Verifiable Distributed Computing via Multi-layer Data Integrity Auditing

Wenying Zheng[1], Zelin Ni[1,5], Tianqi Zhou[2,3,4]*

[1] School of Computer Science and Technology (School of Artificial Intelligence), Zhejiang Sci-Tech University, China
[2] School of Information Science and Engineering (School of Cyber Science and Technology), Zhejiang Sci-Tech University, China
[3] Zhejiang Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University, China
[4] Zhejiang Provincial Innovation Center of Advanced Textile Technology, China
[5] Faculty of Data Science, City University of Macau, China
zhengwy0501@126.com, 1486602868@qq.com, tq_zhou@126.com

Abstract

Distributed learning is regarded as a key solution to support edge computing in the era of digital prosperity. It significantly reduces the data transmission and privacy risks in centralized machine learning through local training with "data not leaving the edge". However, the quality of client data in distributed computing is often uneven, leading to inaccurate uploaded gradients and affecting the convergence and performance of the global model. To address the decline in model performance caused by untrusted client data, this paper proposes a federated learning admission and continuous verification mechanism based on data possession proofs for heterogeneous computing power. This mechanism will perform corresponding dataset verification based on the different computing power of clients. For strong devices, a complete S-PDP verification is executed, and the generated proof is signed with BLS to support multi-signature aggregation verification on the server side; for weak devices, a lightweight E-PDP is executed, only conducting a small number of samplings to reduce the overhead, and multiple rounds of verification are performed to enhance detection capabilities, and the proof is not signed to further reduce the burden on the edge side. Additionally, within the federated learning training cycle, this paper innovatively introduces a continuous random sampling verification mechanism to eliminate clients that fail to pass the proof for consecutive times, thereby continuously filtering out data anomalies or fake nodes while ensuring resource-friendliness, and enhancing the credibility of the federated training process and the robustness of the global model.

Keywords: Distributed learning, Proof of data possession, Hierarchical verification, BLS signature, Edge computing

1 Introduction

With the rapid development of artificial intelligence, an increasing number of datasets need to be leveraged for model training. For the purpose of protecting data privacy, federated learning (FL) [1] has been widely adopted, largely owing to its advanced capability for distributed computing and privacy protection [2-3]. Federated learning reduces the need to transfer massive volumes of data to the cloud for centralized training. To be more precisely, participating clients are not required to upload all local data; instead, they train the model locally and upload the resulting model parameters to a cloud server. The server then aggregates the received gradients to construct a global model. Consequently, federated learning has been applied in numerous fields, including medical imaging [4], smart cities [5], financial investment [6], and autonomous driving [7].

Despite these advantages, when data stored on clients are inaccurate or incomplete, the learned model parameters often fail to meet expectations, which makes it necessary to perform prior validation of data stored on clients. In addition, when verified clients upload their data possession certificates to the server, attackers may attempt to forge these certificates or reuse previously valid ones; therefore, further verification of the submitted certificates is necessary.

To address these challenges, message authentication code (MAC) techniques [8] and digital signature schemes [9] are commonly employed. In MAC-based methods, the sender and verifier share a secret key and authenticate messages through MAC verification. Such methods are efficient and tamper-resistant; however, such approaches lack public verifiability, which prevents third-party auditors from assisting the server in an efficient manner. In contrast, digital signatures are generated using a sender's private key and can be verified by anyone holding

*Corresponding Author: Tianqi Zhou; Email: tq_zhou@126.com
DOI: https://doi.org/10.70003/160792642026032702001O

the corresponding public key, thereby enabling public verification and broad applicability. Nevertheless, RSA signatures typically require longer keys and have relatively slow verification speeds, while ECDSA, although more efficient, still has higher computational costs than HMAC-based techniques.

1.1 Motivations

To ensure the accuracy of customer data in distributed learning under different computing capacity environments, this paper proposes a distributed learning framework with a hierarchical data integrity auditing function, aiming to improve the overall model's accuracy.

- Ensure the accuracy of data for different clients. Due to the significant differences in computing resources among clients, it is impractical to enforce a uniform data validation scheme for all participants. Therefore, the use of a hierarchical auditing method is crucial for verifying the accuracy and integrity of client data sets in a resource-aware manner.
- Reduce the impact of malicious clients to improve model accuracy. During the training process, some clients may launch attacks related to membership or act in a hostile manner, deliberately providing false data, thereby reducing the accuracy of the model. Accordingly, for the server or a third-party auditor, deploying a comprehensive reputation evaluation mechanism can reduce accuracy loss and enable more reliable model training.

1.2 Contributions

Building on the above, we propose a distributed learning scheme with hierarchical data integrity auditing, implemented via bilinear pairings, cryptographic hash functions, and an evaluation function. The main contributions are summarized as follows:

- **A computation-aware, hierarchical data integrity auditing scheme for assessing dataset quality in distributed learning.** In this scheme, the server adopts different verification methods for clients with different computational capacities, ensuring that resource-constrained clients can still complete verification. Moreover, resource-rich clients additionally sign their generated proofs to prevent proof forgery, thereby safeguarding the security of critical data held by strong devices.
- **An admission mechanism and a continuous verification mechanism to reduce the likelihood of malicious clients injecting falsified data.** By requiring PDP as a prerequisite for joining federated learning, each client must prove the integrity and authenticity of a designated dataset before participation. This filters, at the source, nodes whose data are missing, tampered with, or fabricated, reducing the impact of contaminated data on the global model while avoiding disclosure of raw data to preserve privacy and compliance. During training, we further introduce lightweight random-sampling checks and an eviction policy for consecutive verification failures. With probabilistic detection guarantees, the proposed design achieves low-cost, high-confidence online risk control under bounded communication and computation overhead.
- **Comprehensive validation of security and performance.** In terms of security analysis, we show—based on a probabilistic sampling model—that the proposed hierarchical auditing strategy can achieve a 99% detection rate. In terms of efficiency, strong devices employ BLS aggregate signatures, enabling a single verification to confirm signer identity and bind the signature to the corresponding time and message; in large-scale concurrent settings, this reduces verification overhead to **constant time**.

2 Related Work

To mitigate the adverse impact of heterogeneous client data quality on the global model, prior studies have proposed NSPFL—a secure and privacy-preserving federated learning framework with data integrity auditing [10-11]—which introduces a scoring mechanism to select participating clients. In addition, some work has presented a pairing-based certificateless signature (PCLS) scheme and, building on it, an anonymous authentication protocol for privacy-preserving distributed learning (DL) [12]. Regarding privacy protection, secure aggregation, homomorphic encryption, and differential privacy are commonly adopted to reduce the risk of information leakage. Although these methods can tolerate noise or adversarial gradients to some extent, they share a common limitation: filtering or reweighting is typically performed only after updates have been generated, and thus cannot guarantee, at the source, that a client indeed possesses the designated dataset and that the dataset has not been tampered with.

In contrast, our scheme treats Provable Data Possession (PDP) [13] as a prerequisite for admission. There are mainly two types of data possession verification schemes. One is the S-PDP (Secure Provable Data Possession) with high security, and the other is the E-PDP (Efficient Provable Data Possession) that focuses on efficiency. Before submitting any updates, each client must first prove the integrity and authenticity of the required dataset, thereby filtering out missing or tampered data at the source. Furthermore, we introduce continuous random-sampling verification during the training process to enable low-overhead re-auditing, and promptly evict clients that repeatedly fail to provide valid proofs. As a result, data-quality assurance is elevated from post-hoc correction to a probabilistically guaranteed gatekeeping and process-level supervision mechanism.

Existing VPFL schemes [14] center on verifiability. Prior work has investigated a range of cryptographic techniques, including digitally signing each round's updates, constructing auditable logs/ledgers, and employing zero-knowledge proofs [15] to verify the correctness of

protocol steps without revealing the underlying data. While these approaches improve accountability and process auditability, achieving a practical balance between strong verification and communication/computation cost remains challenging in large-scale client settings.

Along this line, our scheme authenticates the PDP proofs submitted by clients using BLS multi-signature aggregation. A single compact verification simultaneously checks the signer's identity and validates the binding among the signature timestamp/date and the message content, offering better scalability than verifying signatures individually. Combined with PDP-based admission and continuous random-sampling verification [16-17], the overall workflow provides effective authentication of proof provenance and content without exposing raw data, maintains low verification overhead at scale, and strengthens end-to-end integrity guarantees.

3 Preliminaries

This section describes federated learning, provable data possession and BLS aggregation signatures.

3.1 Federated Learning

Federated learning (FL) [18] is a distributed learning paradigm in which multiple participants and a coordinating server collaboratively build a single global model without centralizing raw data. At a high level, one FL round proceeds as follows:

- **Step 1:** The server will initialize the global parameters w_0 and selects hyperparameters (e.g., learning rate η, total FL rounds). It then dispatches this information to a subset of clients chosen to engage in the current round.
- **Step 2:** After receiving the broadcasted global weight, each selected client i starts local training on its private dataset D_i. During round t, it produces a local gradient $g_i^{(t)}$ and updates its local model $w_i^{(t)}$. Upon finishing local computation, the client transmits $g_i^{(t)}$ back to the server for aggregation.
- **Step 3:** Having collected the client updates, the server applies an aggregation rule to obtain the global update. A common choice is simple averaging:

$$g^{(t+1)} = \frac{1}{N}\sum_{i=1}^{n} g_i^{(t)}$$

where N is the number of participating clients in round t. The aggregated result is then sent to clients to initialize the next training round.

When a stopping condition is met (e.g., the maximum number of rounds is reached or the validation metric stabilizes), all parties terminate the procedure and output the final global model.

3.2 The Modified BLS Multi-signature Scheme

The modified BLS multi-signature scheme [19-20] works over a bilinear pairing $e : G_0 \times G_1 \rightarrow G_T$ with generators $g_0 \in G_0$, $g_1 \in G_1$, and two hash functions as random oracles: H_0: $M \rightarrow G_0$ and H_1: $G_1^n \rightarrow R^n$ where R:= $\{1, 2, \ldots, 2^{128}\}$ and $1 \le n \le \tilde{N}$. It consists of key generation, signing, aggregation, and verification as follows.

Key generation: choose a random $\alpha \leftarrow Z_q$ as the secret key sk:= (α) and set $h \leftarrow g_1^{\alpha} \in G_1$; output the public key pk:= (h) .

Signing: given a message $m \in M$, compute the group hash $H0(m) \in G_0$ and output the signature $\sigma \leftarrow H_0(m)^{\alpha} \in G_0$.

Aggregate: given pairs $((pk_1, \sigma_1), \ldots, (pk_n, \sigma_n))$ on the same message m, first compute coefficients $(t_1, \ldots, t_n) \leftarrow H_1(pk_1, \ldots, pk_n) \in R^n$. Output the multi-signature: $\sigma \leftarrow \sigma_1^{t_1} \ldots \sigma_n^{t_n} \in G_0$.

Verification: To verify a multi-signature σ on m with public keys $pk_1, \ldots, pk_n$, recompute$(t_1, \ldots, t_n) \leftarrow H_1(pk_1, \ldots, pk_n)$ and the aggregated public key $apk \leftarrow pk_1^{t_1} \ldots pk_n^{t_n} \in G_1$. Accept if $e(g_1, \sigma) \stackrel{?}{=} e(apk, H_0(m))$.

3.3 Provable Data Possession

PDP is a remote data auditing cryptographic mechanism. Without downloading the entire file, it has a high probability of confirming that the server still holds the undamaged file. The PDP is mainly divided into two parts:

Setup: the client C first generates keys, obtaining the public key (N, g) and the secret key (e, d, v) (where $ed \equiv 1 \pmod{\varphi(N)}$), and fixes the system parameter. The file F to be outsourced is then partitioned into n fixed-size blocks $m_1, \ldots, m_n$. For each index i, the client forms an auxiliary string bound to the secret key, $w_i = v \parallel i$, and generates a tag for block i as $T_{i,m_i} = (h(w_i) \cdot g^{m_i})^d \bmod N$, optionally recording metadata such as block length, offset, and version number. The client assembles the tag set $\Sigma = (T_{1,m_1}, \ldots, T_{n,m_n})$ and uploads (pk, F, Σ) to the server S.

Challenge: to sample c blocks ($1 \le c \le n$), the client picks random seeds k_1, $k_2 \in \{0, 1\}^{\kappa}$ and a random $s \in \mathbb{Z}_N^*$, computes $g_s = g^s \bmod N$, and forms $chal = (c, k_1, k_2, g_s, fid, t)$, which is sent to the server. Using the seeds, the server derives audited indices and coefficients $i_j = \pi_{k_1}(j)$ and $a_j = f_{k_2}(j)$ for $j = 1, \ldots, c$, and aggregates tags and blocks to produce the proof. It returns $\mathcal{V} = (T, \rho, fid, t)$. Upon receipt, the client computes $\tau = T^e \bmod N$, reconstructs i_j, a_j and $w_{ij} = v \| i_j$, and checks:

$$\rho \stackrel{?}{=} H(\tau \cdot j = \prod_{j=1}^{c} [h(w_{i_j})^{-a_j}]^s \bmod N .$$

equality means the audit passes, otherwise it fails.

4 Protocol Description

This section provides a detailed description of the hierarchical mechanism for clients with heterogeneous computational capabilities, as well as the overall training procedure. To facilitate understanding of the proposed mechanism, Table 1 summarizes the notation used throughout this paper.

Table 1. Notations

Notations	Descriptions
C_{id}	Set of clients
D_{id}	Dataset identifier
$T_{i,j}$	Sample label aggregation value
P_X	Probability of detecting missing blocks in a single audit round
P_{total}	Cumulative detection probability
r_{id}	Current training round
$H()$	One-way hash function
θ	Continuous Audit failure threshold
C_{vaild}	Participate in the training client
$I \subseteq [1..n]$	Client data chunk index collection

The system model is illustrated in Figure 1. The server preprocesses the target data and binds each audited file to a unique dataset identifier D_{id} before distributing the corresponding training subset to client i. Clients are divided into high-capability and low-capability groups according to their available computation and energy budget. Malicious clients may delete data blocks, tamper with local data, replay previously valid proofs, or submit forged proofs. The server, or a trusted third-party auditor delegated by the server, verifies the received proofs and returns an audit decision without accessing the raw training data.

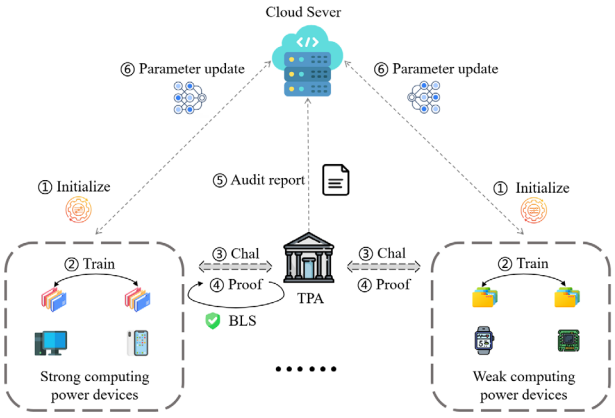


Figure 1. The system model

4.1 Data Auditing for Heterogeneous-Compute Clients

To balance security requirements with resource constraints on end devices, we first stratify clients by computational capability prior to executing Provable Data Possession (PDP) verification, categorizing them into high-capability and low-capability devices. The specific scheme flow is presented in Figure 2 and Figure 3. Accordingly, we apply admission verification strategies with different security strengths and computational overheads. The overall client workflow is as follows: a client first submits a join request; the server performs capability-based stratification and issues a challenge (R); the client generates a proof (P) in response to R and returns it to the server; the server verifies the received P. In addition, for high-capability devices, the server further validates the associated signature to ensure that the client's data remain intact and have not been tampered with. Clients that pass the verification are granted eligibility to participate in federated learning.

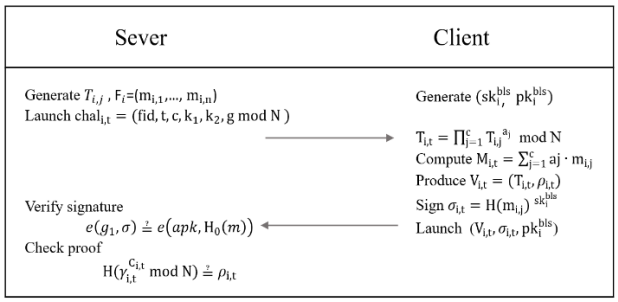


Figure 2. Strong computing client PDP phase

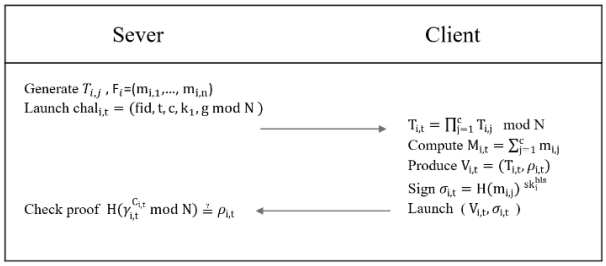


Figure 3. Weak computing client PDP phase

4.1.1 Verification for High-Capability Clients

Step 1: Dataset binding and admission request submission. A high-capability client locally partitions the target dataset D_{id} into blocks according to a unified rule and generates a dataset identifier *dataset id*. Meanwhile, it prepares the S-PDP metadata, so that any subsequent challenge can uniquely reference this dataset and its specific version. The client then submits an admission request to the server, carrying (C_{id}, D_{id}) and the necessary public parameters. Based on these, the server registers and binds the client identity to the dataset, providing traceable evidence for subsequent auditing.

Step 2: Server-side S-PDP challenge generation and issuance. The server generates a random challenge *chal* for the client. The challenge typically consists of a sampled block-index set $I \subseteq [1..n]$ and the corresponding random coefficients $\{v_j\}$, and explicitly includes a round identifier r_{id} to ensure freshness and resist replay attacks. The server sends *chal* to the client and requires it to return the corresponding proof within a specified time window.

Step 3: Client-side generation of a complete S-PDP proof. Upon receiving the challenge, the client follows the S-PDP computation procedure to combine the sampled data blocks and their tags locally, thereby producing a proof. The proof is designed to enable the server to verify that the client indeed possesses the specified dataset without revealing the raw data, while ensuring that missing, tampered, or forged data will, with high probability, lead to verification failure.

Step 4: Proof message construction and BLS signature generation. To ensure non-repudiation and support batch aggregation verification at the server, the client signs the proof-related message using BLS. Specifically, it

constructs a message m that binds the proof to the client identity, dataset, round, time, and challenge: S prevents proof substitution and (C_{id}, D_{id}) prevent cross-dataset misuse and identity impersonation. The client then generates a signature using its private key (sk_i) and submits the proof and signature to the server.

Step 5: Server-side admission verification and decision. After receiving (π_S, σ_i), the server first verifies the correctness of the S-PDP proof and its consistency with the issued challenge, and then validates the BLS signature using the client's public key (pk_i). This ensures that the proof was generated by the claimed client for the given challenge in the specified round and has not been tampered with. Only if both checks succeed does the server grant the high-capability client eligibility to participate in federated learning; otherwise, it rejects the admission request and records the failure status for subsequent risk control and auditing.

4.1.2 Verification for Low-Capability Clients

Step 1: Lightweight preprocessing and admission request submission. Due to limited computation and energy budgets, low-capability clients are required to perform only the lightweight preprocessing and commitment generation needed by E-PDP, thereby minimizing on-device storage and computation overhead. The client submits an admission request to the server and registers (C_{id}, D_{id}), and the minimal public information required by E-PDP, enabling the server to consistently reference the same dataset and the same client across multiple challenge rounds.

Step 2: Server issues small-sample challenges and configures a multi-round admission window. To accommodate resource-constrained devices, the server does not employ large-scale sampling during admission. Instead, in each round it generates a small-sized challenge $chal_t$ with the number of sampled blocks set to k, and attaches the round identifier $rid \triangleq t$ to prevent replay attacks. For each low-capability client, the server configures an admission window of R_0 verification rounds, leveraging multiple small-sample challenges to accumulate coverage and increase the probability of detecting missing or tampered data. In the configuration used in our analysis, k = 230 and R_0 = 2 are sufficient to achieve an overall detection probability close to 99% under the assumed corruption model.

Step 3: Low-capability client generates an E-PDP proof. In each round, upon receiving $chal_t$, the client efficiently computes the corresponding E-PDP proof and returns it to the server. To further reduce device-side overhead, the client performs no signature operation on the proof in this scheme, avoiding the additional computation and energy consumption introduced by pairing-friendly curve signatures, and thus ensuring sustainability for long-term participation in federated learning.

Step 4: Server verifies each round and accumulates pass outcomes. The server verifies the returned proof in every round and records whether the verification succeeds. Since a single small-sample challenge provides limited coverage, the server aggregates the outcomes across rounds and maintains the cumulative number of passes, denoted as *pas*, as an indicator of the client's overall trustworthiness during the admission phase.

Step 5: Multi-round admission decision and feedback. After completing R challenge rounds, the server grants or denies admission according to an accumulated decision rule. If the client meets the consecutive-pass threshold, it is deemed with high probability to possess the target dataset and is allowed to participate in training; otherwise, admission is rejected, and the client may be required to repeat the admission verification in subsequent requests. By combining small-sample auditing and multiple rounds, this strategy substantially enhances detection capability while keeping device-side overhead controllable, achieving a practical balance among low-capability client participation, system scalability, and acceptable security.

4.2 BLS Signature for Data Proof

Algorithm 1. BLS multi-signature phase for proof construction

Algorithm 1 BLS Multi-Signature

Require: Bilinear groups $(q, G_0, G_1, G_T, e, g_1)$; hashes $H_0 : \mathcal{M} \to G_0$, $H_1 : G_1^m \to R^m$ with $R = \{1, \ldots, 2^{128}\}$; number of signers m; a fixed public-key order $\mathcal{PK} = (pk_1, \ldots, pk_m)$

Ensure: Multi-signature $\sigma \in G_0$ and verification result $\{\mathbf{ACCEPT}, \mathbf{REJECT}\}$

Key Generation:
for each node $i = 1$ to m **do**
  Sample secret key $\alpha_i \leftarrow \mathbb{Z}_q$
  Set public key $pk_i \leftarrow g_1^{\alpha_i} \in G_1$
  Publish pk_i
end for
Aggregation Coefficients & Public Key:
Compute weights $(t_1, \ldots, t_m) \leftarrow H_1(pk_1, \ldots, pk_m) \in R^m$
for each node $i = 1$ to m **do**
  Let $apk_i \leftarrow pk_i^{t_i}$
end for
Compute the aggregate public key $apk \leftarrow \prod_{i=1}^m apk_i \in G_1$
Signature Generation:
for each node $i = 1$ to m **do**
  Compute $h \leftarrow H_0(m) \in G_0$
  Local signature $\sigma_i \leftarrow h^{\alpha_i} \in G_0$
end for
Aggregated Signature:
Using the weights, compute $\sigma \leftarrow \prod_{i=1}^m \sigma_i^{t_i} \in G_0$
Broadcast σ
Signature Verification (each verifier):
Recompute $(t_1, \ldots, t_m) \leftarrow H_1(pk_1, \ldots, pk_m)$
for each node $i = 1$ to m **do**
  Recompute and multiply $apk_i \leftarrow pk_i^{t_i}$ to obtain $apk \leftarrow \prod_{i=1}^m apk_i$
end for
if $e(g_1, \sigma) = e(apk, H_0(m))$ **ACCEPT**; otherwise **REJECT**
Return σ and the verification result

After federated-learning clients generate proofs in response to the server's challenges, they apply the BLS scheme to digitally sign the proofs, as detailed in Algorithm 1. Specifically, each client independently generates a private key and computes the corresponding public key. The client then maps the proof message via a hash function, produces a corresponding signature share, and sends the signature share together with its public key to the server. Once all clients have submitted their public keys and signature shares, the server computes the aggregation coefficients from the list of submitted public keys using a hash function, and derives the aggregated public key as well as the aggregated signature. Finally,

the server verifies the aggregate signature using these two aggregated values and the proof message, thereby ensuring that the proof message originates from legitimate clients and that its content remains correct and untampered.

4.3 Continuous Auditing in Federated Learning

During federated learning, the server selects, in randomly chosen rounds, an effective participation set C_{Valid} for the current round based on the previous auditing outcomes. High-capability clients must pass S-PDP proof verification, whereas low-capability clients are required to pass E-PDP sampling-based proof verification; for any client that fails verification, its training update in the current round is discarded, or it is disallowed from participating in subsequent computation. Each client initializes local training from w_t and performs multiple local epochs on its local dataset D_{id}, obtaining a local model w_{t+1}, which is then uploaded to the server. The server aggregates only the updates that pass verification to produce the new global model w_{t+1}. Under FedAvg, the aggregation can be weighted by the local data sizes:

$$w_{t+1} = \sum_{i \in C_{valid,t}} \frac{D_i}{\sum_{j \in C_{valid,t}} D_j} w_t$$

Meanwhile, the server maintains per client verification failure counter $fail[i]$. If a client's verification failures over consecutive rounds or within a sliding window reach a threshold θ, the server removes it from future training rounds to mitigate the impact of missing data or malicious participants on the global model. The above procedure is repeated in subsequent rounds until the maximum number of rounds is reached or the model converges.

5 Security Analysis

5.1 Correctness Analysis

The correctness of the signature can be verified as follows. First, compute the coefficients $(t_1, ..., t_n) = H_1(pk_1, ..., pk_n) \in R^n$, and derive the aggregated public key $apk = \prod_{i=1}^{n} pk_i^{ti} \in G_1$. Then aggregate the individual signatures into a single signature:

$$\sigma = \prod_{i=1}^{n} \sigma_i^{t_i} \in G_0$$

$$\sigma_i = H_0(m)^{\alpha_i}$$

We can derive the following:

$$e(g_1, \sigma) = e\left(g_1, \prod_{i=1}^{n} \sigma_i^{t_i}\right) = \prod_{i=1}^{n} e(g_1, \sigma_i)^{t_i}$$

$$= \prod_{i=1}^{n} e(g_1, \mathrm{H}_0(m))^{a_i t_i}$$

$$e(apk, \mathrm{H}_0(m)) = e\left(\prod_{i=1}^{n} pk_i^{t_i}, \mathrm{H}_0(m)\right)$$

$$= \prod_{i=1}^{n} e(g_2{}^{a_i}, \mathrm{H}_0(m))^{t_i}$$

If the two sides of the above equation are equal, the proposed signature scheme is deemed correct.

5.2 Probabilistic Framework

The PDP protocol enables a verifier to check whether a server retains a randomly selected subset of blocks from a file F. By relying on sampling rather than full retrieval, the verifier can substantially reduce auditing overhead while still maintaining a high likelihood of catching misbehavior.

Consider a file F consisting of n blocks. Suppose the server S has deleted t blocks. In a single audit, the client C challenges the server on c distinct block indices. Let X denote the discrete random variable representing the number of challenged blocks that fall within the set of deleted blocks. We are interested in the probability that the audit detects deletion, that at least one challenged block has been removed:

$$P_X = \Pr\{X \geq 1\} = 1 - \Pr\{X = 0\}$$

$$= 1 - \frac{n-t}{n} \cdot \frac{n-1-t}{n-1} \cdots \frac{n-c+1-t}{n-c+1}$$

Moreover, for any $i \in \{1, ..., c-1\}$, we have

$$\frac{n-i-t}{n-i} \geq \frac{n-1-t}{n-1},$$

which yields the following bounds:

$$1 - \left(\frac{n-t}{n}\right)^c \leq P_X \leq 1 - \left(\frac{n-c+1-t}{n-c+1}\right)^c$$

Here, P_X quantifies the detection probability: if the server deletes t blocks, then an audit that queries c blocks catches the misbehavior with probability P_X. Importantly, when t constitutes a fixed fraction of the file, the detection probability can be made arbitrarily high using a **constant** number of challenged blocks, independent of n. For instance, when $t = 1\% \cdot n$, choosing $c \approx 460$ achieves detection probabilities of at least 99%.

If we sample only 230 blocks per round for low-capability devices, the probability of hitting a corrupted (bad) block in a single round is approximately 90%. The resulting relationship between the number of rounds and the overall probability of detecting a corrupted block is given by:

$$P_{total} = 1 - (1 - P_X)^{r_{id}}$$

Therefore, only two rounds are required for low-capability devices to achieve an overall detection rate of 99%.

6 Performance Analysis

In this section, we evaluated the efficiency of the plan. Our experiments ran on a Windows 11 server equipped with an AMD Ryzen7 H260 (3.80 GHz) and Radeon

(TM) 780M Graphics, alongside 32 GB of RAM. The cryptographic components were implemented using the Pairing-Based Cryptography (PBC) library. The bilinear pairing was instantiated over a pairing-friendly elliptic curve, and the corresponding cyclic groups G_0, G_1, and G_T were initialized according to the default pairing parameters provided by the library configuration. The hash function $H(\cdot)$ used for message digest computation was instantiated as SHA-256, while the mappings H_0 and H_1 were implemented as hash-to-group procedures supported by the pairing library.

For clarity, we also specify the key experimental parameters used in the plots. The number of participating clients was set to 460, the audit sampling size for weak devices was set to k = 230. In addition, for all figures, the parameters not shown on the x-axis were fixed to the same values throughout the experiments.

6.1 Comparison of Computation Cost

Figure 4 compares the auditing scheme for heterogeneous-compute clients proposed in our paper with a global batch-verification baseline. The results show that, while preserving security, our approach substantially improves auditing efficiency via sampling. For low-capability devices, given their limited computational resources, we adopt a strategy combining linear-operation-based encryption with multi-round, small-sample auditing to ensure that verification remains feasible. Figure 5–Figure 6 report the training performance obtained after the auditing procedure; on both MNIST and CIFAR-10, our method achieves the expected learning outcomes.

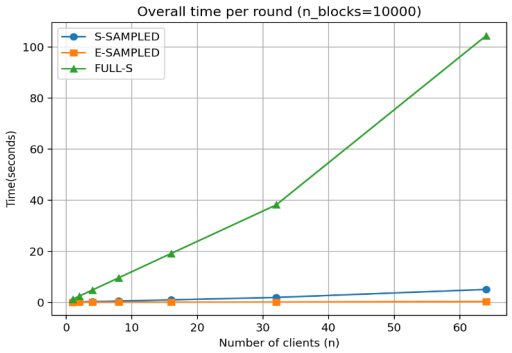


Figure 4. Runtime of different schemes

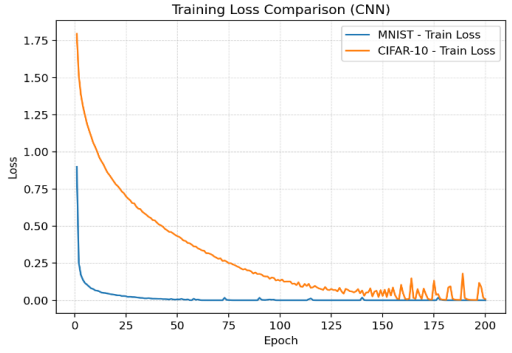


Figure 5. Model training accuracy

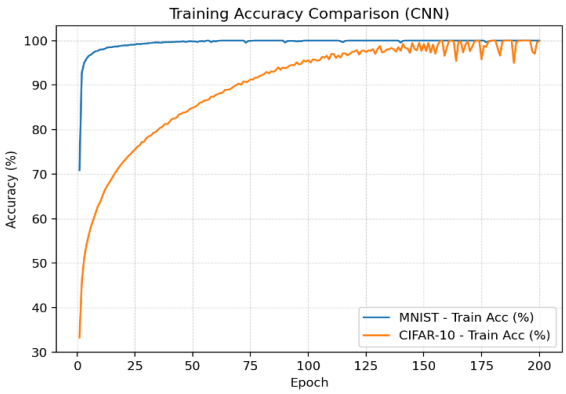
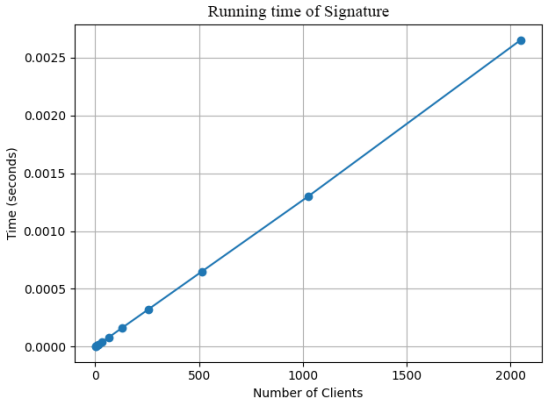


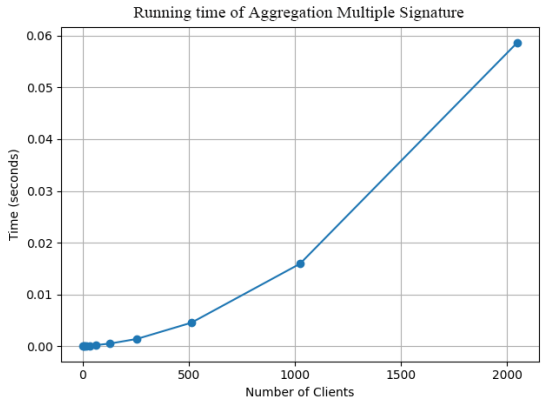
Figure 6. Model training loss value

6.2 Efficiency Assessment

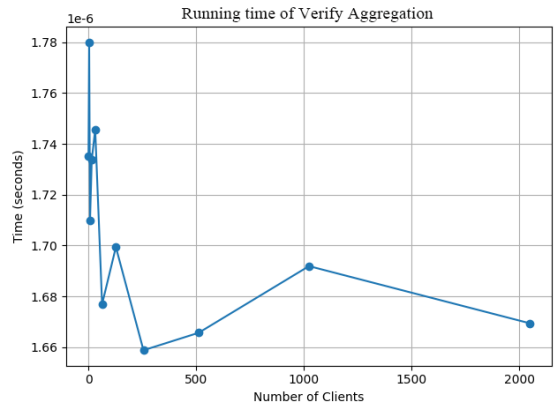
We evaluate the time overhead of employing BLS aggregate signatures to authenticate proofs generated by high-capability clients, as shown in Figure 7, including the detailed time consumption of each component. Figure 7(a) and Figure 7(b) indicate that the time required for signing and aggregating multiple signatures increase as the number of clients increases. Figure 7(c) shows that the time for verification-stage is fluctuat when the client population is small. As the number of clients grows, the verification time becomes increasingly stable. Figure 7(d) shows that the total time cost increases linearly.



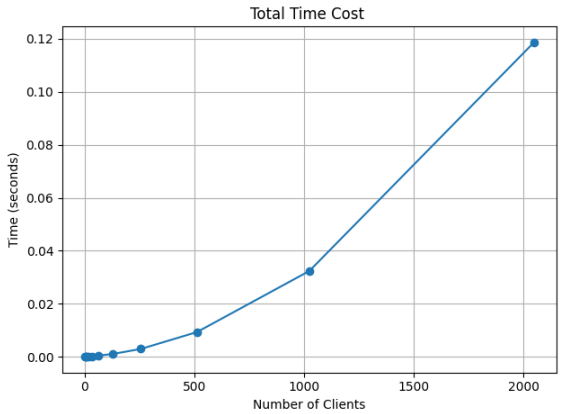
(a)



(b)



(c)



(d)

Figure 7. Runtime of BLS proof-signing

7 Conclusion

A hierarchical PDP-based dataset authentication mechanism tailored to clients with heterogeneous computational capabilities is presented in our paper. By employing differentiated verification strategies, the mechanism can effectively identify clients suffering from data loss or exhibiting malicious behavior. During the process of federated learning, it further performs continuous random-sampling checks on participating clients, enabling efficient identification of malicious clients throughout the training process. Supported by a probabilistic analysis, the proposed design achieves a favorable trade-off between verification efficiency and security guarantees. For proofs submitted by high-capability clients, we incorporate BLS-based aggregate signatures, allowing the server to quickly determine the signature origin and bind it to the corresponding proof content, thereby providing stronger assurance for federated learning computations. In the provable data possession stage, we employ a sampling-based verification approach, substantially reducing the overall verification overhead.

Acknowledgment

The work is supported by the National Natural Science Foundation of China (Nos. 62402444, 62402448, 62441228), the Zhejiang Provincial Natural Science Foundation of China (Nos. LQ24F020012, LQ24F020009), the Program of Zhejiang Key Laboratory of Digital Fashion and Data Governance (No. 2024E10049), the Program for Leading Innovative Research Team of Zhejiang Province (No. 2023R01001), the Fundamental Research Funds of Zhejiang Sci-Tech University under Grants No. 22222266-Y, the "Pioneer" and "Leading Goose" RD Program of Zhejiang (Nos. 2025C02033, 2023C01119).

References

[1] J. Konečný, B. McMahan, D. Ramage, Federated optimization: Distributed optimization beyond the datacenter, *arXiv preprint*, arXiv: 1511.03575, November, 2015.
https://arxiv.org/abs/1511.03575

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-Efficient learning of deep networks from decentralized data, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2017, pp. 1273-1282.

[3] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, *ACM Transactions on Intelligent Systems and Technology*, Vol. 10, No. 2, pp. 1-19, March, 2019.
https://doi.org/10.1145/3298981

[4] D. Ciupek, M. Malawski, T. Pieciak, Federated learning: A new frontier in the exploration of multi-institutional medical imaging data, *arXiv preprint*, arXiv: 2503.20107, March, 2025.
https://arxiv.org/abs/2503.20107

[5] E. Dritsas, M. Trigka, Federated learning for IoT: A survey of techniques, challenges, and applications, *Journal of Sensor and Actuator Networks*, Vol. 14, No. 1, Article No. 9, January, 2025.
https://doi.org/10.3390/jsan14010009

[6] S. Zhang, J. Tay, P. Baiz, The effects of data imbalance under a federated learning approach for credit risk forecasting, *arXiv preprint*, arXiv: 2401.07234, January, 2024.
https://arxiv.org/abs/2401.07234

[7] M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, *Proceedings of the 16th Annual International Cryptology Conference (CRYPTO '96)*, Santa Barbara, CA, USA, 1996, pp. 1-15.

[8] National Institute of Standards and Technology, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication (FIPS) NIST FIPS 186-5, February, 2023.
https://doi.org/10.6028/NIST.FIPS.186-5

[9] D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm, *International Journal of Information Security*, Vol. 1, No. 1, pp. 36-63, August, 2001.
https://doi.org/10.1007/s102070100002

[10] W. Zheng, T. Zhou, Z. A. Bhuiyan, J. Shen, Privacy-Preserving data auditing with efficient corrupted data locating for cloud computing, *Proceedings of IEEE Conference on Computer Communications Workshops*, London, United Kingdom, 2025, pp. 1-6.
https://doi.org/10.1109/INFOCOMWKSHPS65812.2025.11152801

[11] Z. Zhang, Y. Li, NSPFL: A novel secure and privacy-preserving federated learning with data integrity auditing, *IEEE Transactions on Information Forensics and Security*, Vol. 19, pp. 4494-4506, March, 2024. https://doi.org/10.1109/TIFS.2024.3379852

[12] H. Yu, R. Xu, H. Zhang, Z. Yang, H. Liu, EV-FL: Efficient verifiable federated learning with weighted aggregation for industrial IoT networks, *IEEE/ACM Transactions on Networking*, Vol. 32, No. 2, pp. 1723-1737, April, 2024. https://doi.org/10.1109/TNET.2023.3328635

[13] Y. Li, J. Shen, S. Ji, S. Wen, T. Zhu, Y. Xiang, Collusion-Resistant multi-Replica data auditing with optimized metadata generation, *IEEE Transactions on Dependable and Secure Computing*, Vol. 23, No. 1, pp. 417-431, January- February, 2026. https://doi.ieeecomputersociety.org/10.1109/TDSC.2025.3606487

[14] Y. Su, J. Wang, S. H. Tu, K. T. Liao, C. L. Lin, Detecting latent topics and trends in IoT and e-commerce using BERTopic modeling, *Internet of Things*, Vol. 32, Article No. 101604, July, 2025. https://doi.org/10.1016/j.iot.2025.101604

[15] W. Tong, X. Dong, J. Shen, Y. Shen, Z. Dong, Non-Subjective trust mechanism for online ride-hailing services, *IEEE Transactions on Services Computing*, Vol. 19, pp. 530-544, January- February, 2026. https://doi.ieeecomputersociety.org/10.1109/TSC.2025.3630208

[16] D. Liu, Y. Zhang, W. Wang, K. Dev, S. A. Khowaja, Flexible Data Integrity Checking with Original Data Recovery in IoT-Enabled Maritime Transportation Systems, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 24, No. 2, pp. 2618-2629, February, 2023. https://doi.org/10.1109/TITS.2021.3125070

[17] D. Liu, Y. Ding, G. Yu, Z. Zhong, Y. Song, Privacy-preserving Dynamic Auditing for Regenerating Code-Based Storage in Cloud-Fog-Assisted IIoT, *Internet of Things*, Vol. 25, Article No. 101084, April, 2024. https://doi.org/10.1016/j.iot.2024.101084

[18] Y. Liu, X. Hao, W. Ren, R. Xiong, T. Zhu, K. R. Choo, G. Min, A Blockchain-Based decentralized, fair and authenticated information sharing scheme in zero trust internet-of-things, *IEEE Transactions on Computers*, Vol. 72, No. 2, pp. 501-512, February, 2023. https://doi.org/10.1109/TC.2022.3157996

[19] X. Hao, W. Ren, Y. Fei, T. Zhu, K. R. Choo, A Blockchain-Based Cross-Domain and Autonomous Access Control Scheme for Internet of Things, *IEEE Transactions on Services Computing*, Vol. 16, No. 2, pp. 773-786, March-April, 2023. https://doi.org/10.1109/TSC.2022.3179727

[20] Y. Zhao, B. Liu, T. Zhu, M. Ding, X. Yu, W. Zhou, Proactive image manipulation detection via deep semi-fragile watermark, *Neurocomputing*, Vol. 585, Article No. 127593, June, 2024. https://doi.org/10.1016/j.neucom.2024.127593

Biographies

Wenying Zheng received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2022. She is currently working with the School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China. Her research interests include cloud storage security and network security.

Zelin Ni is currently a postgraduate student with the School of Computer Science and Technology (School of Artificial Intelligence), Zhejiang Sci-Tech University, Hangzhou, China. His research interests include computer and network security, federated learning and privacy-preserving artificial intelligence.

Tianqi Zhou received her Ph.D. at Nanjing University of Information Science and Technology in 2023. From 2022 to 2023, she was a visiting Ph.D. student at Kyushu University. She is an Associate Professor at Zhejiang Sci-Tech University. Her research interests include public cryptography and cloud computing.