Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.065887

ARTICLE

VPAFL: Verifiable Privacy-Preserving Aggregation for Federated Learning Based on Single Server

Peizheng Lai[1], Minqing Zhang[1,2,*], Yixin Tang[1], Ya Yue[1] and Fuqiang Di[1,2]

[1]College of Cryptography Engineering, Engineering University of PAP, Xi'an, 710086, China
[2]Key Laboratory of PAP for Cryptology and Information Security, Xi'an, 710086, China
*Corresponding Author: Minqing Zhang. Email: api_zmq@126.com

Received: 24 March 2025; Accepted: 08 May 2025; Published: 03 July 2025

ABSTRACT: Federated Learning (FL) has emerged as a promising distributed machine learning paradigm that enables multi-party collaborative training while eliminating the need for raw data sharing. However, its reliance on a server introduces critical security vulnerabilities: malicious servers can infer private information from received local model updates or deliberately manipulate aggregation results. Consequently, achieving verifiable aggregation without compromising client privacy remains a critical challenge. To address these problem, we propose a reversible data hiding in encrypted domains (RDHED) scheme, which designs joint secret message embedding and extraction mechanism. This approach enables clients to embed secret messages into ciphertext redundancy spaces generated during model encryption. During the server aggregation process, the embedded messages from all clients fuse within the ciphertext space to form a joint embedding message. Subsequently, clients can decrypt the aggregated results and extract this joint embedding message for verification purposes. Building upon this foundation, we integrate the proposed RDHED scheme with linear homomorphic hash and digital signatures to design a verifiable privacy-preserving aggregation protocol for single-server architectures (VPAFL). Theoretical proofs and experimental analyses show that VPAFL can effectively protect user privacy, achieve lightweight computational and communication overhead of users for verification, and present significant advantages with increasing model dimension.

KEYWORDS: Verifiable federated learning; privacy-preserving; homomorphic encryption; reversible data hiding in encrypted domain; secret sharing

1 Introduction

1.1 Motivation

With the widespread adoption of cloud computing technologies [1], the migration of large amounts of user data to the cloud has become an irreversible trend. To address the risks of privacy breaches, encrypting data for storage has become essential. However, this measure introduces new challenges, particularly in performing operations such as metadata embedding and identity authentication while ensuring data confidentiality.

To address these challenges, the technique of reversible data hiding in the encrypted domain (RDHED) [2–4] has emerged as a promising solution. This technology enables the reversible embedding of data, such as identity markers and integrity labels, within encrypted data [5,6]. Upon decryption, both the original data and the embedded data can be completely recovered, providing an innovative approach to managing encrypted data. Despite significant advancements in traditional RDHED methods, most

existing schemes rely on stream cipher encryption mechanisms, which do not support ciphertext operations [2,4–6]. This limitation makes them unsuitable for emerging privacy-preserving computing scenarios, such as federated learning (FL) [7].

In recent years, homomorphic encryption (HE)-based RDHED schemes have gained attention as a potential solution [8–10]. However, these approaches face two major technical challenges. First, their applicability is often limited because most existing research focuses on image data processing, while FL predominantly involves model parameters. Second, compatibility issues arise: existing methods do not adequately address the destructive effects of homomorphic processing on embedded data [11]. For instance, when encrypted data undergo homomorphic processing, such as ciphertext aggregation, embedded data may suffer irreversible distortion, resulting in extraction failures [12–14]. Furthermore, FL usually involves multiple participants, yet enabling each user to independently perform data embedding and extraction independently in decentralized scenarios remains a great challenge.

In a typical FL architecture, users upload their local models to the server, which aggregates the received local models to generate the global model and distributes it back to users. However, this collaborative training mechanism raises concerns about privacy leakage. In particular, a semi-honest server can infer user privacy by analyzing local or global gradients [15–17], while a malicious server could manipulate the aggregation results, thereby compromising the usability of the global model [18]. Existing privacy protection solutions, such as HE [19,20], differential privacy [21,22], and secure multiparty computation [23], can mitigate some privacy risks. However, they fail to address a critical issue: verifying the correctness of model aggregation.

To address these challenges, several verifiable federated learning (VFL) schemes have been successively proposed [18,24–27]. These solutions primarily utilize linear homomorphic hash (LHH) [18,24–26] or dual-aggregation techniques [27] to achieve verifiability. However, they exhibit limitations: the former incurs computational overhead that scales with the model dimension, while the latter suffers from the inflation of communication costs and requires auxiliary protocols for full verifiability [28,29].

1.2 Our Contributions

This study addresses existing challenges by focusing on two core issues: (1) designing a RDHED scheme compatible with homomorphic processing, and (2) integrating this RDHED scheme with cryptographic tools to develop an efficient verifiable privacy-preserving aggregation protocol under a single-server architecture.

To achieve these goals, we propose a joint embedding-extraction mechanism (JEEM). Using the additive homomorphic property of the Paillier encryption algorithm [19], JEEM enables multiple users to collaboratively embed and extract secret messages without altering the original plaintext. Building on JEEM, we further integrate LHH [30] and digital signatures to design a verifiable privacy-preserving aggregation protocol based on a single server (VPAFL).

The contributions of this study can be summarized as follows:

1) RDHED Scheme for Joint Secret Message Embedding and Extraction: This study resolves the compatibility limitations of existing RDHED schemes with homomorphic processing. Each user independently exploits ciphertext redundancy during encryption to embed secret messages. After the server aggregates the ciphertexts, homomorphic properties enable the fusion of all user-embedded messages within the ciphertext space, yielding a joint embedded message. Users can then extract this joint embedded message after decrypting the aggregation result.

2) Efficient Verifiable Aggregation Protocol: Compared to existing LHH-based VFL schemes, VPAFL enhances efficiency by integrating the proposed RDHED scheme. Specifically, VPAFL utilizes the secret message embedded by the users in each communication round as input for the hash value computation. This

design decouples hash generation computational overhead from the model dimension, thereby significantly reducing the computational overhead of users for verification. Furthermore, VPAFL maintains minimal communication overhead of users for verification (below 0.2 KB) and achieves verification of aggregation results within only two interaction rounds under a single-server framework, significantly enhancing practical deployability.

1.3 Organization

The rest of this article is organized as follows: Section 2 reviews related works, while Section 3 introduces preliminary concepts. Section 4 provides an overview of the system and the threat Section 5 presents the proposed RDHED scheme and details the VPAFL protocol. Sections 6 and 7 offer theoretical analyses and experimental results, respectively. Finally, Section 8 concludes this study.

2 Related Works

In this section, we provide a brief review of the work related to RDHED schemes in the homomorphic encrypted domain and VFL.

2.1 RDHED Schemes in the Homomorphic Encrypted Domain

With the widespread application of HE in privacy computing, HE-based RDHED methods have emerged as a research hotspot [12]. These schemes are categorized into two types based on their impact on plaintext: plaintext modification schemes (Type I) and lossless data hiding schemes (Type II). Specifically, Type I methods involve embedding operations that result in changes to plaintext [9,10,13,14]. For example, a typical method modifies the ciphertext value c to produce a decrypted plaintext of the form $2m + b$, where m is the original plaintext, and $b \in \{0, 1\}$ denotes the embedded 1 bit message. During data extraction, the embedded bit b is extracted by computing $(2m + b) \mod 2$, while m is recovered by integer division $\lfloor (2m + b)/2 \rfloor$. However, a critical limitation of Type I schemes is that they are not well suited for directly processing ciphertexts containing embedded messages. As a result, employing Type I methods may limit the processing of ciphertexts.

In contrast, Type II schemes aim to achieve lossless data hiding in ciphertexts (LDH-CT) without altering plaintexts or increasing ciphertext size [8,11,12]. For example, Zheng et al. [12] proposed a LDH-CT scheme based on numerical interval mapping: the data hider modifies the ciphertext to fall into specific subintervals corresponding to embedded bits. Using the homomorphic and probabilistic properties of cryptosystems to ensure invariance of plaintext. Wu et al. [11] proposed a RDHED scheme using random number substitution (RS). In this scheme, binary secret messages are first converted into decimal numbers that then replace the random numbers used during the encryption process to embed the data. However, this method constrains the bit length of embedded messages, as exceeding predefined limits disrupts both encryption and decryption processes.

Despite advances in existing research, two critical challenges hinder the application of RDHED schemes to FL: First, existing RDHED schemes primarily target image data carriers, whereas FL predominantly processes model parameters. Second, existing schemes do not adequately address compatibility with homomorphic processing [12–14]. In particular, when encrypted data containing embedded information undergoes homomorphic computations, such as ciphertext aggregation, the embedded data may suffer irreversible distortion, thereby leading to extraction failure.

2.2 Verifiable Federated Learning

In FL, the trustworthiness of the servers cannot be absolutely guaranteed as malicious servers can return incorrect aggregation results [18,24,25]. Models derived from such compromised servers inevitably underperform in prediction or classification tasks, necessitating VFL schemes to mitigate these risks.

Existing VFL research follows mainly two technical paths: LHH-based schemes [18,24–26] and dual-aggregation frameworks [27]. Xu et al. [18] proposed the first VFL framework, integrating a double-masking protocol [23] for privacy protection with LHH and pseudorandom techniques to achieve verifiable aggregation. However, this scheme suffers from two critical drawbacks: First, communication overheads scale with the model dimension. Second, computationally intensive bilinear pairing operations. To optimize communication efficiency, Guo et al. [24] proposed VeriFL, which decouples communication overhead for verification from model dimensions via LHH combined with equivocal commitments. Recent advances include VPFLI [25] and PriVeriFL [26], where VPFLI [25] designs a novel aggregation protocol to minimize performance degradation caused by heterogeneous client data quality, while PriVeriFL [26] employs a blockwise encryption strategy to alleviate computational bottlenecks of HE, reducing resource demands without compromising security.

However, LHH-based VFL schemes remain plagued by high computational overhead, as the complexity of hash value calculation increases with model dimensions [26]. To enable lightweight verification, Hahn et al. proposed VERSA [27], a dual-aggregation verification framework that eliminates the need for trusted setups and uses a lightweight pseudorandom generator (PRG) to enable efficient verification of the aggregation result. However, recent studies have identified vulnerabilities that compromise its verifiability [28,29].

In summary, neither LHH-based VFL nor dual-aggregate verification-based schemes achieve high efficiency. As shown in Table 2 (detailed in Section 7.4.1), the LHH-based approach incurs significant computational overhead during verification as the model dimension grows, requiring approximately 12,490 s to compute the LHH values for models with dimensions reaching 10,000,000. In contrast, dual-aggregate verification-based schemes face substantial communication overhead: Users must submit both local model updates and validation codes derived from these parameters, which doubles their communication expenditure. Furthermore, as the model dimensionality increases, the communication overhead introduced by validation becomes impractical for real-world applications. Crucially, given the resource constraints of end-user devices, the designed VFL framework should maintain lightweight communication and computational overheads for verification to facilitate practical deployment.

3 Preliminaries

In this section, we provide the foundational concepts necessary to understand our VPAFL scheme.

3.1 Federated Learning

Deep learning has attracted significant attention for its remarkable achievements in various fields, though high-performance deep neural networks (DNNs) typically rely on extensive datasets. However, the data used to train DNNs often contain sensitive information. For example, location-based services [31] could expose personal whereabouts, while goods purchase records can be exploited for targeted advertising. More critically, the leakage of health information or facial data poses serious privacy risks. To address these challenges, FL has emerged as a promising solution [7]. Since its inception, FL has been widely adopted in various applications such as the Internet of Things (IoT), smart healthcare [32], and smart cities [33]. FL is a distributed machine learning paradigm that collaboratively trains a global model by coordinating multiple participants without compromising data privacy. In this architecture, the server does not directly

access user data; instead, it iteratively aggregates parameters to optimize the global model. Let $\mathcal{U}$ denote the set of participating users, where $|\mathcal{U}| = n$ represents the total number of users, and each user $u_i \in \mathcal{U}$ has a private dataset $\mathbf{D}_i$. The FL training process proceeds iteratively through the following stages per communication round:

a) Global Model Distribution: The server broadcasts the current global model $\mathbb{M}^k$ to all users, where k denotes the current communication round.

b) Local Model Training: Each user u_i initializes their local model with $\mathbb{M}^k$ and trains on $\mathbf{D}_i$ using the stochastic gradient descent (SGD) algorithm [34].

c) Model Aggregation: The server aggregates the received local models via the Federated Averaging (FedAvg) algorithm [7] to generate the updated global model $\mathbb{M}^{k+1}$.

Formally, during the k-th communication round, each user u_i updates its local model parameters using the SGD algorithm [34]:

$$\mathbf{w}_i^k = \mathbf{w}_i^{k-1} - l_r \nabla \mathcal{L}(\mathbf{w}_i^{k-1}; \mathbf{D}_i), \tag{1}$$

where l_r denotes the local learning rate, $\mathbf{w}_i^k$ denotes the parameters of the local model for user u_i during the k-th communication round, and $\mathcal{L}(\cdot)$ denotes the loss function. Subsequently, the server aggregates the received local models using the FedAvg algorithm [7] as follows:

$$\mathbf{w}^k = \sum_{u_i \in \mathcal{U}} \frac{|\mathbf{D}_i|}{|\sum_{u_i \in \mathcal{U}} \mathbf{D}_i|} \mathbf{w}_i^k. \tag{2}$$

3.2 Linear Homomorphic Hash

LHH [30] is a one-way and collision-resistant homomorphic hash function that can calculate the hash of a composite data block based on the hash of a single data block. The LHH scheme is formally defined by three algorithms LHH = (**LHH.Gen**, **LHH.Hash**, **LHH.Eval**):

a) **LHH.Gen**(κ, v): Given the security parameters κ and a d-dimensional vector $v = \{v_i, \ldots, v_d\}$, this algorithm outputs public parameters $pp = \{\mathbb{G}, q, \{g_1, \ldots, g_d\}\}$, where $\mathbb{G}$ is a cyclic group of prime order q, g is a generator of $\mathbb{G}$, and $g_1, \ldots, g_d$ are distinct elements in $\mathbb{G}$.

b) **LHH.Hash**(pp, v): For a d-dimensional vector v, this algorithm computes its LHH value of v: $h_v \leftarrow \prod_{i=1}^{d} g_i^{v_i} \in \mathbb{G}$.

c) **LHH.Eval**$(pp, h_1, \ldots, h_l, \alpha_1, \ldots, \alpha_l)$: Given l hash values $h_1, \ldots, h_l$ and l coefficients $\alpha_1, \ldots, \alpha_l \in \mathbb{Z}_q$, this algorithm outputs the linear combination of the l hash values: $h \leftarrow \prod_{i=1}^{l} h_l^{\alpha_l}$.

4 Overview of the System and the Threat Model

4.1 System Model

As illustrated in Fig. 1, the system model of the VPAFL protocol comprises three entities, consistent with previous works [18,24]: The trusted authority (TA), the server, and the users.

- **TA:** The TA initializes the system by generating cryptographic parameters (public/private keys) and distributing the initial global model to all participants. It is considered trustworthy and remains offline after initialization.
- **Server:** The server aggregates encrypted local models received from users, updates the global model, and broadcasts the aggregated result to the user for verification.

- **Users:** In each communication round (except for the first), users download the latest global model from the server as their local mode. During the first communication round, the TA initializes the global model. Each user trains the local model on their private dataset, encrypts the local model parameters with their private key, and uploads the ciphertext to the server. Upon receiving the aggregated result, users verify its correctness. Training continues only if verification succeeds; otherwise, training is aborted.

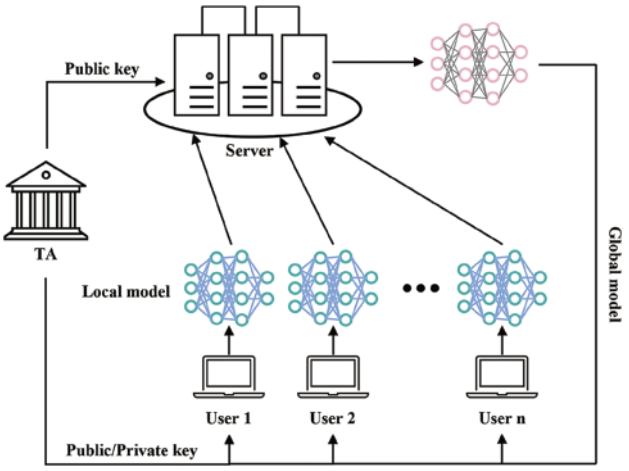

Figure 1: System model of VPAFL

4.2 Threat Model

Our threat model is defined as follows:

- **Semi-honest users:** Users follow the FL protocol faithfully, but may attempt to infer sensitive information from the data of honest users. Additionally, a subset of users may collude with the server to manipulate the aggregation results.
- **Server:** The server may attempt to infer user privacy or forge verifiable aggregation results. In particular, the server can collude with up to $t-1$ users, where t denotes the threshold parameter in Shamir's Secret Sharing (**SS**) protocol [35].
- **Out-of-Scope attacks:** In FL systems, not all participants are trustworthy, as malicious users can launch poisoning attacks [36] aimed at manipulating local models to compromise the performance of the global model. However, this paper does not consider poisoning attacks, as our primary objective focuses on ensuring the correctness of the aggregation results while maintaining user privacy protection.

5 Proposed Scheme

In this section, we first propose a RDHED scheme designed for compatibility with homomorphic processing. In particular, we design a JEEM by utilizing the additive homomorphic properties of the Paillier cryptosystem [19]. Subsequently, we further construct the VPAFL protocol by integrating the proposed RDHED scheme with the LHH [30] and digital signature algorithms.

5.1 Joint Embedding-Extraction Mechanism

Extending the drop-tolerant secure aggregation algorithm (**DTSA**) proposed by Zhao et al. [20], we propose a novel secure aggregation-data hiding (**SADH**) hybrid algorithm that is compatible with homomorphic processing. The core innovation of **SADH** lies in its JEEM, which enables users to collaboratively embed secret messages in ciphertext and extract them after aggregation. For simplicity, we assume that no user dropout occurs during training. The scheme operates as follows:

- **SADH.Gen**$(\kappa, t, \mathcal{U})$: This algorithm initializes cryptographic parameters. The inputs include the security parameter κ, the threshold t for the **SS** protocol [35] and the user set $\mathcal{U}$ with $|\mathcal{U}| = n$. First, generate the public key $PP = \{n, g, h, N\}$ shared among all users, along with the private key $\{SK, SK_i\}$ assigned to each user u_i, and the public parameter N^2 for the server. In particular, two large prime numbers p, q are selected such that $|p| = |q| = \kappa$. Then, compute $N = p \times q$ and $\lambda = lcm(p-1, q-1)$, where $lcm(a, b)$ denotes the least common multiple of a and b. Next, choose a random integer $g \in \mathbb{Z}^*_{N^2}$ that satisfies $gcd(L(g^\lambda \mod N^2), N) = 1$, where $L(x) = x - 1/N$. Subsequently, compute $e = g^\sigma \mod N^2$, where σ is a large prime satisfying $|\sigma| \leq \kappa$. Finally, construct the polynomial $f(x)$ as follows:

$$f(x) = a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_{t-1}x^{t-1} \mod p, \tag{3}$$

  where the private key is denoted as $SK = \{\lambda, \sigma\}$ and $SK_i = s^{q \cdot f(i)} \mod N^2$, the coefficients $a_1, \ldots, a_{t-1} \in \mathbb{Z}_p$ and $s \in \mathbb{Z}_N$ are randomly chosen.
- **SADH.Ence**(SK_i, w_i, d_i): This encryption-embedding hybrid algorithm processes the plaintext w_i and messages d_i as input, encrypts $\mathbf{w}_i$ using SK_i and outputs the ciphertext $[[w_i]]$:

$$[[w_i]] = g^{w_i} \cdot e^{d_i} \cdot SK_i^{(n-1)! \prod_{j \in \mathcal{U}, j \neq i} \frac{j}{j-i}} \mod N^2. \tag{4}$$

  The core mechanism lies in embedding secret messages through redundant ciphertext values without changing the plaintext, as demonstrated in Section 6.1. The key parameters are defined as follows:
  - The embedded message $d_i = r_i \times d_{m_i}$ serves two purposes: (1) to carry a secret message and (2) to allow collaborative generation of joint embedded messages. Where $m_i = \{b_0, \ldots, b_l\}$ denotes a binary sequence of length l, and d_{m_i} denotes its decimal value.
  - The random number $r_i \in \mathbb{Z}_N$ is used to ensure the confidentiality of the message and $|r_i \times d_{m_i}| < \kappa/2$.
  - The user set $\mathcal{U}$ contains n users ($|\mathcal{U}| = n$).

  SADH.Agg$(\{[[w_i]]\}_{u_i \in \mathcal{U}})$: The server aggregates the received ciphertexts as follows:

$$\begin{aligned}
[[w]] &= \prod_{u_i \in \mathcal{U}} [[w_i]] \mod N^2. \\
&= \prod_{u_i \in \mathcal{U}} g^{w_i} \cdot e^{d_i} \cdot SK_i^{(n-1)! \prod_{j \in \mathcal{U}, j \neq i} \frac{j}{j-i}} \mod N^2 \\
&= e^{\sum_{u_i \in \mathcal{U}} d_i} \cdot \left(\prod_{u_i \in \mathcal{U}} g^{w_i} \cdot SK_i^{(n-1)! \prod_{j \in \mathcal{U}, j \neq i} \frac{j}{j-i}} \right) \mod N^2,
\end{aligned} \tag{5}$$

  where $\sum_{u_i \in \mathcal{U}_o} d_i$ denotes the joint embedded message d, formed by fusing the user-embedded secret messages within the ciphertext domain.

SADH.Dece$(SK, [[w]])$: After receiving the aggregated ciphertext $[[w]]$, the user decrypts it with the private key SK and extracts the joint embedded message d through the following steps:

$$\begin{aligned} w = \sum_{u_i \in \mathcal{U}_o} w_i &= \frac{L([[w]]^{\lambda} \mod N^2)}{L(g^{\lambda} \mod N^2)} \mod N \mod \sigma \\ &= \left(\sum_{u_i \in \mathcal{U}_o} w_i + \sigma \cdot \sum_{u_i \in \mathcal{U}_o} d_i \right) \mod \sigma \\ &= \sum_{u_i \in \mathcal{U}_o} w_i. \end{aligned} \tag{6}$$

Define $w' = \left(\sum_{u_i \in \mathcal{U}_o} w_i + \sigma \cdot \sum_{u_i \in \mathcal{U}_o} d_i \right) \mod N$. Because $\sum_{u_i \in \mathcal{U}_o} w_i$ and σ are known parameters, the user can derive $\sum_{u_i \in \mathcal{U}_o} d_i$, which corresponds to the joint embedded message d.

5.2 Verifiable Privacy-Preserving Aggregation for Federated Learning

In the proposed protocol, users first utilize **SADH** to encrypt their local models and embed secret messages, subsequently uploading both the hash and the signature to the server. Following this, the server aggregates the received ciphertexts and returns the aggregation results along with the collected hashes and signatures to the users. Finally, after receiving these, each user decrypts the aggregation result to obtain the global model, extracts the joint embedded message, and performs verification to confirm the correctness of the aggregation result. The specific details of VPAFL (Algorithm 1) is illustrated in protocol in particular, the proposed protocol requires two rounds of interaction, with the specific processes for processes for each round described below:

Algorithm 1: Details of our VPAFL

1: **Round 0 (Generation of public parameters)**

2: **TA:**

- Select a secure parameter κ and a threshold $t(t < n)$ for the **SS** protocol [35]. Define the user set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$.
- Generate public parameters PP and secret keys SK and $SK_i (i \in \{1, 2, \ldots, n\})$ by performing **SADH.Gen**$(\kappa, t, \mathcal{U})$. Distribute the public parameter N^2 to server and $\{SK, SK_i\}$ to each user $u_i \in \mathcal{U}$.
- Compute public parameters pp by performing **LHH.Gen**(κ, v), where v is a 1-dimensional vector.
- For each user $u_i \in \mathcal{U}$, generate a digital signature key pair $\{ssk_i, spk_i\} \leftarrow$ **DS.Gen**$(\cdot)$, where **DS** denotes a digital signature algorithm.
- Initialize the global model $\mathbb{M}$.
- Send public keys $\{PP, pp\}$, global model $\mathbb{M}$, $\{spk_i\}_{i \in \{1,2,\ldots,n\}}$ to all user and assign secret key ssk_i to user $u_i \in \mathcal{U}$.

3: **For each user u_i in $\mathcal{U}$:**

- Train the local model $\mathbb{M}_i^{k-1}$ using private dataset, where k denotes the current communication round, and send a signal of training completion to the server after training.

4: **Server:**

- Upon receiving signals from at least t users, add the user u_i to the set $\mathcal{U}_1$ and compute $\prod_{j \in \mathcal{U}_1, j \neq i} \frac{j}{j-i}$ for each $u_i \in \mathcal{U}_1$.
- Send $\prod_{j \in \mathcal{U}_1, j \neq i} \frac{j}{j-i}$ and the set $\mathcal{U}_1$ $(|\mathcal{U}_1| = c)$ to each user $u_i \in \mathcal{U}_1 \subseteq \mathcal{U}$.

5: **Round 1 (Model encryption & message embedding, model aggregation)**

(Continued)

Algorithm 1 (continued)

6: **For each user u_i in parallel:**

- Compute the ciphertext of $\mathbf{w}_i^{k-1}$ and embed the secret message d_i: $[[\mathbf{w}_i^{k-1}]] \leftarrow \textbf{SADH.Ence}(SK_i, \mathbf{w}_i^{k-1}, d_i)$, where $\mathbf{w}_i^{k-1}$ is the parameters of model $\mathbb{M}_i^{k-1}$.
- Generate the hash value $h_i \leftarrow \textbf{LHH.Hash}(pp, d_i)$ and signature $sig_{h_i} \leftarrow \textbf{DS.Sign}(h_i, ssk_i)$ for verification.
- Send $\{[[\mathbf{w}_i^{k-1}]], h_i, sig_{h_i}\}$ to the server.

7: **Server:**

- Execute the $\textbf{SADH.Agg}(\{[[\mathbf{w}_i^{k-1}]]_{u_i \in \mathcal{U}_1}\})$ to get the aggregation of encrypted models parameter $[[\mathbf{w}^{k-1}]]$.
- Send $\{[[\mathbf{w}^{k-1}]], \{h_j, sig_{h_j}\}_{u_j \in \mathcal{U}_1, j \neq i}\}$ to the user $u_i \in \mathcal{U}_1$.

8: **Round 2 (Model decryption and verification)**

9: **For each user u_i in parallel:**

- Check whether the hash values and the signatures $\{h_j, sig_{h_j}\}_{u_j \in \mathcal{U}_1, j \neq i}$ are same to those send to the server in **Round 1**, abort if it fails.
- Execute the **SADH.Dece** $(SK, [[\mathbf{w}^{k-1}]])$ to get the aggregation of models parameter $\mathbf{w}^{k-1}$ and extract the joint embedded message d.
- Calculate $h \leftarrow \textbf{LHH.Hash}(pp, d)$ and check:

  $h \stackrel{?}{=} \textbf{LHH.Eval}(pp, h_1, \ldots, h_c, 1, \ldots, 1)$.

  Update the global model $\mathbb{M}^k$ if it holds, abort otherwise.

In **Round 0**, TA initializes the system, distributes the public parameters PP, pp, the global model $\mathbb{M}$, and $\{spk_i\}_{i \in \{1,2,\ldots,n\}}$, assigns private keys ssk_i to each user, and transmits the public parameter N^2 to the servers.

In **Round 1**, each user $u_i \in \mathcal{U}_1$ encrypts their local model $\mathbf{w}_i^{k-1}$ while embedding a secret message d_i, obtaining the ciphertext $[[\mathbf{w}_i^{k-1}]] \leftarrow \textbf{SADH.Ence}(SK_i, \mathbf{w}_i^{k-1}, d_i)$, where k denotes the current communication round. To optimize efficiency, for models with multiple parameters, the user embeds d_i solely during the encryption of the first parameter, and subsequent parameters replace the embedded message with a random number $r_i \in \mathbb{Z}_N$. As demonstrated in Section 6.3, embedding the message in even one parameter suffices to verify the aggregation results. Subsequently, the user $u_i \in \mathcal{U}_1$ computes the LHH value h_i of the embedded message d_i and generates a digital signature sig_{h_i} for h_i. These values will later be used to verify the aggregated result. Finally, the user $u_i \in \mathcal{U}_1$ sends $\{[[\mathbf{w}_i^{k-1}]], h_i, sig_{h_i}\}$ to the server. Upon receiving the ciphertexts, the server aggregates them and returns $\{[[\mathbf{w}^{k-1}]], \{h_j, sig_{h_j}\}_{u_j \in \mathcal{U}_1, j \neq i}\}$ to each user $u_i \in \mathcal{U}_1$.

In **Round 2**, each user $u_i \in \mathcal{U}_1$ first checks the validity of the received digital signature $\{sig_{h_j}\}_{u_j \in \mathcal{U}_1, j \neq i}$ to ensure the integrity of the associated hash values $\{h_j\}_{u_j \in \mathcal{U}_1, j \neq i}$. If valid, the user decrypts the aggregation result to obtain the global model and the joint embedded message $(\mathbf{w}^{k-1}, d) \leftarrow \textbf{SADH.Dece}(SK, [[\mathbf{w}^{k-1}]])$. Next, the user verifies the aggregation result by checking whether $\textbf{LHH.Hash}(pp, d) \stackrel{?}{=} \textbf{LHH.Eval}(pp, h_1, \ldots, h_c, 1, \ldots, 1)$, where $c = |\mathcal{U}_1|$ denotes the total number of participating users. If equality holds, the verification succeeds and the user accepts the aggregation result as valid.

6 Theoretical and Comparative Analysis

6.1 Correctness

We define correctness as the ability to ensure that the user gets the correct aggregation result to update the local model when all entities involved in the FL honestly perform the predetermined operations in the protocol.

Theorem 1. *The user can obtain the correct aggregation result if at least t users participate in FL and the server performs the aggregation operation honestly, where t is the threshold of the* **SS** *protocol [35].*

Proof of Theorem 1: Assume $k = 1$. From Eq. (4), the user u_i applies the encryption-embedding hybrid algorithm $\mathbf{SADH.Ence}(SK_i, \mathbf{w}_i, d_i)$ to encrypt the local model $\mathbf{w}_i$ and embed the message d_i as follows:

$$[[\mathbf{w}_i]] = g^{\mathbf{w}_i} \cdot e^{d_i} \cdot SK_i^{(n-1)! \prod_{j \in \mathcal{U}_1, j \neq i} \frac{j}{j-i}} \mod N^2. \tag{7}$$

Then, as shown in Protocol 1, the ciphertext received by server aggregation in **Round 1** is as follows:

$$\begin{aligned}[[\mathbf{w}]] &= \prod_{u_i \in \mathcal{U}_1} [[\mathbf{w}_i]] \mod N^2 \\ &= \prod_{u_i \in \mathcal{U}_1} g^{\mathbf{w}_i} \cdot e^{d_i} \cdot SK_i^{(n-1)! \gamma_i} \mod N^2 \\ &= g^{\sum_{u_i \in \mathcal{U}_1} \mathbf{w}_i} \cdot e^{\sum_{u_i \in \mathcal{U}_1} d_i} \cdot s^{q(n-1)! \cdot \sum_{u_i \in \mathcal{U}_1} f(i) \gamma_i} \mod N^2,\end{aligned} \tag{8}$$

where $\gamma_i = \prod_{j \in \mathcal{U}_1, j \neq i} \frac{j}{j-i}$, and $SK_i = s^{q \cdot f(i)} \mod N^2$. Based on **SS** protocol [35] and Lagrange interpolation formula, we can define a polynomial as follows:

$$F(x) = \sum_{u_i \in \mathcal{U}_1} f(i) \prod_{j \in \in \mathcal{U}_1, j \neq i} \frac{j - x}{j - i} \mod p \tag{9}$$

From Eq. (3), $F(0) = 0$. When $|\mathcal{U}_1| \geq t$, there is $\sum_{u_i \in \mathcal{U}_1} f(i) \gamma_i = a \cdot p$, where $a \in \mathbb{Z}_p$, thus Eq. (8) can be modified as follows:

$$\begin{aligned}[[\mathbf{w}]] &= g^{\sum_{u_i \in \mathcal{U}_1} \mathbf{w}_i} \cdot e^{\sum_{u_i \in \mathcal{U}_1} d_i} \cdot s^{a(n-1)! \cdot N} \mod N^2 \\ &= g^{\sum_{u_i \in \mathcal{U}_1} \mathbf{w}_i} \cdot g^{\sigma \cdot \sum_{u_i \in \mathcal{U}_1} d_i} \cdot s^{a(n-1)! \cdot N} \mod N^2,\end{aligned} \tag{10}$$

where $e = g^{\sigma} \mod N^2$. Finally. users decrypt $[[\mathbf{w}]]$ as:

$$\begin{aligned}\mathbf{w} = \sum_{u_i \in \mathcal{U}_1} \mathbf{w}_i &= \frac{L([[\mathbf{w}]]^{\lambda} \mod N^2)}{L(g^{\lambda} \mod N^2)} \mod N \mod \sigma \\ &= \left(\sum_{u_i \in \mathcal{U}_1} \mathbf{w}_i + \sigma \cdot \sum_{u_i \in \mathcal{U}_1} d_i \right) \mod \sigma \\ &= \sum_{u_i \in \mathcal{U}_1} \mathbf{w}_i.\end{aligned} \tag{11}$$

This concludes the proof. □

6.2 Privacy-Preserving

This section begins by analyzing the security of **SADH** and subsequently evaluates the security of VPAFL under two adversarial scenarios: (1) a malicious server operating independently and (2) a malicious server colluding with up to $t-1$ users.

Theorem 2. *If* **DTSA** *is indistinguishability under chosen-plaintext attack (IND-CPA) security, then the* **SADH** *is IND-CPA security.*

Proof of Theorem 2: As mentioned earlier, according to Eq. (4), the user replaces the random number used in the encryption process with the product of a decimal number and the random number $r_i \in \mathbb{Z}_N$. This substitution does not compromise security, as the decimal multiplier does not alter the underlying randomness of r_i. Therefore, the modifications introduced by **SADH** to **DTSA** [20] do not weaken the security of the scheme. Furthermore, **DTSA** [20] is IND-CPA security, then the security of **SADH** is guaranteed.

Theorem 3. *(Security Against Malicious Server) In the absence of collusion between the malicious server and semi-honest users, the privacy of all honest users is preserved.*

Proof of Theorem 3: To formally prove privacy guarantees, we employ a standard hybrid argument [37], proved as follows: Under the (**SADH**, **LHH**)-hybrid model, assuming that the security parameter of VPAFL is κ, the threshold of **SS** protocol is t, and the total number of users is n. For simplicity, we define Server as S, $V = S \cap \mathcal{U}_1$, where $\mathcal{U}_1 = \{u_1, \ldots, u_c\}$, $c \leq n$ and $\mathcal{U}_1 \in \mathcal{U}$. The joint view of all entities in V can be denoted as a random variable $\mathbf{REAL}_V^{\mathcal{U},t,\kappa}$. There is also a probabilistic polynomial-time (PPT) simulator $\mathbf{SIM}_V^{\mathcal{U},t,\kappa}$. In order to prove the security of VPAFL, it is necessary to prove that the outputs of $\mathbf{SIM}_V^{\mathcal{U},t,\kappa}$ and $\mathbf{REAL}_V^{\mathcal{U},t,\kappa}$ are indistinguishable, which is formally expressed as follows:

$$\mathbf{REAL}_V^{\mathcal{U},t,\kappa} \equiv \mathbf{SIM}_V^{\mathcal{U},t,\kappa}. \tag{12}$$

hyb0 First, we create a series of random variables, which are indistinguishable from the joint real view of V in $\mathbf{REAL}_V^{\mathcal{U},t,\kappa}$ in the actual implementation of the VPAFL.

hyb1 In this hybrid, we change the behavior of the user u_i through the simulator, and replace the real local model $\mathbf{w}_i$ with the random vector v_i. Then, the user u_i is invoked to encrypt the random vector v_i and embed the message d_i. Note that in Section 5, d_i is defined as the product of two components: (1) a decimal number converted from a binary sequence, and (2) a random number $r_i \in \mathbb{Z}_N$. However, in the simulation phase, the simulator replaces d_i with an random value $r_a \in \mathbb{Z}_N$. The encryption process is adjusted as follows:

$$[[v_i]] = \leftarrow \mathbf{SADH.Ence}(SK_i, v_i, r_a), \tag{13}$$

where SK_i denotes the secret key of user u_i. The simulator then invokes the user u_i to calculate the hash value of r_a: $h_i \leftarrow \mathbf{LHH.Hash}(pp, r_a)$, and generates the digital signature of h_i: $sig_{h_i} \leftarrow \mathbf{DS.Sign}(h_i, ssk_i)$. Finally, the user u_i sends $\{[[v_i]], h_i, sig_{h_i}\}$ to the server. Reviewing the proof of **Theorem 2**, **SADH** guarantees that $[[v_i]]$ is computationally indistinguishable from $[[\mathbf{w}_i]]$. In particular, it has been proved that the output of LHH is random and indistinguishable under different inputs. Thus, VPAFL guarantees the same distribution between **hyb1** and **hyb0**.

hyb2 In this hybrid, the server aggregates the received ciphertext and returns the aggregation result, the received LHH value and the digital signature to each user:

$$[[v]] \leftarrow \mathbf{SADH.Agg}(\{[[v_i]]_{u_i \in \mathcal{U}_1}\}). \tag{14}$$

Similarly, the security of **SADH** algorithm ensures the same distribution between **hyb2** and **hyb1**.

As mentioned above, based on the security of the **SADH** and **LHH** algorithms, we prove that the view in the PPT simulator $\mathbf{SIM}_V^{\mathcal{U},t,\kappa}$ is indistinguishable from the real view of V in the $\mathbf{REAL}_V^{\mathcal{U},t,\kappa}$, i.e., that the $\mathbf{REAL}_V^{\mathcal{U},t,\kappa} \equiv \mathbf{SIM}_V^{\mathcal{U},t,\kappa}$. □

Theorem 4. *(Security against Colluding Malicious Server and Users) Even if a malicious server colludes with up to $t-1$ users, the privacy of all honest users is preserved.*

Proof of Theorem 4: According to Eq. (7), the user u_i encrypted local model $\mathbf{w}_i$ as follows:

$$\begin{aligned}[[\mathbf{w}_i]] =& g^{\mathbf{w}_i} \cdot e^{d_i} \cdot SK_i^{(n-1)!\prod_{j\in\mathcal{U}_1, j\neq i}\frac{j}{j-i}} \mod N^2 \\ =& g^{\mathbf{w}_i} \cdot e^{d_i} \cdot s^{q\cdot f(i)\cdot\gamma_i(n-1)!},\end{aligned} \tag{15}$$

where $\mathcal{U}_1 = \{u_1, \ldots, u_c\}$, where $t(t \le c \le n)$ denote the threshold of the **SS** protocol [35], and $\gamma_i = \prod_{j\in\mathcal{U}_1, j\neq i}\frac{j}{j-i}$. According to the **SS** protocol [35], if the malicious server colludes with $\ge t$ users, Eq. (15) can be modified as follows:

$$[[\mathbf{w}_i]] = g^{\mathbf{w}_i} \cdot e^{d_i} \cdot s^{a\cdot N\cdot(n-1)!}, \tag{16}$$

where $a \in \mathbb{Z}_p$. At this point, the encrypted local model $[[\mathbf{w}_i]]$ can be decrypted using the private key SK, otherwise it cannot be decrypted. □

According to **Theorem 3** and **Theorem 4**, we further demonstrate the resistance of VPAFL to inference attacks under two threat scenarios: (1) attacks initiated solely by the server and (2) collusion between the server and malicious users.

Regarding the global model, since all users transmit encrypted model parameters, the server exclusively operates on ciphertexts during aggregation. This prevents the server from directly analyzing sensitive information. Even if the server colludes with a user to obtain the private key SK, the inherent lack of auxiliary training data ensures that meaningful inference remains infeasible.

For local updates, in the VPAFL protocol, users employ **SADH** to encrypt local model updates. Under the first threat model (only malicious server), **Theorem 3** guarantees that the server cannot decrypt the encrypted parameters of any user without access to the corresponding private key SK_i. Under the second threat model (server-user collusion), **Theorem 4** ensures confidentiality even if up to $t-1$ users conspire with the server: the colluding parties cannot decrypt the local updates of honest users. Since adversaries cannot analyze relevant sensitive information from ciphertext, VPAFL satisfies security requirements against inference attacks in both scenarios.

6.3 Verifiability

Theorem 5. *We define verifiability as the ability of each client to independently verify the correctness of the aggregation results under the threat model defined in Section 4.2.*

Proof of Theorem 5: According to the operation of the malicious server on the aggregation results, we consider two scenarios to prove the effectiveness of the verification.

Scenario 1 (Partial Model Aggregation): Let the total number of users be n, and let t denote the threshold of the **SS** protocol [35]. Assume the server aggregates the models received from c $(t \le c < n)$ users. According to Eqs. (7) and (8), the results obtained by the server aggregation as follows:

$$[[\mathbf{w}]] = g^{\sum_{i=1}^{c}\mathbf{w}_i} \cdot e^{\sum_{i=1}^{c} d_i} \cdot s^{q(n-1)!\cdot\sum_{i=1}^{c} f(i)\gamma_i} \mod N^2. \tag{17}$$

According to Eq. (10), the user decrypts the ciphertext using private key SK to to obtain the global model $\mathbf{w}$ and extracts the joint embedded message $\sum_{i=1}^{c} d_i$. Subsequently, the user will check if the following equation is true.

$$\textbf{LHH.Hash}\left(pp, \sum_{i=1}^{c} d_i\right) \stackrel{?}{=} \textbf{LHH.Eval}(pp, h_1, \ldots, h_n, 1, \ldots, 1). \tag{18}$$

Based on the definitions in Section 3.2, we have:

$$\textbf{LHH.Eval}(pp, h_1, \ldots, h_n, 1, \ldots, 1) = \prod_{i=1}^{n} h_i = g_1^{\sum_{i=1}^{n} d_i} \neq g_1^{\sum_{i=1}^{c} d_i}. \tag{19}$$

Obviously Eq. (18) does not hold and therefore the aggregation result fails to pass validation.

Scenario 2 (Collusion Attack): The server colludes with c ($c < t$) users to forge aggregated results that pass verification, where t denote the threshold of the **SS** protocol [35]. Assume the total number of users is n ($c < t < n$). To forge an aggregation result that passes the validation, the server must craft a result of the following form:

$$[[\mathbf{w}]] = g^{\Delta \mathbf{w}} \cdot e^{\sum_{i=1}^{n} d_i} \cdot s^{q(n-1)! \cdot \sum_{i=1}^{n} f(i)\gamma_i}. \tag{20}$$

Let $\Delta\mathbf{w}$ denote the modified aggregation results. To forge a verified aggregation result, the server must ensure the joint embedded message $\sum_{i=1}^{n} d_i$ can be extracted from $[[\mathbf{w}]]$. However, since the server colludes with only $c(c < t)$ users, it cannot access the secret messages $\{d_j\}_{j \notin \mathcal{U}_c}$ of non-colluding users and must instead guess their values. Suppose the server attempts this forgery by manipulating the aggregation result as follows:

$$[[\mathbf{w}]] = g^{\Delta \mathbf{w}} \cdot e^{\sum_{i=1}^{c} d_i + \sum_{i=c+1}^{n} \Delta d_i} \cdot s^{q(n-1)! \cdot \sum_{i=1}^{n} f(i)\gamma_i}, \tag{21}$$

where $\sum_{i=c+1}^{n} \Delta d_i$ denotes the guess of the server of the secret messages embedded by non-colluding users. As defined in Section 5, the user extracts the joint embedded message $\sum_{i=1}^{c} d_i + \sum_{i=c+1}^{n} \Delta d_i$ from the ciphertext $[[\mathbf{w}]]$. To verify the correctness of the aggregation results, the user checks if the following equation is true:

$$\textbf{LHH.Hash}\left(pp, \sum_{i=1}^{c} d_i + \sum_{i=c+1}^{n} \Delta d_i\right) \stackrel{?}{=} \textbf{LHH.Eval}(pp, h_1, \ldots, h_n, 1, \ldots, 1), \tag{22}$$

where $\textbf{LHH.Eval}(pp, h_1, \ldots, h_n, 1, \ldots, 1) = \prod_{i=1}^{n} h_i = g_1^{\sum_{i=1}^{n} d_i} \neq g_1^{\sum_{i=1}^{c} d_i + \sum_{i=c+1}^{n} \Delta d_i}$, therefore the forged aggregation result cannot pass the verification. □

6.4 Comparison

We compare VPAFL with existing LHH-based VFL schemes, including VerifyNet [18], VeriFL [24], VPFLI [25], and PriVeriFL [26], as shown in Table 1. VerifyNet [18] and VeriFL [24] are based on the double-masking protocol [23], which effectively protects the local gradients of users but fails to protect the privacy of aggregation results. Moreover, these schemes require multiple rounds of interaction between users and the server, increasing communication overhead. In particular, they do not address the risk that corrupted clients colluding with a malicious server are forged to bypass verification. To mitigate collusion attacks during verification, VPFLI [25], PriVeriFL [26], and our proposed VPAFL employ LHH combined with digital signatures, ensuring collusion-resistant verification.

Table 1: Comparison of VFL protocols

Protocol	Local privacy (WITH collusion)	Aggregation privacy (without collusion)	Collusion-resistant verification	Verification complexity	Rounds of interactions
VerifyNet [18]	✓	×	×	$O(2d)$	4
VeriFL [24]	✓	×	×	$O(d + d/t_r)$	4
VPFLI [25]	✓	✓	✓	$O(2d)$	3
PriVeriFL [26]	✓	✓	✓	$O(2d)$	2
VPAFL (our)	✓	✓	✓	$O(1)$	2

Regarding aggregation privacy, VPFLI [25] introduces a blinding factor to mask the aggregation results, while VPAFL and PriVeriFL [26] delegate decryption authority to users. This approach prevents the server from directly decrypting the aggregation results, thus enhancing privacy protection. However, it is important to note that neither VPFLI [25], PriVeriFL [26], nor VPAFL can fully preserve aggregation results in privacy under collusion attacks. Since FL inherently involves collaborative training of a unified global model, the server only needs to collude with a single user to obtain the global model.

In terms of verification complexity, we assume that the computational complexity of each call to LHH is $O(d)$, where d represents the model dimension. In VerifyNet [18] and VeriFL [24], users perform LHH twice per verification: once to generate the hash value and once to verify the result. Thus, the verification complexity is $O(2d)$ per communication round. To optimize this, VeriFL [24] employs an amortized verification mechanism. Specifically, users sample a set of random coefficients to compute the linear combination of hash aggregations across multiple communication rounds (for example, after t_r rounds). They then verified whether the combined hash matches the hash of the linear combination (using the same coefficients) applied to the aggregation results of those rounds. Consequently, VeriFL [24] reduces the verification complexity to $O(d + d/t_r)$. In contrast, VPAFL further optimizes the process by integrating the proposed RDHED with LHH (see Section 7.4.1 for details), thus reducing the verification complexity to $O(1)$. Moreover, VPAFL requires only two interaction rounds, establishing it as a more efficient protocol compared to existing solutions.

7 Experimental Results

In this section, similar to previous studies [18,25], we evaluate the performance of VPAFL in terms of fidelity, computational overhead, and communication overhead.

7.1 Experimental Settings

This subsection describes the experimental settings for VPAFL.

Model Architectures and Datasets: We employ two deep neural network architectures: AlexNet [38] for the CIFAR-100 [39] classification task and FedCNN for the MNIST [40] and CIFAR-10 [39] classification task.

Federated Learning Settings: Based on the open-source personalized FL framework (https://github.com/TsingZ0/PFLlib, accessed on 7 May 2025) [41], we simulate a horizontal FL environment where users employ the SGD algorithm [34] for local model updates during each communication round, with the server using the FedAvg algorithm [7] for model aggregation.

All experiments were conducted on an Ubuntu 22.04 workstation equipped with an Intel Xeon Platinum 8352V 2.10 GHz CPU, 60 GB RAM, and a single NVIDIA 4090 GPU. Our implementation uses Python 3.9

with stable libraries. The LHH is implemented using NIST P-256 curves. FedCNN is a convolutional neural network with the following architecture: $Conv(x, 32, 5\times5) \rightarrow ReLU \rightarrow MaxPool(2, 2) \rightarrow Conv(32, 64, 5\times5) \rightarrow ReLU \rightarrow MaxPool(2, 2) \rightarrow FC(1024, 512) \rightarrow ReLU \rightarrow FC(512, 10)$, where x denotes the dimension of the input, $Conv$ denotes the convolutional layer, $ReLU$ denotes the type of activation function, $MaxPool$ denotes the pooling layer, and FC denotes the fully connected layer.

7.2 Evaluation Metrics

Fidelity: We measure fidelity using the accuracy of the model on the classification task, denoted by Acc.

Computational and Communication Overhead: Similarly to previous studies [18,24], we measure computational and communication overhead to evaluate the efficiency of the VPAFL.

Baseline: Regarding the selection of the baseline for overhead comparison, we adopt VPFLI [25] because it also uses the DTSA algorithm [20] as the privacy-preserving strategy. In particular, VPFLI [25] introduces a weight aggregation protocol to mitigate the degradation of model performance caused by heterogeneous user data quality. To ensure comparability, we exclusively retain the core modules relevant to VFL during baseline reproduction.

7.3 Fidelity

To validate that the proposed scheme maintains aggregation accuracy without compromising privacy guarantees, we evaluate its performance on three datasets: MNIST [40], CIFAR-10, and CIFAR-100 [39], adopting identical training configurations (e.g., learning rate, batch size) for direct comparison with the FedAvg algorithm [7]. As shown in Fig. 2, the proposed VPAFL protocol achieves an aggregation effectiveness comparable to that of FedAvg [7], with a stable convergence behavior observed across all datasets. This consistency originates from the mathematically lossless encryption and decryption of the plaintext model parameters, thereby preserving data integrity throughout the FL process.

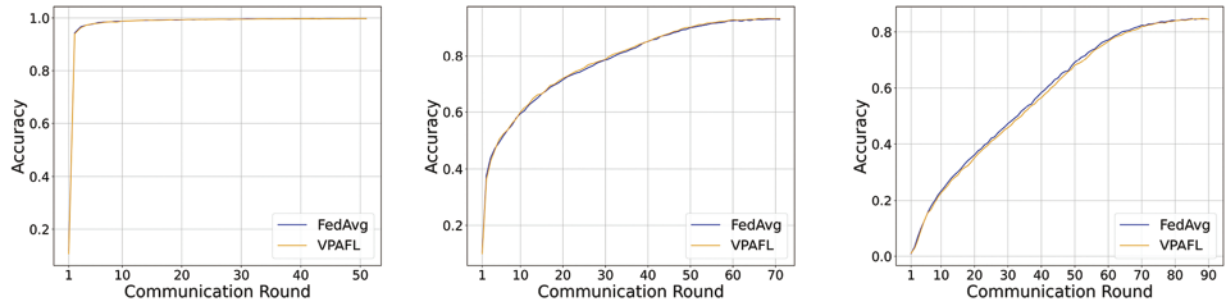

Figure 2: The task accuracy (Acc) on MNIST, CIFAR-10 and CIFAR-100

7.4 Computational and Communication Overhead

To systematically analyze system efficiency, we define n as the number of participating users, $\kappa = 512$ as the security parameter and d as the model dimension. To isolate variable impacts, we evaluate computational and communication overhead under two scenarios: (1) Different number of users (n from 100 to 1000) with fixed model dimension $d = 5000$. (2) Different model dimension (d from 1000 to 10,000) with fixed number of users $n = 500$.

7.4.1 Computational Overhead

The computational overhead of the user in VPAFL is primarily determined from modular arithmetic operations in $\mathbb{Z}_N$ and $\mathbb{Z}_{N^2}$. Let T_{m_1} and T_{m_2} denote the time costs of single modular multiplications in $\mathbb{Z}_N$ and $\mathbb{Z}_{N^2}$, respectively, and T_e for modular exponentiation in $\mathbb{Z}_{N^2}$. Based on Eqs. (4) and (6), the encryption overhead per user is theoretically $d \cdot (3T_e + 2T_{m_2})$, while decryption requires $d \cdot (T_{m_1} + T_e)$. Thus, the total computational overhead per user becomes $d \cdot (T_{m_1} + 2T_{m_2} + 4T_e)$.

Fig. 3 illustrates the computational overhead of a single user per communication round. The computational overhead exhibits a positive correlation with both the number of participating users n and the model dimension d. This growth comes from increasing computational demands in both the model encryption and the decryption phases. Mathematically, as the user count n grows, the exponent of the SK term in Eq. (4) increases, resulting in more computationally intensive operations. The higher model dimension d not only expands the set of encryption parameters, but also prolongs the decryption time. In particular, the VPAFL protocol shows superior operational efficiency compared to VPFLI [25], achieving an average reduction in computational overhead of 10 s.

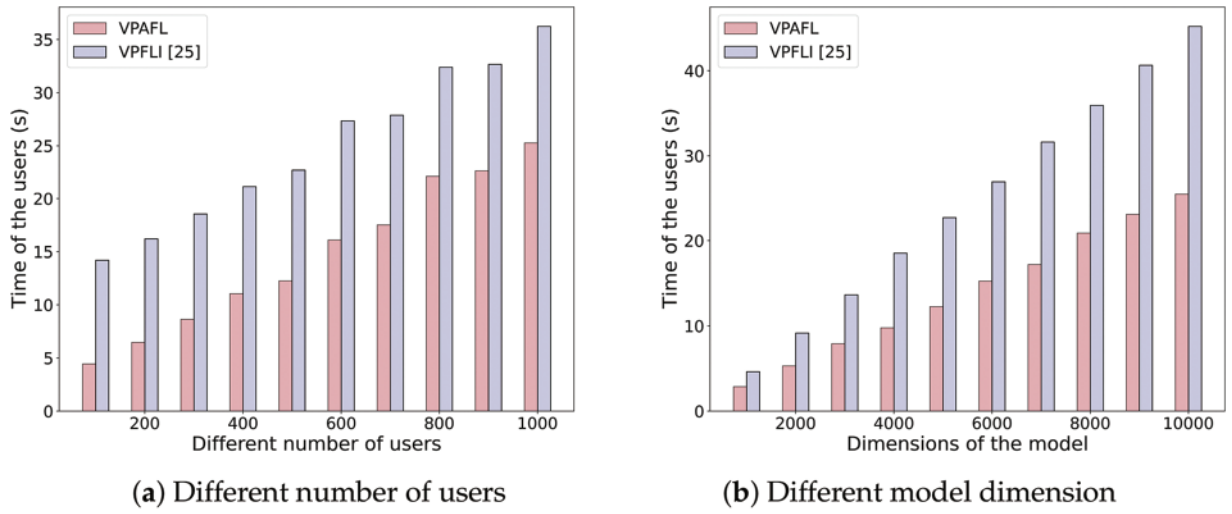

Figure 3: Computational overheads of the users [25]

To further evaluate the efficiency of the verification process, Fig. 4 quantifies the computational overhead for a single user to verify aggregated results. The experimental results show that verification overhead increases with the number of users n, due to the increasing computational demands required to validate a growing number of signatures. Specifically, the verification overhead reaches a maximum of 152.73 ms at $n = 1000$. In contrast, variations in model dimension d exhibit negligible influence on verification overhead, with the results stabilizing at approximately 80 ms in all tested configurations. In particular, VPAFL achieves substantially lower verification overhead than VPFLI [25], reducing the average computational costs by approximately 8 s under identical conditions.

In addition, we further investigate the performance of VPAFL in large-scale user scenarios. To evaluate scalability, we measure the computational overhead per user during validation under a fixed model dimension $d = 1000$ while varying the number of participating users from 1000 to 5000 in increments of 1000 and compared it with VPFLI [25] and PriVeriFL [26].

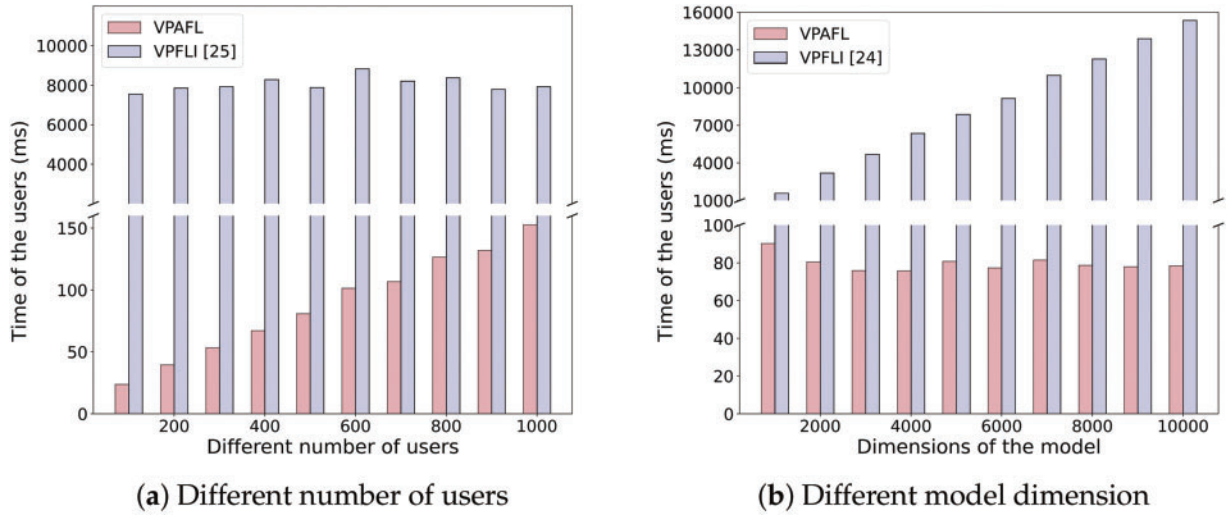

Figure 4: Computational overheads of the users for verification [24,25]

As illustrated in Fig. 5, the results demonstrate that the computational overhead of the users for verification in all three schemes increases with the number of participating users. This growth pattern originates mainly from the cost of **LHH.Eval**, whose computational complexity scales with the number of participating users. While the verification process requires checking more digital signatures as user numbers expand, this component contributes minimal overhead, approximately 0.3 ms even at 5000 users. Note that although VPFLI [25] and PriVeriFL [26] employ distinct cryptographic algorithms, they share the same verification mechanism. Therefore, the computational overheads of the users for verification remain identical in both schemes.

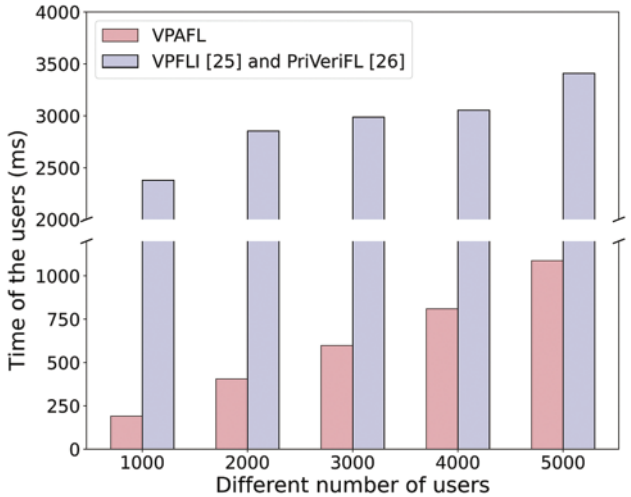

Figure 5: Computational overheads of the users for verification with large user scenarios [25,26]

In particular, VPAFL maintains significantly lower overhead than both VPFLI [25] and PriVeriFL [26]. This advantage arises from differences in LHH implementation, and we elucidate the precise reasons for this disparity in the following analysis.

The performance superiority of VPAFL originates from a fundamental distinction in hash computation mechanisms between the two schemes. Although VPFLI [25] employs LHH to decouple communication overhead from model dimension d, its hash calculation operates directly on the model parameters themselves, resulting in computational demands proportional to d. In contrast, VPAFL integrates RDHED with LHH, restricting the hash computation to the lightweight secret messages embedded by users during each communication round. This strategic change in computational input—from high-dimensional model parameters to compact secret messages—decouples hash-related costs from d, addressing a critical efficiency bottleneck in existing LHH-based VFL schemes [18,24–26]. The benefits of this innovation are significantly amplified in high-dimensional model scenarios, as evidenced by the quantitative comparisons in Table 2. For model sizes of 10,000, 100,000, 1,000,000 and parameters, the hash computation time of VPAFL remains consistently below 4 ms, while the LHH-based VFL [18,24–26] requires up to 12,490 s when d = 10,000,000, a disparity that strikingly validates the efficiency gains enabled by RDHED integration.

Table 2: The time cost of calculating the hash value

Dimensions of the model	Scheme	
	Proposed scheme	**Scheme [18,24–26]**
10,000	2.3 ms	9568.51 ms
100,000	2.89 ms	91,861.36 ms
1,000,000	2.9 ms	895,392.4 ms
10,000,000	3.85 ms	12,490,847.49 ms

In summary, the computational overhead of the users for verification in VPAFL is lightweight, making it more conducive to practical deployment, particularly in scenarios involving large-scale models or numerous participating users.

In VPAFL, the server is responsible for aggregating the encrypted local models received and its computational overhead is $d \cdot (n-1) \cdot T_{m_2}$. Fig. 6 clearly shows that the calculation overhead of the server increases when the number n of users and the model dimension d increase. Among them, the computational overhead of the server in VPAFL is slightly lower than that of the server in VPFLI, because the server in VPAFL only involves aggregating local models and does not decrypt the ciphertext.

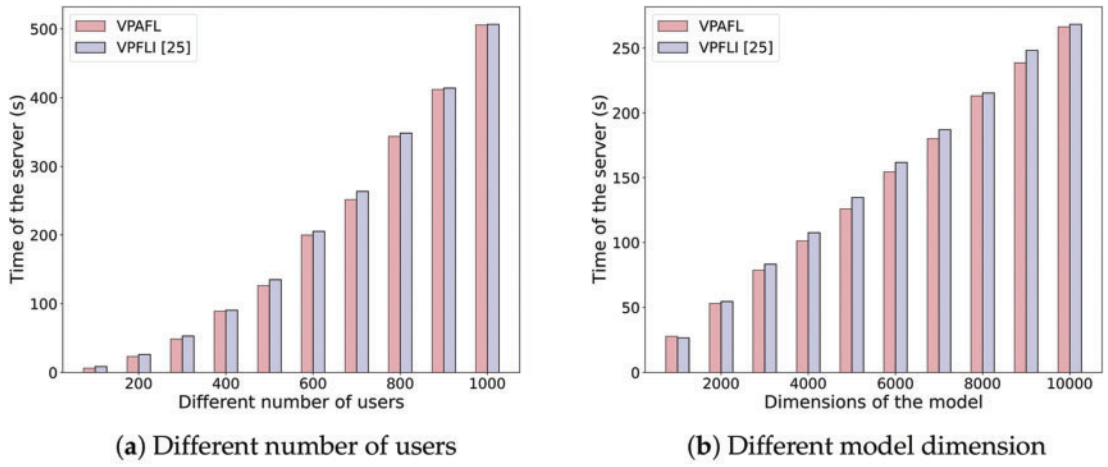

Figure 6: Computational overheads of the server [25]

7.4.2 Communication Overhead

In the experiment, we measure the communication overhead by the size of the uploaded information. As shown in Fig. 7, the communication overhead of the users is not related to the number of users, but related to the model dimension d. With increasing model dimension d, the size of the ciphertext generated by users becomes larger, leading to greater communication overhead. For the server, as shown in Fig. 8, its communication overhead is positively correlated with the number of users n and the model dimension d. The larger the number of users n, the more messages need to be transmitted, and the larger the model dimension d, the larger the ciphertext size generated by server aggregation.

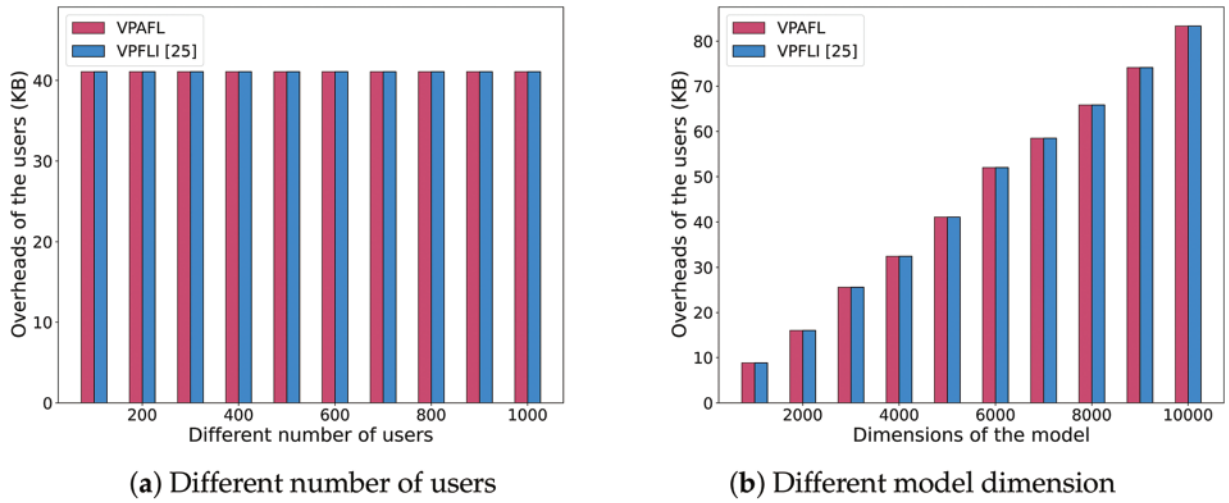

Figure 7: Communication overheads of the users [25]

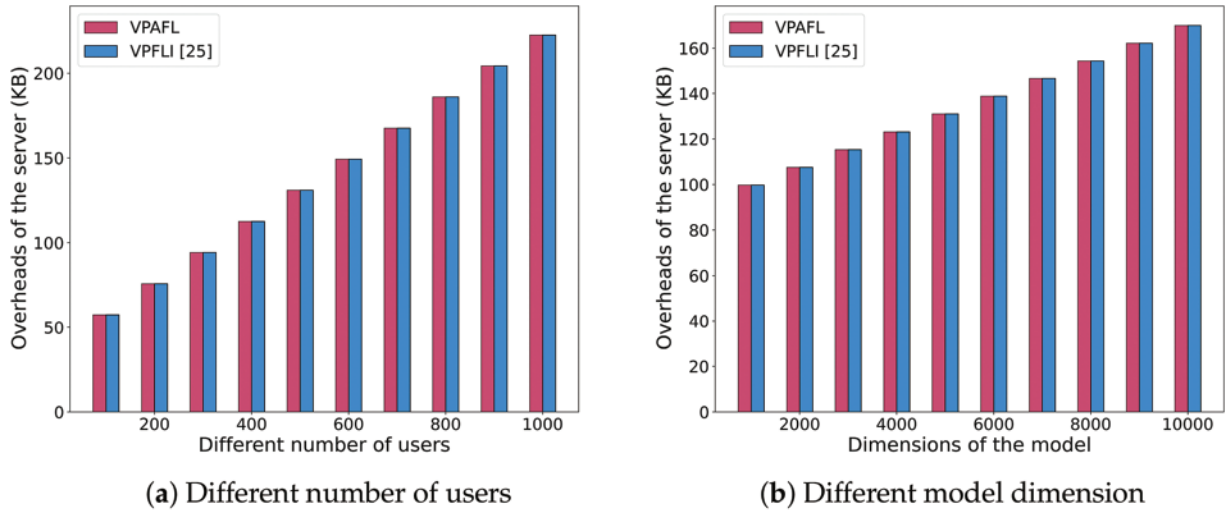

Figure 8: Communication overheads of the server [25]

Finally, we evaluate the communication overhead associated with the verification. Since our implementation reproduces only the verifiable module of VPFLI [25] while maintaining consistency with the proposed scheme in other components, both schemes exhibit identical communication overhead under

identical experimental configurations. For the user, LHH and digital signature algorithms transform variable length messages into fixed size outputs, thus decoupling verification-related communication overhead from both the number of users n and the model dimension d. Conversely, the verification-related communication overhead of the server is determined by the cost of broadcasting the received hash values and signatures to all users, causing the communication costs of the server to scale linearly with the number of users n.

Fig. 9 shows the communication overhead for individual users and servers under fixed model dimensions d = 5000 and varying numbers of participating users n. The results show that the communication overhead for verification per user remains below 0.2 KB regardless of n, while the server overhead increases linearly with n due to the increasing volume of hash values and signatures broadcast. Importantly, since user devices typically face stringent computational and storage resource constraints compared to servers, the low verification-related communication overhead for users (0.2 KB) in VPAFL enhances its practicality for real-world deployment.

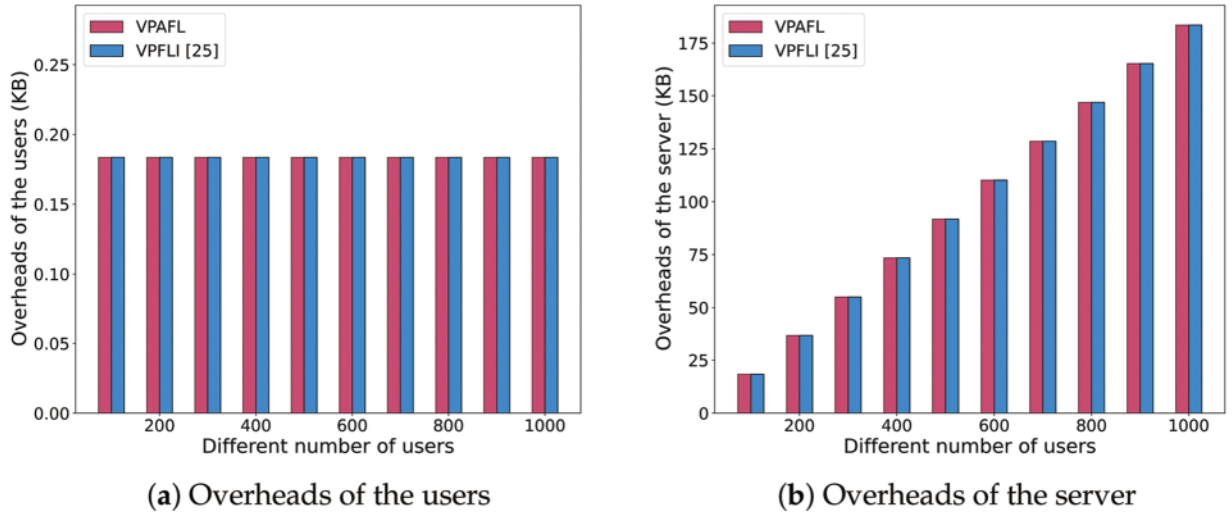

Figure 9: Communication overheads for verification when d = 5000 with different number of users [25]

8 Conclusion

In this paper, we propose a reversible data hiding in encrypted domains (RDHED) scheme that designs a joint message embedding and extraction mechanism. Building on this RDHED scheme, we further design VPAFL, a verifiable privacy-preserving aggregation protocol for single-server architectures, by combining linear homomorphic hash and digital signature algorithms. Unlike prior verifiable federated learning schemes based on linear homomorphic hash, VPAFL computes hash values using secret messages embedded by users during each communication round, thereby decoupling the computational overhead of hash generation from the model dimension. Theoretical analysis demonstrates the security and feasibility of VPAFL, while the experiment results confirm that the computational and communication overheads of the users for verification are lightweight.

Acknowledgement: The authors appreciate the valuable comments from the reviewers and editors.

Funding Statement: This work was supported in part by the National Natural Science Foundation of China under Grants 62102450, 62272478 and the Independent Research Project of a Certain Unit under Grant ZZKY20243127.

Author Contributions: Conceptualization: Peizheng Lai, Minqing Zhang; Experimental operation and data proofreading: Peizheng Lai, Yixin Tang, Ya Yue; Analysis and interpretation of results: Peizheng Lai, Minqing Zhang, Fuqiang Di; Draft manuscript preparation: Peizheng Lai, Ya Yue; Figure design and drawing: Peizheng Lai, Yixin Tang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used to support the findings of this study are publicly available on Internet as follows: MNIST: http://yann.lecun.com/exdb/mnist/ (accessed on 24 March 2025); CIFAR-10 and CIFAR-100: https://www.cs.toronto.edu/kriz/cifar.html (accessed on 24 March 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Tari Z, Yi X, Premarathne US, Bertok P, Khalil I. Security and privacy in cloud computing: vision, trends, and challenges. IEEE Cloud Computing. 2015;2(2):30–8. doi:10.1109/MCC.2015.45.
2. Cao X, Du L, Wei X, Meng D, Guo X. High capacity reversible data hiding in encrypted images by patch-level sparse representation. IEEE Trans Cybern. 2015;46(5):1132–43. doi:10.1109/TCYB.2015.2423678.
3. Shi YQ, Li X, Zhang X, Wu HT, Ma B. Reversible data hiding: advances in the past two decades. IEEE Access. 2016;4:3210–37. doi:10.1109/access.2016.2573308.
4. Qian Z, Zhou H, Zhang X, Zhang W. Separable reversible data hiding in encrypted JPEG bitstreams. IEEE Trans Depend Secure Comput. 2016;15(6):1055–67. doi:10.1109/tdsc.2016.2634161.
5. Zheng S, Li D, Hu D, Ye D, Wang L, Wang J. Lossless data hiding algorithm for encrypted images with high capacity. Multimed Tools Appl. 2016;75(21):13765–78. doi:10.1007/s11042-015-2920-y.
6. Puteaux P, Puech W. An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. IEEE Trans Inf Forensics Security. 2018;13(7):1670–81. doi:10.1109/TIFS.2018.2799381.
7. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics; 2017; New York, NY, USA: PMLR. p. 1273–82
8. Wu HT, Cheung YM, Tian Z, Liu D, Luo X, Hu J. Lossless data hiding in NTRU cryptosystem by polynomial encoding and modulation. IEEE Trans Inf Forensics Security. 2024;19(11):3719–32. doi:10.1109/TIFS.2024.3362592.
9. Anushiadevi R, Amirtharajan R. Design and development of reversible data hiding-homomorphic encryption & rhombus pattern prediction approach. Multimedia Tools Appl. 2023;82(30):46269–92. doi:10.1007/s11042-023-15455-1.
10. Zhou N, Zhang M, Wang H, Ke Y, Di F. Separable reversible data hiding scheme in homomorphic encrypted domain based on NTRU. IEEE Access. 2020;8:81412–24. doi:10.1109/ACCESS.2020.2990903.
11. Wu HT, Cheung YM, Zhuang Z, Xu L, Hu J. Lossless data hiding in encrypted images compatible with homomorphic processing. IEEE Trans Cybern. 2022;53(6):3688–701. doi:10.1109/TCYB.2022.3163245.
12. Zheng S, Wang Y, Hu D. Lossless data hiding based on homomorphic cryptosystem. IEEE Trans Depend Secure Comput. 2019;18(2):692–705. doi:10.1109/TDSC.2019.2913422.
13. Xiang S, Luo X. Reversible data hiding in homomorphic encrypted domain by mirroring ciphertext group. IEEE Trans Circuits Syst Video Technol. 2018;28(11):3099–110. doi:10.1109/TCSVT.2017.2742023.
14. Ke Y, Zhang MQ, Liu J, Su TT, Yang XY. Fully homomorphic encryption encapsulated difference expansion for reversible data hiding in encrypted domain. IEEE Trans Circuits Syst Video Technol. 2020;30(8):2353–65. doi:10.1109/TCSVT.2019.2963393.
15. Hitaj B, Ateniese G, Perez-Cruz F. Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017; New York, NY, USA. p. 603–18.
16. Zhu L, Liu Z, Han S. Deep leakage from gradients. Adv Neural Inf Process Syst. 2019;32:14747–56.
17. Ma C, Li J, Ding M, Yang HH, Shu F, Quek TQ, et al. On safeguarding privacy and security in the framework of federated learning. IEEE Network. 2020;34(4):242–8. doi:10.1109/MNET.001.1900506.

18. Xu G, Li H, Liu S, Yang K, Lin X. VerifyNet: secure and verifiable federated learning. IEEE Trans Inf Forensics Security. 2019;15:911–26. doi:10.1109/TIFS.2019.2929409.
19. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques; 1999; Berlin/Heidelberg: Springer. p. 223–38.
20. Zhao J, Zhu H, Wang F, Lu R, Li H, Tu J, et al. CORK: a privacy-preserving and lossless federated learning scheme for deep neural network. Inform Sciences. 2022;603(3):190–209. doi:10.1016/j.ins.2022.04.052.
21. Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, et al. Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans Inf Forensics Security. 2020;15:3454–69. doi:10.1109/TIFS.2020.2988575.
22. Wei K, Li J, Ma C, Ding M, Chen W, Wu J, et al. Personalized federated learning with differential privacy and convergence guarantee. IEEE Trans Inf Forensics Security. 2023;18:4488–503. doi:10.1109/TIFS.2023.3293417.
23. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017; New York, NY, USA. p. 1175–91.
24. Guo X, Liu Z, Li J, Gao J, Hou B, Dong C, et al. VeriFL: communication-efficient and fast verifiable aggregation for federated learning. IEEE Trans Inf Forensics Security. 2020;16:1736–51. doi:10.1109/TIFS.2020.3043139.
25. Ren Y, Li Y, Feng G, Zhang X. VPFLI: verifiable privacy-preserving federated learning with irregular users based on single server. IEEE Trans Services Computing. 2024;18(2):1124–36. doi:10.1109/TSC.2024.3520867.
26. Wang L, Polato M, Brighente A, Conti M, Zhang L, Xu L. PriVeriFL: privacy-preserving and aggregation-verifiable federated learning. IEEE Trans Services Computing. 2024;18(2):998–1011. doi:10.1109/TSC.2024.3451183.
27. Hahn C, Kim H, Kim M, Hur J. VERSA: verifiable secure aggregation for cross-device federated learning. IEEE Trans Depend Secure Comput. 2021;20(1):36–52. doi:10.1109/TSC.2024.3451183.
28. Xu Y, Zhang H, Zhao S, Zhang X, Li W, Gao F, et al. Comments on "VERSA: verifiable secure aggregation for cross-device federated learning". IEEE Trans Depend Secure Comput. 2024;21(4):4297–8. doi:10.1109/TDSC.2023.3272338.
29. Luo F, Wang H, Yan X. Comments on "VERSA: verifiable secure aggregation for cross-device federated learning". IEEE Trans Depend Secure Comput. 2024;21(1):499–500. doi:10.1109/TDSC.2023.3253082.
30. Bellare M, Goldreich O, Goldwasser S. Incremental cryptography: the case of hashing and signing. In: Advances in Cryptology-CRYPTO'94: 14th Annual International Cryptology Conference; 1994 Aug 21–25; Santa Barbara, CA, USA: Springer. p. 21–5.
31. Huang H, Huang T, Wang W, Zhao L, Wang H, Wu H. Federated learning and convex hull enhancement for privacy preserving WiFi-based device-free localization. IEEE Trans Consum Electron. 2024;70(1):2577–85. doi:10.1109/TCE.2023.3342834.
32. Tian Y, Wang S, Xiong J, Bi R, Zhou Z, Bhuiyan MZA. Robust and privacy-preserving decentralized deep federated learning training: focusing on digital healthcare applications. IEEE/ACM Trans Comput Bi. 2024;21(4):890–901. doi:10.1109/TCBB.2023.3243932.
33. Wehbi O, Arisdakessian S, Guizani M, Wahab OA, Mourad A, Otrok H, et al. Enhancing mutual trustworthiness in federated learning for data-rich smart cities. IEEE Internet Things J. 2025;12(3):3105–17. doi:10.1109/JIOT.2024.3476950.
34. Bottou L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics; 2010 Aug 22–27; Paris, France: Springer. p. 177–86.
35. Shamir A. How to share a secret. Commun ACM. 1979;22(11):612–3. doi:10.1145/359168.359176.
36. Jagielski M, Oprea A, Biggio B, Liu C, Nita-Rotaru C, Li B. Manipulating machine learning: poisoning attacks and countermeasures for regression learning. In: 2018 IEEE Symposium on Security and Privacy (SP); 2018; San Francisco, CA, USA. p. 19–35.
37. Gentry C, Groth J, Ishai Y, Peikert C, Sahai A, Smith A. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. J Cryptol. 2015;28(4):820–43. doi:10.1007/s00145-014-9184-y.
38. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst. 2012;25:1097–105.

39. Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Handb Systemic Autoimmune Dis. 2009;1(4):1–60.
40. LeCun Y. The MNIST database of handwritten digits; 1998. [cited 2025 May 7]. Available from: http://yann.lecun.com/exdb/mnist/.
41. Zhang J, Liu Y, Hua Y, Wang H, Song T, Xue Z, et al. PFLlib: personalized federated learning algorithm library. arXiv:231204992. 2023.