federated Learning on User-Held Data. *arXiv* **2016**, arXiv:1611.04482. [\[CrossRef\]](#)
- Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the ACM SIGSAC Conferences on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191. [\[CrossRef\]](#)
- Ács, G.; Castelluccia, C. I Have a DREAM! (DiffeRentially privatE smArt Metering). In Proceedings of the Information Hiding (IH 2011); Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany, 2011; Volume 6958, pp. 118–132. [\[CrossRef\]](#)
- Goryczka, S.; Xiong, L. A Comprehensive Comparison of Multiparty Secure Additions with Differential Privacy. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 463–477. [\[CrossRef\]](#)
- Elahi, T.; Danezis, G.; Goldberg, I. PrivEx: Private Collection of Traffic Statistics for Anonymous Communication Networks. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 1068–1079. [\[CrossRef\]](#)
- Jansen, R.; Johnson, A. Safely Measuring Tor. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1553–1567. [\[CrossRef\]](#)
- So, J.; Güler, B.; Avestimehr, A.S. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. *arXiv* **2020**, arXiv:2002.04156. [\[CrossRef\]](#)
- Kim, J.; Park, G.; Kim, M.; Park, S. Cluster-Based Secure Aggregation for Federated Learning. *Electronics* **2023**, *12*, 870. [\[CrossRef\]](#)
- Leontiadis, I.; Elkhayaoui, K.; Molva, R. Private and Dynamic Timeseries Data Aggregation with Trust Relaxation. In *Proceedings of the International Conferences on Cryptology and Network Security (CANS 2014)*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8813, pp. 305–320. [\[CrossRef\]](#)
- Rastogi, V.; Nath, S. Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 10), Indianapolis, IN, USA, 6–10 June 2010; pp. 735–746. [\[CrossRef\]](#)
- Halevi, S.; Lindell, Y.; Pinkas, B. Secure Computation on the Web: Computing without Simultaneous Interaction. In *Proceedings of the Advances in Cryptology—CRYPTO 2011*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6841, pp. 132–150. [\[CrossRef\]](#)
- Leontiadis, I.; Elkhayaoui, K.; Önen, M.; Molva, R. PUDA—Privacy and Unforgeability for Data Aggregation. In *Proceedings of the Cryptology and Network Security (CANS 2015)*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 9476, pp. 3–18. [\[CrossRef\]](#)
- Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345. [\[CrossRef\]](#)
- Fang, H.; Qian, Q. Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning. *Future Internet* **2021**, *13*, 94. [\[CrossRef\]](#)
- Park, J.; Lim, H. Privacy-Preserving Federated Learning Using Homomorphic Encryption. *Appl. Sci.* **2022**, *12*, 734. [\[CrossRef\]](#)
- Liu, W.; Zhou, T.; Chen, L.; Yang, H.; Han, J.; Yang, X. Round Efficient Privacy-Preserving Federated Learning based on MKFHE. *Comput. Stand. Interfaces* **2024**, *87*, 103773. [\[CrossRef\]](#)
- Jin, W.; Yao, Y.; Han, S.; Gu, J.; Joe-Wong, C.; Ravi, S.; Avestimehr, S.; He, C. FedML-HE: An Efficient Homomorphic-Encryption-based Privacy-Preserving Federated Learning System. *arXiv* **2023**, arXiv:2303.10837.

26. Wibawa, F.; Catak, F.O.; Kuzlu, M.; Sarp, S.; Cali, U. Homomorphic Encryption and Federated Learning based Privacy-Preserving CNN Training: COVID-19 Detection Use-Case. In Proceedings of the 2022 European Interdisciplinary Cyber-Security Conference (EICC '22), Barcelona, Spain, 15–16 June 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 85–90. [\[CrossRef\]](#)
27. Hijazi, N.M.; Aloqaily, M.; Guizani, M.; Ouni, B.; Karray, F. Secure Federated Learning with Fully Homomorphic Encryption for IoT Communications. *IEEE Internet Things J.* **2024**, *11*, 4289–4300. [\[CrossRef\]](#)
28. Sanon, S.P.; Reddy, R.; Lipps, C.; Schotten, H.D. Secure Federated Learning: An Evaluation of Homomorphic Encrypted Network Traffic Prediction. In Proceedings of the IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023; pp. 1–6. [\[CrossRef\]](#)
29. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic Encryption for Arithmetic of Approximate Numbers. In Proceedings of the Advances in Cryptology-ASIACRYPT 2017; Lecture Notes in Computer Science; Springer: Cham, Switzerland; 2017; Volume 10624. [\[CrossRef\]](#)
30. Park, S.; Lee, J.; Harada, K.; Chi, J. Masking and Homomorphic Encryption-Combined Secure Aggregation for Privacy-Preserving Federated Learning. *Electronics* **2025**, *14*, 177. [\[CrossRef\]](#)
31. LeCun, Y.; Cortes, C.; Burges, C.J. MNIST Handwritten Digit Database. 2010. Available online: <http://yann.lecun.com/exdb/mnist> (accessed on 3 October 2025).
32. Xu, G.; Li, H.; Liu, S.; Yang, K.; Lin, X. VerifyNet: Secure and Verifiable Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 911–926. [\[CrossRef\]](#)
33. Guo, X.; Liu, Z.; Li, J.; Gao, J.; Hou, B.; Dong, C.; Baker, T. VeriFL: Communication-Efficient and Fast Verifiable Aggregation for Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2020**, *16*, 1736–1751. [\[CrossRef\]](#)
34. Buyukates, B.; So, J.; MahdaviFar, H.; Avestimehr, S. LightVeriFL: A Lightweight and Verifiable Secure Aggregation for Federated Learning. *arXiv* **2022**, arXiv:2207.08160. [\[CrossRef\]](#)
35. Behnia, R.; Riasi, A.; Ebrahimi, R.; Chow, S.S.M.; Padmanabhan, B.; Hoang, T. Efficient Secure Aggregation for Privacy-Preserving Federated Machine Learning (e-SeaFL). In Proceedings of the 2024 Annual Computer Security Applications Conference (ACSAC), Honolulu, HI, USA, 9–13 December 2024; pp. 778–793. [\[CrossRef\]](#)
36. Peng, K.; Li, J.; Wang, X.; Zhang, X.; Luo, B.; Yu, J. Communication-Efficient and Privacy-Preserving Verifiable Aggregation for Federated Learning. *Entropy* **2023**, *25*, 1125. [\[CrossRef\]](#)
37. Zhou, S.; Wang, L.; Chen, L.; Wang, Y.; Yuan, K. Group Verifiable Secure Aggregate Federated Learning based on Secret Sharing. *Sci. Rep.* **2025**, *15*, 9712. [\[CrossRef\]](#)
38. Xu, B.; Wang, S.; Tian, Y. Efficient Verifiable Secure Aggregation Protocols for Federated Learning. *J. Inf. Secur. Appl.* **2025**, *93*, 104161. [\[CrossRef\]](#)
39. Li, G.; Zhang, Z.; Du, R. LVSA: Lightweight and Verifiable Secure Aggregation for Federated Learning. *Neurocomputing* **2025**, *648*, 130712. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.