# SVTAgg: Secure, Verifiable, and Traceable Model Aggregation for Edge-Assisted Federated Learning

Lingshuang Liu, *Student Member, IEEE*, Xiangman Li, *Student Member, IEEE*, Xue Qin, *Student Member, IEEE*, Jianbing Ni, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this paper, we propose SVTAgg, a secure, verifiable, and traceable model aggregation protocol for edge-assisted federated learning (EFL). Unlike conventional federated learning architectures that assume trustworthy aggregation servers, SVTAgg explicitly addresses the privacy and integrity risks posed by untrusted edge servers which are responsible for intermediate aggregation in EFL. Specifically, to preserve the confidentiality of local gradients, SVTAgg combines secret sharing with a pairwise double masking mechanism that prevents information leakage while still supporting efficient model aggregation. To guarantee the correctness of aggregated results, SVTAgg incorporates homomorphic verifiable tags, enabling lightweight verification of intermediate model updates without revealing client data. Furthermore, SVTAgg introduces client traceability by integrating homomorphic signatures with proxy re-signing signatures, providing authenticated provenance of submitted gradients and enabling the identification and attribution of end devices participating in local model training. Security analysis shows that SVTAgg ensures strong gradient privacy, aggregation correctness, and client traceability against realistic adversarial edge settings. Experimental results demonstrate that SVTAgg can achieve similar convergence performance compared with standard EFL, while maintaining low computational and communication overhead.**

***Index Terms*—Edge-assisted federated learning, secure model aggregation, privacy preservation, verifiable aggregation, traceability.**

## I. Introduction

EDGE intelligence [1] leverages edge computing [2] to deploy AI services closer to data sources, offering a scalable solution to the distributed learning demands of massive Internet of Things (IoT) ecosystems and the growing need for real-time data analytics, decision-making, and productivity enhancement. As a key enabler of edge intelligence, Federated Learning (FL) enables collaborative model training directly at end devices without exposing raw data, thereby preserving privacy while supporting distributed AI workloads [3]. Building on this paradigm, Edge-assisted Federated Learning (EFL) [4] integrates FL with multi-tier edge computing, allowing decentralized devices, such as IoT sensors and mobile phones, to train local models and offload intermediate aggregation to nearby edge servers. These edge servers act as intermediary coordinators that efficiently aggregate local updates, mitigate device heterogeneity, reduce the burden on cloud servers, and significantly improve system responsiveness. EFL offers several advantages, including reduced latency and bandwidth overhead, accelerated model training at both device and edge levels, and enhanced support for latency-sensitive applications such as anomaly detection in IoT networks and autonomous driving systems [5].

While EFL preserves data locality by allowing each end device to retain raw data and upload only their local model updates to the edge servers, these updates themselves may expose sensitive information. Despite not containing explicit data samples, gradient vectors and model parameters are known to be vulnerable to inference and model inversion attacks [6], enabling an adversarial observer to reconstruct features or even approximate original inputs. Although local model updates are usually encrypted during model uploading to protect against external eavesdroppers, a curious edge server, who performs intermediate aggregation, can still exploit the uploaded gradients to infer private data characteristics. This concern is amplified in well-generalized deep models such as DenseNet and ResNet [7], where the expressive gradient space has been shown to leak substantial information about the underlying training samples. Consequently, without dedicated protection mechanisms, EFL remains susceptible to serious privacy leakage despite its decentralized design.

Recently, secure model aggregation protocols [8] have been proposed in EFL to protect the privacy of local model updates uploaded from end devices to the edge servers. By encrypting or masking local gradients, these protocols enable edge servers to perform intermediate aggregation without learning any individual participant update, thereby mitigating inference and inversion attacks from curious edge nodes. Although secure aggregation effectively prevents edge servers from extracting sensitive information, it does not ensure that an edge server has executed the aggregation correctly [9]. A dishonest or lazy edge server may skip certain updates, manipulate intermediate results, or return arbitrary aggregated outputs, and such misbehavior would propagate to the cloud-level aggregation without being detected. This integrity issue is particularly significant in EFL, where edge servers play a critical coordination role in

Received 2 March 2026; revised 24 May 2026; accepted 8 June 2026. Date of publication 24 June 2026; date of current version 1 September 2026. *(Corresponding author: Jianbing Ni.)*

Lingshuang Liu, Xue Qin, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: l325liu@uwaterloo.ca; xue.qin@uwaterloo.ca; sshen@uwaterloo.ca).

Xiangman Li and Jianbing Ni are with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: xiangman.li@queensu.ca; jianbing.ni@queensu.ca).

Digital Object Identifier 10.1109/TDSC.2026.3707153

1545-5971 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

model aggregation and their outputs directly influence the global model's convergence and performance. Although prior work has proposed privacy-preserving and verifiable aggregation mechanisms for traditional FL [10], [11], their application to edge-assisted hierarchical aggregation at edge servers remains largely unexplored. Moreover, verifiable secure aggregation introduce additional verification-related information at the edge server. The potential leakage of local gradients through such verifiable tags has not been thoroughly investigated. Consequently, a crucial gap remains in ensuring both privacy and verifiable integrity for model aggregation in EFL systems.

In addition to privacy preservation and aggregation correctness, EFL introduces raises critical concerns related to client traceability and accountability. In EFL, the cloud server lacks direct visibility into individual updates submitted by end devices, making it difficult to accurately assess client contributions. This limited observability complicates contribution auditing, incentive allocation, and the attribution of malicious behavior. The absence of reliable traceability becomes especially problematic in adversarial or incentive-driven environments. Malicious clients may upload poisoned or low-quality updates to degrade global model performance, while free-riding participants may attempt to claim rewards without performing meaningful local training. Although a few of secure aggregation protocols integrate digital signatures to authenticate client identities or contributions, these approaches often leak information about local model gradients, incur additional communication overhead, and even create technical dilemmas in EFL settings. Individual signatures become unverifiable for the cloud server after intermediate aggregation at the edge without compromising privacy. Therefore, designing effective traceability mechanisms in EFL requires carefully balancing gradient privacy, aggregation correctness, and verifiable contribution attribution.

In this paper, we propose SVTAgg, a secure, verifiable, and traceable aggregation protocol tailored for EFL. SVTAgg builds upon the state-of-the-art Secure Aggregation (SecureAgg) protocol by Bonawitz et al. [12], which employs a pairwise double-masking mechanism based on secret sharing to protect the confidentiality of local model gradients. To extend SecureAgg to hierarchical EFL settings, we integrate it with homomorphic signatures and proxy re-signatures, enabling verifiable aggregation across both edge servers and the central cloud server. This design supports scalable and efficient global model construction while preserving gradient privacy in multi-tier architectures. A key benefit of incorporating homomorphic signatures and proxy re-signing signatures is the ability to cryptographically verify the correctness of aggregated gradients, thereby mitigating potential edge-server misbehavior such as selective update dropping, aggregation manipulation, or arbitrary result fabrication. Furthermore, the produced homomorphic verifiable tags provide a mechanism for client traceability, allowing the system to authenticate and identify participating end devices when accountability or dispute resolution is required. However, enabling public verifiability introduces new privacy risks. In particular, verifiable tags may become vulnerable to offline guessing attacks, where an adversary attempts to infer local model gradients by exhaustively testing candidate parameters against publicly verifiable tags. To prevent such gradient leakage, SVTAgg carefully integrates the masking mechanism in SecureAgg to disable public verifiability at the individual update level. Instead, only verifiable tags corresponding to aggregated model updates are verifiable. This design preserves aggregation-level correctness verification and traceability while preventing adversaries from recovering individual local gradients.

The main contributions of this paper can be summarized as follows:

- We propose SVTAgg, the first local model aggregation protocol that simultaneously achieves gradient privacy, aggregation verifiability, and client traceability.
- By integrating homomorphic signatures and proxy re-signatures into SecureAgg, SVTAgg enables secure and verifiable hierarchical aggregation across both edge servers and the cloud, supporting scalable global model construction in multi-tier EFL architectures.
- SVTAgg guarantees the correctness of aggregation performed by potentially untrusted edge servers and supports the traceability of participating end devices, ensuring accountability without compromising the privacy guarantees of EFL.
- SVTAgg prevents offline guessing attacks on verifiable tags by utilizing the masking mechanism in SecureAgg, ensuring that individual local gradients remain confidential while preserving verifiability at the aggregate level.

The remainder of this paper is organized as follows. In Section II, we introduce the state-of-the-art designs of secure model aggregation and verifiable aggregation in federated learning. In Section III, we present the system model, the threat model, and design goals, and in section IV, we introduce the technical primitives used to design SVTAgg. In Section V, our SVTAgg is presented, followed by its security analysis in Section VI. Finally, we show the performance evaluation in Section VII and draw a conclusion in Section VIII.

## II. Related Work

In this section, we review the existing privacy-preserving model aggregation mechanisms in FL and the verifiable model aggregation protocols in FL.

### A. Secure Model Aggregation

Secure model aggregation has become a foundational component of FL, enabling collaborative model training while safeguarding user data. The seminal work by Bonawitz et al. [12] established the first practical secure aggregation protocol, leveraging pairwise masking and additive secret sharing to protect individual model updates even in the presence of client dropouts and honest-but-curious servers. This work catalyzed extensive follow-up research aimed at improving efficiency, scalability, and robustness. Bell et al. [13] optimized the original design to reduce computational overhead, while So et al. [14] proposed Turbo-Aggregate, which employs sparse graph coding to reduce interaction complexity from $O(n^2)$ to $O(nlog\ n)$. Building on these ideas, Kadhe et al. [15] introduced FastSecAgg, an FFT-based multi-secret-sharing construction that substantially

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

enhances scalability. LightSecAgg [16] further reduces communication overhead by eliminating the need to reconstruct individual masks, and FSSA [17] shortens the protocol from four rounds to three under the honest-but-curious adversarial model, yielding additional efficiency gains. In addition, to improve communication efficiency, Fereidooni et al. [18] presented SAFELearn, a generic design for efficient private FL systems that protects against inference attacks. In contrast to previous works, SAFELearn only needs 2 rounds of communication in each training iteration without the dependance of a trusted third party.

Recent work of secure model aggregation has significantly extended classical privacy-preserving designs to address malicious adversaries, decentralization, and robustness in FL. Tang et al. [19] achieved near-optimal corruption and dropout tolerance under malicious models using a non-colluding server–initiator architecture and extractable, equivocable commitments, while Xu et al. [20] proposed TAPFed to defend against inference attacks via threshold functional encryption in multi-aggregator settings. Decentralized and trustworthy aggregation has been explored through architectures such as DeTA [21], which combines parameter-level shuffling with confidential computing, and blockchain- and TEE-based frameworks [22] that ensure aggregation integrity and auditability. Efficiency-oriented designs include multi–private-key homomorphic aggregation without trusted third parties [23], secret-sharing–based fault-tolerant protocols [24], and privacy-preserving aggregation tailored for the Internet of Vehicles (IoV) environments [25]. Beyond confidentiality, recent approaches consider robustness, including PriRoAgg [26] with zero-knowledge proofs for robust aggregation, zero-knowledge–based aggregation integrity under Byzantine attacks [27], and secure aggregation augmented with backdoor detection [28]. In short, these works illustrate the evolution of secure aggregation toward comprehensive security guarantees that jointly address privacy, correctness, robustness, and practical deployment constraints.

Secure model aggregation in EFL has attracted increasing attention due to the presence of untrusted edge servers, heterogeneous resources, and non-IID data distributions. Shen et al. [29] proposed LiPFed, a lightweight privacy-preserving FL framework that employs decentralized secure aggregation for edge networks by integrating blockchain and additive secret sharing, thereby protecting both local and global model privacy while enabling smart-contract–based detection of malicious edge models. Zhao et al. [30] introduced a deep reinforcement learning-based framework that optimizes secure aggregation and resource orchestration in hierarchical FL assisted by untrusted MEC servers, improving efficiency and fairness under system constraints. To cope with non-IID data and Byzantine attacks, He et al. [31] proposed PPBR, a hybrid design combining condensed local differential privacy and packed linearly homomorphic encryption to achieve robust, privacy-preserving aggregation across edge and cloud layers, while tolerating device and edge-server dropouts. Wang et al. [32] developed a secure aggregation framework in semi-asynchronous EFL based on supervised differential privacy and control variables, mitigating privacy leakage and accuracy degradation under client dropouts. More recently, Wang et al. [33] proposed RaSA, a robust and adaptive secure aggregation protocol that jointly addresses non-IID data, system heterogeneity, and malicious edge servers through adaptive weighting and coded computing, further enhancing convergence and resilience in adversarial EFL environments. However, these frameworks primarily concentrate on secure model aggregation in EFL, leaving the issues of verifiable aggregation and client traceability largely unexplored.

### B. *Verifiable Model Aggregation*

Verifiable secure aggregation is crucial for ensuring the integrity and correctness of FL outcomes, especially in the presence of misbehaving servers or unreliable end devices. The representative works include VerifyNet by Xu et al. [11] and VeriFL by Guo et al. [34], which introduce dual-masking mechanisms and homomorphic commitment schemes to verify aggregation correctness without revealing individual model updates. These studies established the foundational design principles for incorporating verifiability into secure aggregation protocols. LightVeriFL [35] further improves efficiency by employing compact homomorphic hash functions and succinct commitment structures, substantially reducing communication and computation overhead while preserving robustness against client dropouts. In addition, Trusted Execution Environments (TEEs) have been explored to enable efficient and practical verifiability. For example, OPSA [36] proposes a one-pass secure aggregation protocol that leverages TEEs to streamline aggregation, while ensuring that verifiability remains enforceable outside the enclave. This design reduces dependence on trusted hardware while maintaining strong integrity guarantees for aggregation results.

More recently, higher-assurance constructions have been proposed to offer stronger correctness and auditability guarantees. Pairing-based protocols utilize bilinear maps to verify gradient correctness, albeit at the cost of significant cryptographic overhead. In contrast, Zhang et al. [37] introduced a zero-knowledge virtual machine (zkVM)-based architecture that combines homomorphic encryption with non-interactive proofs to verify correct local training, achieving both privacy and verifiability. In parallel, blockchain-assisted approaches have gained traction for their auditability and transparency. For instance, PPVFL [38] integrates homomorphic encryption with ECDSA-based digital signatures to support publicly verifiable aggregation, while VerifBFL [39] further leverages zk-SNARKs and incrementally verifiable computation (IVC) on blockchain infrastructures to enable end-to-end auditability across FL rounds. VerifyDFL by Wang et al. [40] advances verifiability in decentralized FL by enabling clients to locally validate the correctness of others' gradients using zero-knowledge input proofs, while preserving privacy. Similarly, WVFL [41] introduces a secure weighted aggregation protocol that mitigates the impact of erroneous local models while allowing devices to verify aggregation correctness, with all intermediate values kept encrypted to ensure privacy. VeSAFL [42] decentralizes model updates across edge nodes, reduces dependency on centralized cloud servers, and fortifies

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

system resilience and reliability by leveraging multi-server aggregators for privacy-preserving FL in edge computing environments. To address robustness against adversarial updates in hierarchical settings, SHIELD [43] proposes a poisoning-resistant aggregation mechanism specifically tailored for hierarchical FL, overcoming the limitations of flat FL robust aggregation methods. Complementing these efforts, PriVeriFL [10] explores the observation that not all model parameters leak privacy, leading to low-expansion homomorphic aggregation schemes that jointly achieve privacy and verifiability, with threshold variants to defend against collusion between aggregators and malicious clients. Efficiency-oriented verifiable designs, such as LightVeriFL [35], further reduce communication and computation overhead using constant-length homomorphic hashes and one-shot recovery under dropouts. Finally, Xie et al. [44] proposed FLVA, a verifiable aggregation framework that avoids bilinear groups through a three-party key negotiation protocol, ensuring privacy-preserving ciphertext aggregation with reduced cryptographic complexity.

However, existing research primarily focuses on verifiability of aggregated models in conventional FL, while verifiability in EFL remains largely underexplored. Moreover, traceability of participating clients has not been systematically investigated, leaving a critical gap in accountability and forensic analysis under adversarial or incentive-driven EFL settings.

## III. Problem Statement

In this section, we first introduce the main architecture of EFL and then describe the threat model and the design goals.

### A. Edge-Assisted Federated Learning (EFL)

The EFL system adopts a hierarchical architecture consisting of three layers: clients, edge servers, and a cloud server. Clients are end devices, such as smart vehicles, unmanned aerial vehicles, and portable medical terminals, that collect data about users and their surrounding environments and perform local model training, optionally assisted by AI training toolkits. The edge servers comprise local servers and edge devices in mobile networks, such as roadside units, micro base stations, and macro base stations, which are deployed in close proximity to the clients. The cloud server $A$ centrally coordinates the training of neural network models with a set of edge servers $\{A_j\}_{j=1}^{L}$. Each edge server $A_j$ further coordinates local training with a randomly selected subset of clients within its coverage area. The clients associated with $A_j$ are denoted by $U_j = \{u_{i,j}\}_{i=1}^{N_j}$, where $N_j$ represents the number of clients. Collectively, all clients $\{U_j\}_{j=1}^{L}$ collaborate to train a shared global neural network model $\Omega$ under the EFL paradigm.

The learning objective of EFL can be formulated as the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{W}^m} \frac{1}{N}\sum_{j=1}^{L}\sum_{i=1}^{N_j}\mathbb{E}_{\xi_{i,j}\sim D_{i,j}}\left[l(\boldsymbol{w},\xi_{i,j})\right], \tag{1}$$

where $N=\sum_{j=1}^{L}N_j$ is the total number of clients, $l(\cdot)$ denotes the local loss function over training samples $\xi_{i,j}$ drawn from client $u_{i,j}$'s local dataset $D_{i,j}$, $\boldsymbol{w}$ represents the global model parameters, and $\mathbb{E}[\cdot]$ denotes expectation.

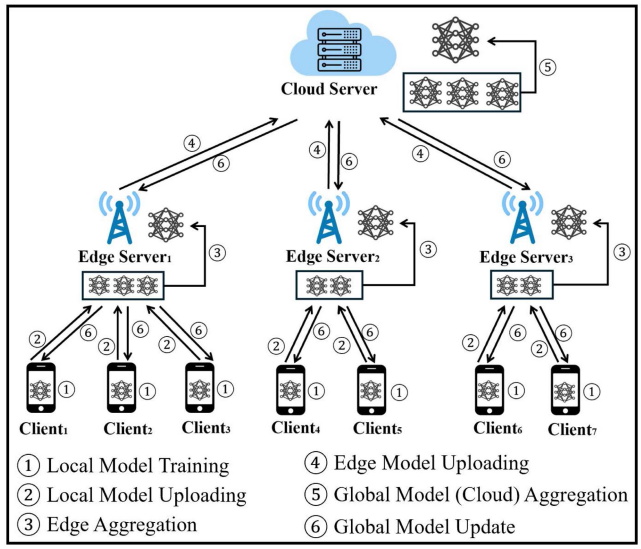



Fig. 1. System architecture of SVTAgg.

An EFL system follows the pipeline described below.

- *FL Task Release:* Upon receiving a FL task, the cloud server $A$ initializes the global model $\boldsymbol{w}_a^{(0)}$.
- *Global Model Download:* During training round $r$, each edge server $A_j$ downloads the current global model $\boldsymbol{w}_a^{(r)}$ from the cloud server $A$ and distributes it to selected clients $u_{i,j}$ within its coverage area.
- *Local Model Training:* Each client $u_{i,j}$ performs local training using stochastic gradient descent (SGD) and updates its local model as

$$\boldsymbol{w}_{i,j}^{(r+1)} = \boldsymbol{w}_a^{(r)} - \eta\nabla l\left(\boldsymbol{w}_a^{(r)}, D_{i,j}\right), \tag{2}$$

  where $\eta$ denotes the local learning rate.
- *Edge Aggregation:* The updated local models are periodically transmitted to the nearby edge server $A_j$, which aggregates the received updates from clients in $U_j$.
- *Cloud Aggregation:* Each edge server $A_j$ forwards its aggregated model update $\boldsymbol{w}_j^{(r+1)}$ to the cloud server $A$. The cloud server then aggregates all edge-level results to obtain the updated global model $\boldsymbol{w}_a^{(r+1)}$.
- *Global Model Update:* The cloud server $A$ disseminates $\boldsymbol{w}_a^{(r+1)}$ to all edge servers $\{A_j\}_{j=1}^{L}$, which subsequently forward it to the selected clients. Clients update their local models accordingly and proceed to the next training round.

This iterative process continues until a stopping criterion is met, typically when the global model converges or reaches a target accuracy. The system architecture of SVTAgg is shown in Fig. 1.

### B. Threat Model

The sensitive information includes each client's training data samples, sample labels, and local model updates in every training round of EFL. The clients are assumed to be

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

semi-honest (honest-but-curious): They follow the prescribed protocol correctly but may attempt to infer private information about other targeted clients, such as the distribution of their training data or even their local models based on messages observed from communication channels, without engaging in active protocol deviations. We assume that clients do not collude with any other entities, including other clients or edge servers.

The cloud server is assumed to be fully trusted and is responsible for initiating EFL tasks for training target models. It coordinates all participating edge servers and oversees the overall model training process. In contrast, edge servers are not assumed to be fully trusted, but this does not imply that they are inherently malicious. Instead, we adopt a potentially adversarial model in which edge servers may behave honestly or deviate arbitrarily from the prescribed protocol, including observing intermediate messages, selectively dropping updates, or manipulating aggregation results. This assumption captures realistic EFL deployments, where edge servers may be dynamically joined and operated by different administrative domains. In addition, a malicious edge server may attempt to infer private information from protocol transcripts across multiple training rounds. We assume that the adversary has full control over corrupted parties but cannot break standard cryptographic primitives (e.g., commitment binding, MAC unforgeability, or zero-knowledge proof soundness) or compromise the trusted execution environments of honest clients.

Finally, due to the inherent mobility of client devices, an arbitrary fraction of semi-honest clients may drop out during protocol execution in EFL. This dynamic environment highlights the importance of designing mechanisms that tolerate client dropouts while still achieving client traceability and accountability in each round of model aggregation, without compromising security or correctness guarantees.

### C. Design Goals

We aim to protect the clients' model updates, prevent the misbehavior of the edge servers, and trace the clients who contribute to the final global model in EFL.

- *Security:* SVTAgg preserves the confidentiality of each client's local model update against semi-honest and malicious entities, including the untrusted edge servers and curious clients. No adversary can learn any information about an individual update beyond what is revealed by the aggregated model.
- *Verifiability:* SVTAgg enables the cloud server to verify the correctness of edge-level aggregations. Any deviation from the prescribed aggregation protocol by a malicious edge server is detected by the cloud server.
- *Traceability:* SVTAgg supports traceability, allowing the identification of clients who contributed to a particular local model. The contribution of each client on the local model training can be recognized.

## IV. Technical Primitives

In this section, we introduce the technical primitives used in the design of SVTAgg, including pairwise marking, secret sharing, and key agreement.

### A. Pairwise Masking

Pairwise masking [12] enables a server to compute the sum of clients' local model updates while preventing the disclosure of any individual update, even in the presence of client dropouts and an honest-but-curious server.

Consider a FL system with a set of clients $\mathcal{U} = \{u_1, \ldots, u_n\}$ and a server that aggregates client updates. Each client $u_i$ holds a local model update vector $\boldsymbol{w}_i \in R^d$. The goal of secure aggregation is to allow the server to learn only the aggregated update $\sum_{i=1}^{n} \boldsymbol{w}_i$, and nothing else about any individual $\boldsymbol{w}_i$.

*1) Pairwise Mask Generation:* For every unordered pair of clients $(u_i, u_j)$ with $i < j$, the two clients establish a shared secret key $k_{i,j}$ using a key agreement protocol (e.g., Diffie–Hellman) as a mask seed. From this shared key, both clients derive a pseudorandom mask vector $\boldsymbol{r}_{i,j} = PRG(k_{i,j})$, where $PRG(\cdot)$ is a secure pseudorandom generator that expands the shared key into a vector of the same dimension as the model update.

*2) Mask Construction:* Each client $u_i$ constructs its total mask by summing all pairwise masks it shares with other clients, assigning opposite signs to ensure cancellation during aggregation:

$$\boldsymbol{m}_i = \sum_{j>i} \boldsymbol{r}_{i,j} - \sum_{j<i} \boldsymbol{r}_{j,i}.$$

This signed construction guarantees that for every pair $(i, j)$, the mask $\boldsymbol{r}_{i,j}$ added by client $u_i$ is exactly canceled by the corresponding subtraction performed by client $u_j$ when all masked updates are summed.

Each client $u_i$ computes its masked update as $\tilde{\boldsymbol{w}}_i = \boldsymbol{w}_i + \boldsymbol{m}_i$, and uploads $\tilde{\boldsymbol{w}}_i$ to the server. Individually, each $\tilde{\boldsymbol{w}}_i$ is information-theoretically hidden by the random mask $\boldsymbol{m}_i$.

*3) Aggregation and Mask Cancellation:* Upon receiving the masked updates from all participating clients, the server computes

$$\sum_{i=1}^{n} \tilde{\boldsymbol{w}}_i = \sum_{i=1}^{n} \boldsymbol{w}_i + \sum_{i=1}^{n} \boldsymbol{m}_i.$$

By construction of the pairwise masks, the aggregate mask term satisfies $\sum_{i=1}^{n} \boldsymbol{m}_i = \boldsymbol{0}$, as every $\boldsymbol{r}_{i,j}$ appears once with a positive sign and once with a negative sign. Consequently, the server obtains exactly the desired aggregated update $\sum_i \boldsymbol{w}_i$ while learning nothing about any individual update.

A key challenge in practical FL systems is client dropout. If a client drops out after masks have been generated but before submitting its masked update, the cancellation property no longer holds. To address this, the protocol incorporates an additive secret sharing mechanism: each client secret-shares its pairwise mask seeds among other clients. In the event of a dropout, the server can reconstruct the missing masks using shares from surviving clients and remove them from the aggregate, thereby restoring correctness without compromising privacy.

However, if the server mistakenly claims a client drops out after the client's masked updates arrive, it can reconstruct the client's mask by using secret shares of mask seeds from other surviving clients.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

*4) Double Mask Construction:* To address this new threat, each client $u_i$ chooses a random value $\gamma_i$ as an additional mask seed and derives its secondary mask $PRG(\gamma_i)$ to double mask its model update. The double masked model update is

$$\tilde{\boldsymbol{w}}_i = \boldsymbol{w}_i + PRG(\gamma_i) + \boldsymbol{m}_i.$$

The client generates and distributes the secret shares of the random mask seed to other clients for the cancellation of the secondary mask during model aggregation.

### B. Secret Sharing

Secret sharing enables a secret to be divided into multiple pieces, called *shares*, which are distributed among a set of participants such that only authorized subsets of participants can reconstruct the secret, while unauthorized subsets learn nothing about it. A widely adopted instantiation is the $(t, n)$-threshold secret sharing scheme. A secret $s \in F$ (where $F$ is a finite field) is shared among $n$ participants in a way that any subset of at least $t+1$ participants can jointly reconstruct the secret $s$, while any subset of at most $t$ participants obtains no information about $s$.

In Shamir's secret sharing scheme, the dealer selects a random polynomial $f(x) = s + a_1 x + a_2 x^2 + \cdots + a_t x^t$, where the coefficients $a_1, \ldots, a_t$ are chosen uniformly at random from $F$. The secret $s$ is embedded as the constant term of the polynomial. Each participant $P_i$ is assigned a public, nonzero identifier $i \in F$ and receives a share $\mathsf{share}_i = f(i)$, so that all shares are $(i, \mathsf{share}_i)_{i \in n} = S.Share(s, t, n)$.

Given any subset of $t+1$ valid shares $\{(i, f(i))\}$, the secret $s = f(0)$ can be reconstructed using Lagrange interpolation:

$$s = S.Recon(\{\mathsf{share}_i\}_{i \in I}, t) = \sum_{i \in I} f(i) \cdot \prod_{\substack{j \in I \\ j \neq i}} \frac{j}{j - i},$$

where $I$ denotes the index set of participating shares. Fewer than $t+1$ shares provide no information about $s$.

### C. Key Agreement

Key agreement enables two or more parties to establish a shared secret key over an insecure communication channel, without prior shared secrets. The established key can subsequently be used to derive cryptographic material, such as encryption keys, message authentication codes, or pseudorandom masks. The most widely deployed key agreement protocol is the Diffie–Hellman (DH) key exchange. Let $(G, q, g)$ be a cyclic group of a prime order $q$ with a generator $g$. Alice selects a random private key $a \in Z_q^*$ and publishes the public key $g^a$. Given Bob's public key $g^b$, both parties can compute the shared secret

$$K = KA.aggree(g^a, g^b) = g^{ab}.$$

The security of Diffie-Hellman relies on the hardness of the Computational Diffie-Hellman (CDH) Problem.

## V. Proposed SVTAgg

In this section, we present the technical intuition and present the detailed constructions of SVTAgg in EFL.

### A. Technical Intuition

EFL adopts a hierarchical architecture in which local model updates are aggregated at both the edge and cloud levels. To achieve the design goals of security, verifiability, and traceability, it is necessary to extend the existing EFL framework with appropriate cryptographic primitives. To protect local model gradients contributed by clients, we leverage pairwise double masking to protect local model parameters before submitting them to the edge server. This technique is both effective and efficient, and naturally supports client dropouts, making it well suited for EFL environments.

Our main novelty lies in extending pairwise double masking to additionally support verification and traceability. To this end, we employ homomorphic signatures, rather than the homomorphic hash commonly used in the literature, which typically provide verifiability only and potentially need extra digital signatures to prevent forgery attacks. The homomorphic verifiable tags generated by homomorphic signatures can be aggregated at the edge servers, enabling verification of aggregated results as well as identification of contributing clients at the cloud server. Although conceptually straightforward, integrating homomorphic signatures into EFL introduces several nontrivial challenges.

First, homomorphic signatures generated by different clients are signed under different keys and over different model parameters, which prevents direct aggregation at the edge level. To address this issue, we integrate homomorphic signatures with proxy re-signatures, allowing an edge server to re-sign local model updates from multiple clients to a common signer, namely the cloud server. As a result, the aggregated re-signatures produced by the edge server become valid verifiable tags on the aggregated local models.

Second, homomorphic signatures are generally publicly verifiable, which improves transparency of verification results but also introduces a security vulnerability. Specifically, a curious edge server may launch offline guessing attacks by exhaustively searching for local model parameters consistent with the publicly verifiable tags, exploiting the relatively small parameter space of model updates. To mitigate this risk, we draw inspiration from designated verifier signatures, where only a specific verifier can validate a signature. However, directly designating the cloud server as the sole verifier would undermine transparency. In SVTAgg, we instead combine this idea with pairwise double masking: homomorphic signatures become verifiable only after re-signing and aggregation, when the random masks added by clients are removed. If a client drops out, the corresponding secret keys reconstructed via secret sharing are required to complete verification, which is consistent with the recovery mechanism used in pairwise double masking to ensure correct edge-level aggregation.

Third, it is critical to ensure that the same local model gradients are used both to generate the masked updates and to produce the corresponding homomorphic verifiable tags. While this requirement is not primarily intended to prevent malicious client behavior, inconsistencies may arise due to client-side operational errors or other unintended factors. Such inconsistencies can lead to verification failures of edge-level aggregation results,

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

thereby undermining the correctness guarantees of the protocol. To prevent verification errors caused by mismatched gradients, it is essential to enable consistency verification without revealing the underlying model parameters to edge servers. To address this challenge, we incorporate zero-knowledge proofs, allowing each client to prove that its masked update and corresponding homomorphic signatures are derived from the same set of model gradients, without disclosing any sensitive information.

Overall, SVTAgg is not a straightforward integration of existing privacy-preserving and secure computation techniques into EFL, but a principled design that addresses the core challenges of hierarchical aggregation in EFL by carefully adapting and extending these techniques. For instance, prior works achieve verifiable aggregation using homomorphic hash techniques; however, such approaches primarily ensure integrity and typically require additional cryptographic primitives (e.g., digital signatures) to provide authenticity and prevent forgery. In contrast, our design leverages homomorphic signatures to construct homomorphic verifiable tags, which inherently guarantee both aggregation correctness and unforgeability under standard cryptographic assumptions. Moreover, these verifiable tags support seamless aggregation across multiple clients and, when combined with proxy re-signing, enable authenticated provenance of model updates. This allows the cloud server to verify aggregated results while attributing contributions to participating clients. Such dual functionality is particularly critical in hierarchical EFL settings, where aggregation correctness and client accountability must be ensured simultaneously without revealing individual model updates.

### B. Detailed SVTAgg

We present the notations of SVTAgg, as shown in Table I and the detailed procedures of SVTAgg. It is bootstrapped by a system initialization phase executed by the cloud server. The cloud server initializes the system parameters based on a security parameter $\kappa$. Specifically, given the security parameter $\kappa$, let $p$ be a large prime and generate a bilinear map $\hat{e} : G \times G \to G_T$, here $G$ and $G_T$ are cyclic groups of the prime order $p$, $g, g_0, g_1$ are generators of $G$, hash functions $H : \{0,1\}^* \to G$ and $H_{\mathsf{FS}} : \{0,1\}^* \to Z_p^*$, and a pseudo-random generator $PRG$. The system parameters include $(p, \hat{e}, G, G_T, g, g_0, g_1, H, H_{\mathsf{FS}}, PRG)$.

SVTAgg operates over multiple training rounds and involves coordinated interactions among the cloud server, edge servers, and clients' end devices. The protocol is conceptually divided into two phases: key preparation and model training.

During the key preparation phase, all entities initialize the required cryptographic materials. Specifically, the cloud server, edge servers, and clients generate their respective public–private key pairs, and edge servers additionally obtain proxy re-signing keys from the cloud server. Meanwhile, clients securely generate the values needed to protect local model gradients during training, including mask seeds used for pairwise masking of local models, and secret shares of clients' private keys and random mask seeds for model unmasking and aggregation. The detailed procedures of the key preparation phase are illustrated in Fig. 2.

TABLE I
NOTATIONS

| Symbol | Description |
|---|---|
| $A$ | Cloud server |
| $A_j$ | Edge server $j$ |
| $u_{i,j}$ | Client $i$ associated with edge server $A_j$ |
| $U_j$ | Set of clients under edge server $A_j$ |
| $N_j$ | Number of clients under $A_j$ |
| $w$ | Global model parameters |
| $w_a^{(r)}$ | Global model at round $r$ |
| $w_{i,j}^{(r)}$ | Local model of client $u_{i,j}$ at round $r$ |
| $x_{i,j}$ | Local model update (gradient) of client $u_{i,j}$ |
| $X_{i,j}$ | Masked model update of client $u_{i,j}$ |
| $z_j$ | Aggregated model update at edge server $A_j$ |
| $Z$ | Global aggregated model at the cloud server |
| $\eta$ | Learning rate |
| $\gamma_{i,j}$ | Random seed for secondary masking |
| $r_{i,j}$ | Pairwise mask for client $u_{i,j}$ |
| $\tilde{w}_i$ | Masked local model update |
| $\sigma_{i,j}$ | Homomorphic signature for client $u_{i,j}$ |
| $\sigma_j$ | Aggregated verifiable tag at edge server $A_j$ |
| $PK_{i,j}$, $SK_{i,j}$ | Public and private keys of client $u_{i,j}$ |
| $RK_{i,j}$ | Proxy re-signing key for client $u_{i,j}$ |
| $e(\cdot,\cdot)$ | Bilinear pairing operation |
| $H(\cdot)$ | Cryptographic hash function |
| $PRG(\cdot)$ | Pseudorandom generator |
| $S.Share$, $S.Recon$ | Secret sharing and reconstruction functions |
| $\pi_{i,j}$ | Zero-knowledge proof generated by client $u_{i,j}$ |
| $t_j$ | Threshold for secret sharing at edge server $A_j$ |

$s \xleftarrow{\$} G$ denotes $s$ is randomly chosen from a group $G$.

In the model training phase, clients' end devices locally train their models and protect the resulting model gradients using pairwise double masking, producing encrypted model updates (model ciphertexts). In parallel, clients generate homomorphic verifiable tags using homomorphic signatures. To ensure consistency between the masked model updates and the corresponding verifiable tags, each client also generates a zero-knowledge proof, attesting that both are derived from the same local model gradients without revealing the gradients themselves. The model ciphertexts, homomorphic verifiable tags, and zero-knowledge proof transcripts are then submitted to the corresponding edge server. Upon receiving these information, the edge server verifies the zero-knowledge proofs, performs edge-level aggregation to obtain aggregated local model parameters, and executes proxy re-signing on the homomorphic verifiable tags using the proxy re-signing key issued for the cloud server, and aggregate the re-signed verifiable tags. This re-signing step ensures that the aggregated verifiable tags can be validated at the cloud level. During edge-level aggregation, clients may be involved in providing their secret shares to reconstruct the private keys of dropped-out clients, as well as the random mask seeds of surviving clients. They are used to mask local model gradients. Subsequently, the cloud server conducts cloud-level aggregation to derive the global model, verifies its correctness using the re-signed homomorphic verifiable tags, and distributes the updated global model to clients via the edge servers for the next training round. The detailed steps of the model training phase are shown in Fig. 3, and the information flow is given in Fig. 4.

SVTAgg supports traceability when necessary, such as in the presence of aggregation inconsistencies, suspected poisoning

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

**Key Preparation of SVTAgg**

- **Round 1 (GenerateKey)**
  *Client* $u_{i,j}$:
  - Generate the private-public key pair $(SK_{i,j}, PK_{i,j})$, $SK_{i,j} = (\alpha_{i,j}, \beta_{i,j})$, and $PK_{i,j} = (g^{\alpha_{i,j}}, g^{\beta_{i,j}})$, where $\alpha_{i,j} \xleftarrow{\$} Z_p^*$, $\beta_{i,j} \xleftarrow{\$} Z_p^*$ and send the public key $PK_{i,j}$ to the edge server $A_j$.

  *Edge Server* $A_j$:
  - Generate the private-public key pair $(s_j, g^{s_j})$, where $s_j \xleftarrow{\$} Z_p^*$.
  - Receive the public keys from at least $t_j$ clients (denote with $N_{j,0}$ this set of clients, suppose $|N_{j,0}| > t_j$), where $t_j$ is the threshold of the Shamir's $t_j$ out of $N_{j,0}$ protocol used for the edge server $A_j$.
  - Forward the received public keys to all clients $u_{i,j} \in N_{j,0}$ and the cloud server.

  *Cloud Server* $A$:
  - Generate the private-public key pair $(b, g^b)$, where $b \xleftarrow{\$} Z_p^*$.
  - For each participating client $u_{i,j}$ in the coverage area of the edge server $A_j$, compute the proxy re-signing key for $u_{i,j}$ as $RK_{i,j} = g^{\beta_{i,j} b}$ and send $RK_{i,j}$ to the edge server $A_j$.
- **Round 2 (ShareKeys)**
  *Client* $u_{i,j}$:
  - Receive the message from the edge server $A_j$,
  - Select a random value $\gamma_{i,j} \xleftarrow{\$} Z_p^*$ as a random mask seed for the secondary masking of its local model gradients and generate the shares of $\gamma_{i,j}$ as $\{m, \gamma_{im,j}\}_{m \in N_{j,0}} = S.Share(\gamma_{i,j}, t_j, |N_{j,0}|)$, where $\gamma_{im,j}$ is the share of the client $u_{i,j}$ to the client $u_{m,j}$.
  - Generate the shares of $\alpha_{i,j} \xleftarrow{\$} Z_p^*$ as $\{(m, \alpha_{im,j})\}_{m \in N_{j,0}} = S.Share(\alpha_{i,j}, t_j, |N_{j,0}|)$, where $\alpha_{im,j}$ is the share of the private key of the client $u_{i,j}$ to the device $u_{m,j}$.
  - Compute $C_{im,j} = Enc(KA.agree(g^{\alpha_{i,j}}, g^{\alpha_{m,j}}), i||m||\gamma_{im,j}||\alpha_{im,j}), m \in N_{j,0}\backslash\{i\}$, where $Enc()$ is a symmetric encryption algorithm with the secret key $KA.agree(g^{\alpha_{i,j}}, g^{\alpha_{m,j}})$. $r_{im,j} = KA.agree(g^{\alpha_{i,j}}, g^{\alpha_{m,j}})$ is a mask seed used for pairwise masking of the client $u_{i,j}$
  - Send $C_{im,j}$ to the edge server $A_j$.

  *Edge Server* $A_j$:
  - Receive the messages from at least $t_j$ clients (denote with $N_{j,1}$ this set of clients, $N_{j,1} \subseteq N_{j,0}$) who generate the secret shares. If the size of $N_{j,1}$ is less than $t_j$, abort.
  - Broadcast $\{C_{im,j}\}_{i \in N_{j,1}}$ to each client $u_{m,j}, m \in N_{j,1}\backslash\{i\}$.
- **Round 3 (ObtainShares)**
  *Client* $u_{i,j}$:
  - Receive $\{C_{mi,j}\}_{m \in N_{j,1}\backslash\{i\}}$ from the edge server $A_j$.
  - Calculate the shared key with every client $m \in N_{j,1}\backslash\{i\}$ as $r_{mi,j} = KA.agree(g^{\alpha_{m,j}}, g^{\alpha_{i,j}})$ and decrypt $\{C_{mi,j}\}_{m \in N_{j,1}\backslash\{i\}}$ with $r_{mi,j}$ to obtain $m||i||\gamma_{mi,j}||\alpha_{mi,j}$ for each $m \in N_{j,1}\backslash\{i\}$.

Fig. 2. Key preparation of SVTAgg.

attacks, or disputes regarding client contributions. Specifically, each client generates a homomorphic signature on its local update, which is transformed via proxy re-signing at the edge server and aggregated into a verifiable tag at the cloud server. Upon verification, the cloud server can authenticate the provenance of the aggregated update with the public key of all clients and identify the set of contributing clients. Importantly, this traceability mechanism does not compromise client privacy: individual model updates remain protected by pairwise double masking and are never revealed during tracing. Instead, the mechanism enables attribution at the contribution level without exposing the underlying gradients, thereby achieving a principled balance between accountability and privacy preservation.

Finally, due to the non-guaranteed reliability of network connections in EFL environments, some clients may drop out during the execution of SVTAgg. To maintain correctness and security, SVTAgg monitors the number of active clients in each round and ensures that it remains above the secret-sharing threshold required by each edge server. Accordingly, edge servers track the number of participating clients in each training round and proceed with aggregation only when the threshold condition is satisfied.

To implement the zero-knowledge proof $PoK_{i,j}$ in a non-interactive way using Fiat-Shamir heuristic.

Prover $P(\mathsf{stmt}_{i,j}; \mathsf{wit}_{i,j})$: Given $\mathsf{wit}_{i,j} = (x_{i,j}, PRG(\gamma_{i,j}), r_{i,j}, \beta_{i,j})$ and $\mathsf{stmt}_{i,j} = (X_{i,j}, Y_{i,j}, \sigma_{i,j})$,

- Compute $h_{i,j} = H(PK_{i,j} \| j \| g^b)$ and $A_{i,j} = h_{i,j}\, g_0^{x_{i,j}}\, g_1^{r_{i,j}} \in G$.
- Sample fresh randomness $a_x, a_m, a_r, a_\beta \xleftarrow{\$} Z_q^*$.
- *Commitments:* Define $T_Y = g_0^{a_x + a_m + a_r}$, $T_A = g_0^{a_x} g_1^{a_r}$, $T_\sigma = A_{i,j}^{a_\beta}$.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

**Model Training of SVTAgg**

- **Round 1 (LocalMask)**
  *Client* $u_{i,j}$:
  - Train a model with the local data to generate its local model gradients $x_{i,j}$.
  - Calculate a mask with the mask seed $r_{im,j}, m \in N_{j,1} \backslash \{i\}$ for pairwise masking of the local gradients as $r_{i,j} = \sum_{i<m} PRG(r_{im,j}) - \sum_{m<i} PRG(r_{mi,j})$ and encrypt the local gradients as $X_{i,j} = x_{i,j} + PRG(\gamma_{i,j}) + r_{i,j}$.
  - Calculate the homomorphic signature $\sigma_{i,j} = (H(PK_{i,j}||j||g^b) g_0^{x_{i,j}} g_1^{r_{i,j}})^{\frac{1}{\beta_{i,j}}}$.
  - Calculate the zero-knowledge proof:

$$PoK_{i,j} = \left\{ \begin{array}{r} (x_{i,j}, PRG(\gamma_{i,j}), r_{i,j}, \beta_{i,j}) : X_{i,j} = x_{i,j} + PRG(\gamma_{i,j}) + r_{i,j} \wedge \\ Y_{i,j} = g_0^{X_{i,j}} = g_0^{x_{i,j} + PRG(\gamma_{i,j}) + r_{i,j}} \wedge \\ \sigma_{i,j} = (H(PK_{i,j}|j||g^b) g_0^{x_{i,j}} g_1^{r_{i,j}})^{\frac{1}{\beta_{i,j}}} \end{array} \right\}.$$

  - Send the message $\Omega_i = (X_{i,j}, \sigma_{i,j}, PoK_{i,j})$ to the edge server $A_j$.

  *Edge Server* $A_j$:
  - Receive the messages from at least $t_j$ clients (denote with $N_{j,2}$ this set of clients); Otherwise, aborts.
  - Send the list of $N_{j,2}$ to the clients in $N_{j,2}$.
- **Round 2 (Aggregate&Unmask)**
  *Client* $u_{i,j}$:
  - Receive from the edge server $A_j$ a list of clients in $N_{j,2}$, verify that $N_{j,2} \subseteq N_{j,1}$ and $|N_{j,2}| > t_j$; Otherwise, abort.
  - Send the secret shares of private keys of dropout clients and random mask seeds of surviving clients to the edge server $A_j$, which consists of $\gamma_{mi,j}$ for clients in $N_{j,2} \setminus \{i\}$ and $\alpha_{mi,j}$ for clients in $N_{j,1} \setminus N_{j,2}$.

  *Edge Server* $A_j$:
  - Collect the responses from at least $t_j$ clients (denote with $N_{j,3}$ this set of clients); Otherwise, aborts.
  - Verify $PoK_{i,j}$ and aborts if $PoK_{i,j}$ is failed.
  - For each $i \in N_{j,1} \setminus N_{j,2}$, calculate $\alpha_{i,j} = S.Recon(\{\alpha_{im,j}\}_{m \in N_{j,3}}, t_j)$, $r_{im,j} = KA.agree(g^{\alpha_{i,j}}, g^{\alpha_{m,j}})$ for $m \in N_{j,2}$, and $r_{i,j} = \sum_{i<m} PRG(r_{im,j}) - \sum_{m<i} PRG(r_{mi,j})$.
  - For each $i \in N_{j,2}$, calculate $\gamma_{i,j} = S.Recon(\{\gamma_{im,j}\}_{m \in N_{j,3}}, t_j)$.
  - Compute $z_j = \sum_{i \in N_{j,2}} x_{i,j}$ as $\sum_{i \in N_{j,2}} x_{i,j} = \sum_{i \in N_{j,2}} X_{i,j} - \sum_{i \in N_{j,2}} \gamma_{i,j} + \sum_{i \in N_{j,1} \setminus N_{j,2}} r_{i,j}$.
  - Compute $\sigma_j = \prod_{i \in N_{j,2}} \hat{e}(\sigma_{i,j}, RK_{i,j})$.
  - Encrypt $D_j = Enc(KA.agree(g^b, g^{s_j}), z_j||\sigma_j||\{r_{i,j}\}_{i \in N_{j,1} \setminus N_{j,2}})$ and send $D_j$ to the central server.

  *Cloud Server* $A$:
  - Receive the aggregated model gradients from all the edge servers.
  - For each edge server $j$, decrypt $D_j$ as $z_j||\sigma_j||\{r_{i,j}\}_{i \in N_{j,1} \setminus N_{j,2}} = Dec(KA.agree(g^b, g^{s_j}), D_j)$, and verify the validity of the aggregated model gradients as $\sigma_j \hat{e}(g_1^{\sum_{i \in N_{j,1} \setminus N_{j,2}} r_{i,j}}, g^b) = \hat{e}(\prod_{i \in N_{j,2}} H(PK_{i,j}|j||g^b) g_0^{z_j}, g^b)$. Aborts if the verification is invalid.
  - Aggregate all the valid aggregated model gradients from the edge servers (denote with $S$ this set of model gradients), as $Z = \sum_{j \in S} z_j$ and returns the global model parameters $Z$ to the edge servers.
- **Round 3 (ModelBackward)**
  *Edge Server* $A_j$:
  - Forward the global model parameters $Z$ from the cloud server to all the clients in $N_{j,0}$.

  *Client* $u_{i,j}$:
  - Initialize the new round of local model training based on the local dataset and the received global model $Z$ from the edge server $A_j$.

Fig. 3. Model training of SVTAgg.

- *Fiat–Shamir challenge:* Compute $c = H_{\mathsf{FS}}(\mathsf{stmt}_{i,j} \,\|\, A_{i,j} \,\|\, T_Y \,\|\, T_A \,\|\, T_\sigma) \in Z_q$.
- *Responses:* Compute

$$s_x := a_x + c\, x_{i,j} \pmod q,$$
$$s_m := a_m + c\, PRG(\gamma_{i,j}) \pmod q,$$
$$s_r := a_r + c\, r_{i,j} \pmod q,$$
$$s_\beta := a_\beta + c\, \beta_{i,j}^{-1} \pmod q.$$

- Output the non-interactive proof $\pi_{i,j} = (A_{i,j},\ T_Y,\ T_A,\ T_\sigma,\ s_x,\ s_m,\ s_r,\ s_\beta)$.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

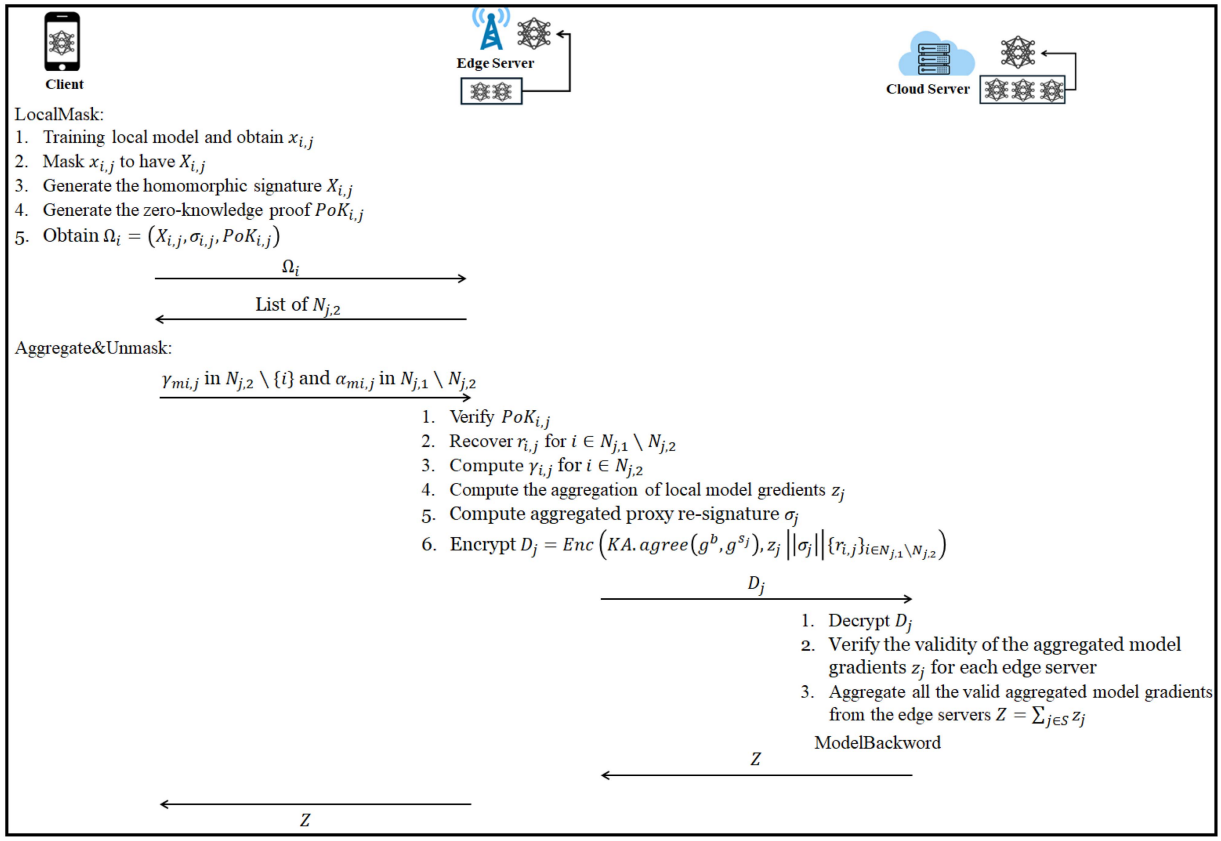



Fig. 4. Information flow of model training in SVTAgg.

Verifier $V(\mathsf{stmt}_{i,j}, \pi_{i,j})$: Given $\pi_{i,j} = (A_{i,j}, T_Y, T_A, T_\sigma, s_x, s_m, s_r, s_\beta)$ and $\mathsf{stmt}_{i,j} = (X_{i,j}, Y_{i,j}, \sigma_{i,j})$,

- Compute $h_{i,j} = H(PK_{i,j} \| j \| g^b)$.
- Recompute the Fiat–Shamir challenge

$$c := H_{\mathsf{FS}}(\mathsf{stmt}_{i,j} \,\|\, A_{i,j} \,\|\, T_Y \,\|\, T_A \,\|\, T_\sigma) \in Z_q.$$

- Verify the following three equations:

$$g_0^{s_x + s_m + s_r} \stackrel{?}{=} T_Y \cdot Y_{i,j}^c$$

$$g_0^{s_x}\, g_1^{s_r} \stackrel{?}{=} T_A \cdot (A_{i,j}/h_{i,j})^c$$

$$A_{i,j}^{s_\beta} \stackrel{?}{=} T_\sigma \cdot \sigma_{i,j}^c.$$

- Accept if all checks pass.

## VI. Security Analysis

In this section, we analyze the security properties of SVTAgg, following the security goals in section III-C, including security, verifiability, and traceability.

• *Security:* The security of STVAgg is to ensure the confidentiality of each client's local model update against both semi-honest and malicious entities. To achieve this, we leverage the pairwise masking mechanism originally proposed in [12], as

$$\sum_{i=1}^{n} \tilde{\boldsymbol{w}}_i = \sum_{i=1}^{n} \boldsymbol{w}_i + \sum_{i=1}^{n} \sum_{j>i} \boldsymbol{r}_{i,j} - \sum_{i=1}^{n} \sum_{j<i} \boldsymbol{r}_{j,i}.$$

The confidentiality of client updates is based on the notion of computational indistinguishability, where an adversary cannot distinguish the observed protocol outputs from uniformly random values. Neither the edge servers nor the cloud server can infer any client's private update, even in the presence of client dropouts or partial collusion. That is, no information about the local model update $x_{i,j}$ is leaked from the masked value $X_{i,j}$, as well as the output $Y_{i,j}$.

Compared to the original secure aggregation protocol in [12], the only additional output that could potentially leak information about the client's local update is the homomorphic signature $\sigma_{i,j}$. Therefore, it is necessary to prevent any leakage of $x_{i,j}$ from $\sigma_{i,j}$. To defend against offline guessing attacks, a known vulnerability of publicly verifiable signatures, we incorporate a random blinding factor $r_{i,j}$ into the verifiable tag generation. When $r_{i,j}$ is sampled uniformly and independently for each client, the resulting verifiable tag $\sigma_{i,j}$ is computationally indistinguishable from a random group element, thereby revealing no information about $x_{i,j}$. The uniform randomness and independence of $r_{i,j}$ have been rigorously established in Hyb5–Hyb9 of the proof of Theorem 6.3 in [12], and we omit the repetition here.

Since $g_1^{r_{i,j}}$ is never revealed, an adversary cannot remove the blinding factor from $\sigma_{i,j}$. As a result, $x_{i,j}$ cannot be inferred from $\sigma_{i,j}$, and the confidentiality of the client's local model update remains preserved.

• *Verifiability:* The verifiability is achieved based on the correctness of homomorphic signature $\sigma_j$, which is aggregated from $\sigma_{i,j}$ after proxy re-signing. If there is no client drop-out, $N_{j,1} =$

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

$N_{j,2}$, so that $\sum_{i\in N_{j,2}} r_{i,j} = \sum_{i\in N_{j,2}}(\sum_{i\in N_{j,2},m\in N_{j,2}\backslash\{i\},i<m} PRG(r_{im,j}) - \sum_{i\in N_{j,2},m\in N_{j,2}\backslash\{i\},m<i} PRG(r_{mi,j})) = 0$, so that $z_j = \sum_{i\in N_{j,2}} X_{i,j} - \sum_{i\in N_{j,2}} PRG(\gamma_{i,j})$, and $\sigma_j = \prod_{i\in N_{j,2}} \hat{e}((H(PK_{i,j}||j||g^b)g_0^{x_{i,j}})^{\frac{1}{\beta_{i,j}}}, RK_{i,j})$. $\sigma_j$ could be verified with the input of the aggregated local model parameters and the public key of the cloud server $A$. The correctness of $\sigma_j$ is verified in the (3).

$$
\begin{aligned}
&\sigma_j \\
&= \hat{e}\left(\prod_{i\in N_{j,2}} H(PK_{i,j}||j||g^b)g_0^{z_j}, g^b\right) \\
&= \hat{e}\left(\left(\prod_{i\in N_{j,2}} H(PK_{i,j}||j||g^b)g_0^{\sum_{i\in N_{j,2}} x_{i,j}} g_1^{\sum_{i\in N_{j,2}} r_{i,j}}\right)^{\frac{1}{\beta_{i,j}}},\right. \\
&\quad \left. g^{\beta_{i,j}b}\right) \\
&= \prod_{i\in N_{j,2}} \hat{e}\left(\left(H\left(PK_{i,j}||j||g^b\right) g_0^{x_{i,j}} g_1^{r_{i,j}}\right)^{\frac{1}{\beta_{i,j}}}, RK_{i,j}\right) \\
&= \prod_{i\in N_{j,2}} \hat{e}(\sigma_{i,j}, RK_{i,j}). \qquad (3)
\end{aligned}
$$

If there are some clients drop-out, $N_{j,1} \neq N_{j,2}$, the central server $A$ needs to utilize the recovered $r_{i,j}, i \in N_{j,1}\backslash N_{j,2}$ to conduct the verification. The correctness of $\sigma_j$ is verified in the (4), shown at the bottom of the this page.

• *Traceability:*To achieve traceability, SVTAgg must guarantee the integrity and authenticity of clients' local model updates. Specifically, each client generates a homomorphic signature $\sigma_{i,j}$ to authenticate its local update, while the edge server aggregates these signatures into $\sigma_j$ to enable verification by the cloud server. Hence, the integrity of local updates relies on the unforgeability of both the individual homomorphic signatures and the aggregated verifiable tag. We prove the unforgeability of the individual homomorphic signatures and aggregated verifiable tag separately. We note that $r_{i,j}$ is sampled uniformly and independently for each client, which has been rigorously established in Hyb5–Hyb9 of the proof of Theorem 6.3 in [12].

*a) Unforgeability of Individual Homomorphic Signature:* Client $u_{i,j}$ generates a homomorphic signature using its private key as $\sigma_{i,j} = (H(PK_{i,j}\|j\|g^b)\, g_0^{x_{i,j}} g_1^{r_{i,j}})^{\frac{1}{\beta_{i,j}}}$. The unforgeability of this signature can be reduced to the Diffie–Hellman Inversion (DHI) assumption [45]. That is, given $h, h^s \in G$ for unknown $s \in Z_p$, no probabilistic polynomial-time (PPT) adversary can compute $h^{1/s}$ with non-negligible probability.

*Theorem 1:* The homomorphic signature on the local model updates is existentially unforgeable against adaptive chosen message attacks under the security model [46], provided that the DHI problem in $G$ is difficult to be addressed with a non-negligible probability in PPT.

*Proof:* Assume there exists a PPT adversary $\mathcal{A}$ that can break the existential unforgeability of the homomorphic signatures with non-negligible probability. We construct a simulator $\mathcal{B}$ that solves the DHI problem with non-negligible probability, thereby contradicting the hardness assumption.

Let $h$ be a generator of the cyclic group $G$ of prime order $p$. The simulator $\mathcal{B}$ is given a DHI challenge $(h, h^s)$ for an unknown $s \in Z_p$, and its goal is to compute $h^{1/s}$. The simulator $\mathcal{B}$ plays the role of the challenger and interacts with $\mathcal{A}$ as follows.

- $\mathcal{B}$ selects random values $r, r_1, r_2 \leftarrow Z_p$ and sets the public key as $PK = h^{s/r}$, the public parameters as $g_0 = h^{sr_1}$ and $g_1 = h^{sr_2}$. These parameters are forwarded to $\mathcal{A}$.
- $\mathcal{B}$ programs the random oracle $H(\cdot)$ and maintains a list to ensure consistency of responses. Upon receiving a hash query $(PK_{i,j}, j, g^b)$ from $\mathcal{A}$, $\mathcal{B}$ flips a biased coin $\theta_{i,j} \in \{0,1\}$ and samples $\phi_{i,j} \leftarrow Z_p$.
  - If $\theta_{i,j} = 0$, $\mathcal{B}$ sets $H(PK_{i,j}\|j\|g^b) = h^{\phi_{i,j}}$.
  - Otherwise, if $\theta_{i,j} = 1$, $\mathcal{B}$ sets $H(PK_{i,j}\|j\|g^b) = h^{s\phi_{i,j}}$.

$$
\begin{aligned}
\sigma_j \hat{e}\left(g_1^{\sum_{i\in N_{j,1}\backslash N_{j,2}} r_{i,j}}, g^b\right) &= \hat{e}\left(\prod_{i\in N_{j,2}} H(PK_{i,j}||j||g^b)g_0^{z_j}, g^b\right) \\
&= \hat{e}\left(\prod_{i\in N_{j,2}} H(PK_{i,j}||j||g^b)g_0^{z_j} g_1^{\sum_{i\in N_{j,1}} r_{i,j}}, g^b\right) \\
&= \hat{e}\left(\left(\prod_{i\in N_{j,2}} H(PK_{i,j}||j||g^b)g_0^{\sum_{i\in N_{j,2}} x_{i,j}} g_1^{\sum_{i\in N_{j,2}} r_{i,j}}\right)^{\frac{1}{\beta_{i,j}}}, g^{\beta_{i,j}b}\right) \hat{e}\left(g_1^{\sum_{i\in N_{j,1}\backslash N_{j,2}} r_{i,j}}, g^b\right) \\
&= \prod_{i\in N_{j,2}} \hat{e}\left((H(PK_{i,j}||j||g^b)g_0^{x_{i,j}} g_1^{r_{i,j}})^{\frac{1}{\beta_{i,j}}}, RK_{i,j}\right) \hat{e}\left(g_1^{\sum_{i\in N_{j,1}\backslash N_{j,2}} r_{i,j}}, g^b\right) \\
&= \prod_{i\in N_{j,2}} \hat{e}(\sigma_{i,j}, RK_{i,j}) \hat{e}\left(g_1^{\sum_{i\in N_{j,1}\backslash N_{j,2}} r_{i,j}}, g^b\right). \qquad (4)
\end{aligned}
$$

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

At last, $\mathcal{B}$ adds a tuple $(PK_{i,j}, j, g^b, \theta_{i,j}, \phi_{i,j}, h_{i,j})$ to the list, and returns $h_{i,j}$ to $\mathcal{A}$.

- When $\mathcal{A}$ queries the signing oracle on $(PK_{i,j}, j, g^b, x_{i,j}, r_{i,j})$, $\mathcal{B}$ first checks whether the corresponding hash query exists. If $(PK_{i,j}, j, g^b)$ has not been queried, $\mathcal{B}$ generates the corresponding $(\theta_{i,j}, \phi_{i,j}, h_{i,j})$ for $(PK_{i,j}, j, g^b)$.
  - If $\theta_{i,j} = 0$, $\mathcal{B}$ aborts and returns failure.
  - If $\theta_{i,j} = 1$, $\mathcal{B}$ computes $\sigma_{i,j} = h^{\phi_{i,j} + r_1 x_{i,j} + r_2 r_{i,j}}$.

  Observe that $\sigma_{i,j}$ is a valid signature on $(PK_{i,j} break j, g^b, x_{i,j}, r_{i,j})$ under the public key $h^{\frac{s}{r}}$. Finally, $\mathcal{B}$ returns $\sigma_{i,j}$ and adds $(PK_{i,j}, j, g^b, x_{i,j}, r_{i,j}, \sigma_{i,j})$ to the list.
- Eventually, $\mathcal{A}$ outputs a forgery $(PK_{i,j}, j, g^b, \hat{x}_{i,j}, \hat{r}_{i,j}, \hat{\sigma}_{i,j})$ such that no signing query has been made for $(PK_{i,j}, j, g^b, \hat{x}_{i,j}, \hat{r}_{i,j})$. If the forgery is invalid, $\mathcal{B}$ aborts and returns failure. Otherwise, $\mathcal{B}$ retrieves the corresponding tuple from the hash list.
  - If $\hat{\theta}_{i,j} = 1$, $\mathcal{B}$ aborts and returns failure.
  - If $\hat{\theta}_{i,j} = 0$, then $H(PK_{i,j} \| j \| g^b) = h^{\hat{\phi}_{i,j}}$ and $\hat{\sigma}_{i,j} = h^{\hat{\phi}_{i,j} r/s} \cdot h^{r_1 \hat{x}_{i,j} + r_2 \hat{r}_{i,j}}$. Thus, $\mathcal{B}$ computes

$$h^{1/s} = \left( \hat{\sigma}_{i,j} \cdot h^{-(r_1 \hat{x}_{i,j} + r_2 \hat{r}_{i,j})} \right)^{1/\hat{\phi}_{i,j}},$$

  solving the DHI challenge.

Therefore, if the DHI problem is hard, no PPT adversary can forge valid homomorphic signatures on clients' local model updates with non-negligible probability, completing the proof.

*b) Unforgeability of Aggregated Verifiable Tag:* We show that it is computationally infeasible to forge a valid aggregated edge-level model–signature pair $(z_j, \sigma_j)$ in PPT under the Conference Key-Sharing (CONF) assumption [47] in the group $G_T$. Specifically, this means that there exists no PPT algorithm that can solve the CONF problem, i.e., given $g, g^s, g^{sv} \in G$, where $s, v \in Z_p$, to compute $\hat{e}(g,g)^v \in G_T$.

*Theorem 2:* The probability that any PPT adversary can generate a valid aggregated verifiable tag $\sigma_j$ that is not equal to the aggregation of the individual homomorphic signatures is negligible, provided that the CONF problem is computationally hard.

*Proof:* Suppose there exists a PPT adversary $\mathcal{A}$ that can break the unforgeability of the aggregated verifiable tag with non-negligible probability. Then, we construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to solve the CONF problem.

Let $g$ be a generator of $G$. Algorithm $\mathcal{B}$ is given $g, D = g^s, D_1 = g^{sv}$, where $s, v \in Z_p$, and its goal is to compute $D_2 = \hat{e}(g,g)^v$. Algorithm $\mathcal{B}$ simulates the challenger and is allowed to access a signing oracle $\mathcal{SO}$ that outputs valid signatures on individual local model updates. It interacts with $\mathcal{A}$ as follows.

- $\mathcal{B}$ randomly selects $\rho_{i,j} \in Z_p$ and sets the public key $PK_{i,j} = D^{\rho_{i,j}}$ and the re-signing key $RK_{i,j} = D_1^{\rho_{i,j}}$ for all $1 \le i \le N_{j,2}$ and $1 \le j \le L$, where $L$ denotes the number of edge servers. $\mathcal{B}$ further selects random values $\gamma_0, \gamma_1 \in Z_p$ and sets $g_0 = g^{\gamma_0}$ and $g_1 = g^{\gamma_1}$. Finally, $\mathcal{B}$ sends $(\{PK_{i,j}, RK_{i,j}\}_{1 \le i \le N_{j,2}}, g, g_0, g_1)$ to $\mathcal{A}$.
- $\mathcal{A}$ may adaptively request signatures on local model updates under any public key in $\{PK_{i,j}\}$. For each query, $\mathcal{B}$ forwards the request to the signing oracle $\mathcal{SO}$, receives the signature $\sigma_{i,j}$, and returns it to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs an aggregated verifiable tag $\hat{\sigma}_j$ on an aggregated edge-level model $\hat{z}_j$ computed from $(PK_{i,j}, j, g^b, \hat{x}_{i,j}, \hat{r}_{i,j})$ for all $1 \le i \le N_{j,2}$. If $\hat{\sigma}_j$ is not valid, $\mathcal{B}$ aborts and reports failure. Otherwise, $\hat{\sigma}_j$ satisfies the verification equation:

$$\hat{\sigma}_j \hat{e}\left( g_1^{\sum_{i \in N_{j,1} \setminus N_{j,2}} \hat{r}_{i,j}}, g^v \right) = \hat{e}\left( \prod_{i \in N_{j,2}} h_{i,j} g_0^{\hat{z}_j}, g^v \right),$$

  where $h_{i,j} = H(PK_{i,j} \| j \| g^b)$. Let $\sigma_j$ denote the honestly generated aggregated verifiable tag corresponding to $(PK_{i,j}, j, g^b, x_{i,j}, r_{i,j}, \sigma_{i,j})$ for $1 \le i \le N_{j,2}$, which satisfies:

$$\sigma_j \hat{e}\left( g_1^{\sum_{i \in N_{j,1} \setminus N_{j,2}} r_{i,j}}, g^v \right) = \hat{e}\left( \prod_{i \in N_{j,2}} h_{i,j} g_0^{z_j}, g^v \right).$$

  If $\hat{z}_j = z_j$ and $\hat{r}_{i,j} = r_{i,j}$ for all $i$, then $\hat{\sigma}_j = \sigma_j$. Otherwise, define $\Delta z_j = \hat{z}_j - z_j$ and $\Delta r_{i,j} = \hat{r}_{i,j} - r_{i,j}$, where at least one is nonzero.
  - If $\hat{\sigma}_j \neq \sigma_j$, dividing the two verification equations yields

$$\frac{\hat{\sigma}_j}{\sigma_j} = \hat{e}\left( \prod_{i \in N_{j,2}} g_0^{\Delta z_j} g_1^{-\sum_{i \in N_{j,1} \setminus N_{j,2}} \Delta r_{i,j}}, g^v \right).$$

    Substituting $g_0 = g^{\gamma_0}$ and $g_1 = g^{\gamma_1}$ gives

$$\frac{\hat{\sigma}_j}{\sigma_j} = \hat{e}\left( g^{\sum_{i \in N_{j,2}} \gamma_0 \Delta z_j - \sum_{i \in N_{j,1} \setminus N_{j,2}} \gamma_1 \Delta r_{i,j}}, g^v \right).$$

    Rearranging yields

$$\hat{e}(g,g)^v = \left( \frac{\sigma_j}{\hat{\sigma}_j} \right)^{\sum_{i \in N_{j,2}} \gamma_0 \Delta z_j - \sum_{i \in N_{j,1} \setminus N_{j,2}} \gamma_1 \Delta r_{i,j}},$$

    which solves the CONF problem.
  - Otherwise, the pairing term equals 1, implying

$$D_2 = \hat{e}(g,g)^v = 1^{\sum_{i \in N_{j,2}} \gamma_0 \Delta z_j - \sum_{i \in N_{j,1} \setminus N_{j,2}} \gamma_1 \Delta r_{i,j}},$$

    which again yields a solution to the CONF problem.

Therefore, if the CONF problem cannot be solved with non-negligible probability in probabilistic polynomial time, then no adversary can forge aggregated verifiable tag on edge-level model updates, completing the proof.

*c) Inconsistency Attack Prevention:* The above results guarantee that the integrity and authenticity of each client's local model update are ensured by the unforgeability of the individual homomorphic signatures and the aggregated verifiable tags. Consequently, any client that contributes to the aggregated global model can be reliably identified and traced by the central server through the verification of the re-signed aggregated tags. However, integrity alone is insufficient to guarantee traceability. A client could attempt to evade traceability by using inconsistent

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

values when generating the masked update $X_{i,j}$ and the homomorphic signature $\sigma_{i,j}$, e.g., uploading a benign model update in $X_{i,j}$ while signing a different malicious update in $\sigma_{i,j}$. Such an attack would pass confidentiality protection but undermine verifiability and traceability.

To prevent this inconsistency attack, SVTAgg requires each client to generate a zero-knowledge proof of knowledge $PoK_{i,j}$ demonstrating that the same local model update $x_{i,j}$ is used consistently in both the masked model $X_{i,j}$ and the homomorphic signature $\sigma_{i,j}$, without revealing $x_{i,j}$ itself.

Formally, $PoK_{i,j}$ proves knowledge of $(x_{i,j}, PRG(\gamma_{i,j}), r_{i,j}, \beta_{i,j})$ such that $X_{i,j} = x_{i,j} + PRG(\gamma_{i,j}) + r_{i,j}$, and $\sigma_{i,j} = (H(PK_{i,j}\|j\|g^b)\, g_0^{x_{i,j}} g_1^{r_{i,j}})^{1/\beta_{i,j}}$, thereby cryptographically binding the same $x_{i,j}$ to both the encrypted update and the signature.

- *Soundness* of the zero-knowledge proof ensures that any client that successfully passes verification must have used the same $x_{i,j}$ in both constructions; otherwise, no witness satisfying the equations exists.
- *Zero-knowledge* guarantees that the edge server and the cloud server learn nothing about $x_{i,j}$ beyond the validity of the statement.
- *Completeness* ensures that honest clients are always accepted.

Therefore, the zero-knowledge proof enforces consistency between $X_{i,j}$ and $\sigma_{i,j}$ while preserving confidentiality, closing the final loophole for traceability. As a result, any misbehaving client that injects corrupted updates can be cryptographically identified and held accountable by the central server, completing the security guarantees of SVTAgg.

## VII. Performance Evaluation

In this section, we conduct a comprehensive evaluation of SVTAgg, focusing on three critical dimensions: model accuracy, computational efficiency, and communication overhead.

### A. Experimental Setup

To evaluate the effectiveness and efficiency of SVTAgg, we conduct comprehensive experiments under an EFL setting. All experiments are performed using ResNet-18 on the EMNIST dataset, which consists of handwritten digits and letters. To emulate practical EFL scenarios, the dataset is partitioned following the default EMNIST federated split, resulting in a non-IID data distribution across clients. This configuration reflects heterogeneous user data distributions commonly observed in real-world deployments.

The EFL architecture consists of multiple clients, several edge servers, and a cloud server. Clients perform local model training using SGD and upload masked local gradients along with homomorphic tags. Edge servers conduct secure edge-level aggregation and proxy re-signature operations before forwarding aggregated results to the cloud server for global aggregation and verification.

All experiments are conducted on a server equipped with an Intel Xeon Gold 6242R CPU (3.10 GHz) and an NVIDIA RTX A6000 GPU. The GPU is used exclusively for accelerating local model training, while all cryptographic operations, including pairwise double masking, homomorphic signature generation, zero-knowledge proof verification, and proxy re-signature, are executed on the CPU. We simulate a cloud server that releases training tasks and coordinates all connected edge servers. The connections between the cloud server and edge servers are assumed to be stable, and edge servers remain online throughout training. The number of edge servers varies across different tasks, and each edge server coordinates its associated clients. In experiments, when evaluating model accuracy, per-round computational cost, and scalability, the number of clients per edge server is kept constant. To study the impact of client dropouts, we additionally consider settings where connections between edge servers and clients are unstable, allowing clients to drop out during training. This setting ensures that the reported cryptographic overhead reflects realistic deployment conditions in resource-constrained edge environments.

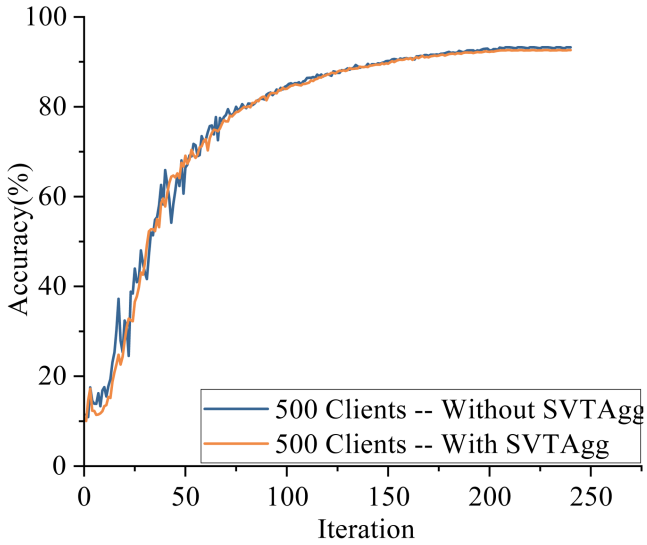



Fig. 5. Model accuracy with SVTAgg and without SVTAgg.

### B. Model Accuracy

Fig. 5 presents the test accuracy comparison between standard EFL and EFL integrated with SVTAgg under 500 participating clients. The number of edge servers are 5 and each edge server coordinates 100 clients. The client drop-out is not considered here. The results show that both schemes exhibit nearly identical convergence trends. The global model accuracy increases steadily during training and stabilizes at approximately 93%. Importantly, the integration of security, verifiability, and traceability mechanisms does not introduce observable degradation in learning performance. The overlap between the "With SVTAgg" and "Without SVTAgg" curves demonstrates that the proposed cryptographic enhancements are orthogonal to the optimization process. This confirms that SVTAgg preserves the statistical efficiency of EFL while strengthening security guarantees.

Fig. 6 further illustrates model accuracy alongside computational cost during training. The accuracy curve increases

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

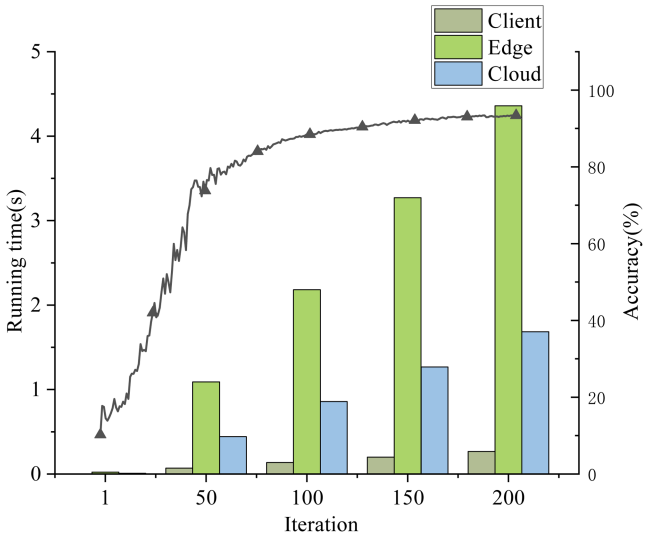



Fig. 6. Model accuracy and computational cost of model training of SVTAgg.

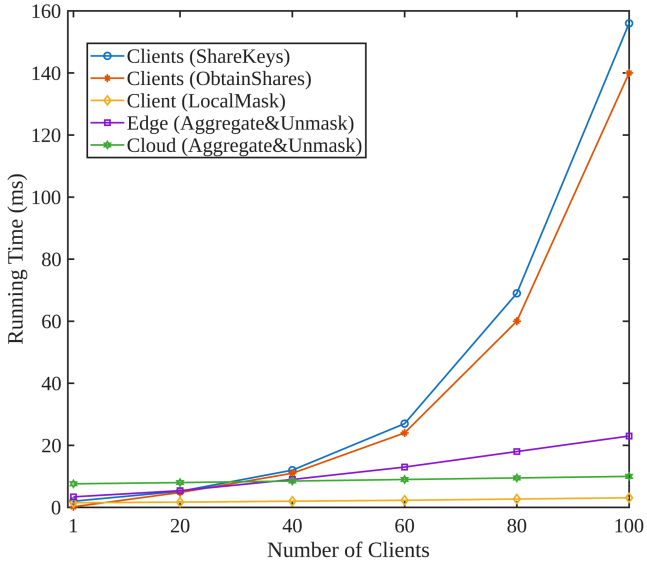



Fig. 7. Computational cost of different parties in SVTAgg.

smoothly as training progresses, indicating stable convergence behavior. No oscillation or instability is observed due to the incorporation of secure aggregation and verification procedures. These results validate that SVTAgg maintains the effectiveness of collaborative model training in EFL systems.

Overall, the experimental results confirm that SVTAgg achieves strong security properties without sacrificing model utility, making it practical for secure EFL deployments.

### C. Computational Overhead Analysis

We evaluate the computational overhead incurred by SVTAgg at the client, edge, and cloud levels.

1) *Round Computational Cost.* Fig. 6 shows the runtime distribution across different entities during model training of SVTAgg. The number of clients is 100, and an edge server coordinates the training process of clients. The client-side cost includes local masking, homomorphic signature generation, and zero-knowledge proof construction. The edge server cost primarily stems from proof verification, aggregation, and proxy re-signature operations. The cloud server cost involves aggregated tag verification and global model aggregation. First, the running time of the client, the edge server, and the cloud server cumulatively increases as the number of training iterations grows, since each round introduces additional cryptographic and aggregation operations. Then, the results for each entity indicate that: 1) Client-side cryptographic overhead remains relatively small compared to local training time. 2) Edge servers incur higher overhead due to aggregation and verification responsibilities. 3) Cloud server overhead remains modest because it processes only aggregated results rather than individual client updates. Despite the additional cryptographic operations, the overall runtime per training round remains within a few seconds, demonstrating the practical feasibility of SVTAgg.

2) *Scalability with Respect to Client Population.* Fig. 7 illustrates the computational cost as the number of clients associated with a single edge server increases. In this setting, up to 100 clients are coordinated by one edge server, which performs intermediate aggregation, while the cloud server verifies the aggregated local models and distributes the validated global model. On the client side, the costs of the ShareKeys and ObtainShares phases grow approximately quadratically with the number of participating clients. This trend is expected due to the pairwise double-masking and secret sharing mechanisms. Specifically, each client must generate secret shares for other clients and establish shared keys with all other clients within the same edge domain. As the client population increases, the number of pairwise interactions rises proportionally, leading to a noticeable increase in computational overhead. However, the operations are simple and the cost is low even there are 100 clients in each edge server. In addition, during the LocalMask (model training) round, each client generates its protected local update, homomorphic signature, and zero-knowledge proof independently. Therefore, this portion of the cost remains essentially unaffected by the number of clients. At the edge server, the Aggregate&Unmask round also scales approximately linearly with the client population. This behavior aligns with the need to verify zero-knowledge proofs, reconstruct secret shares when necessary, and aggregate masked gradients and verifiable tags. Although the edge server bears a heavier workload than individual clients, the growth rate remains moderate, demonstrating good scalability within a single edge domain. At the cloud server, the computational cost remains nearly constant as the number of clients increases. Since the cloud only processes aggregated model updates and aggregated homomorphic verifiable tags from each edge server, its workload is largely independent of the underlying client population. Only a slight increase is observed due to verification of aggregated verifiable tags. Overall, the cryptographic operations introduced by the model training phase of SVTAgg exhibit near-linear scaling rather than quadratic growth with respect to the number of clients. This confirms that the protocol design is efficient and suitable for large-scale EFL deployments.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

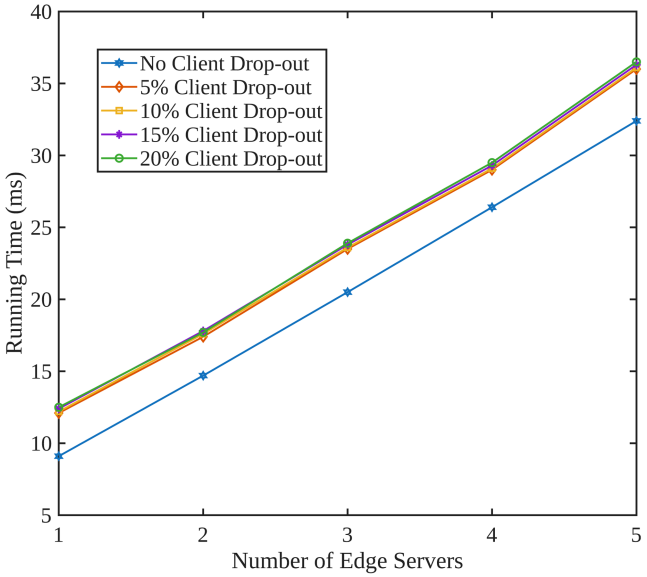



Fig. 8. Computational cost of cloud server with client drop-out.

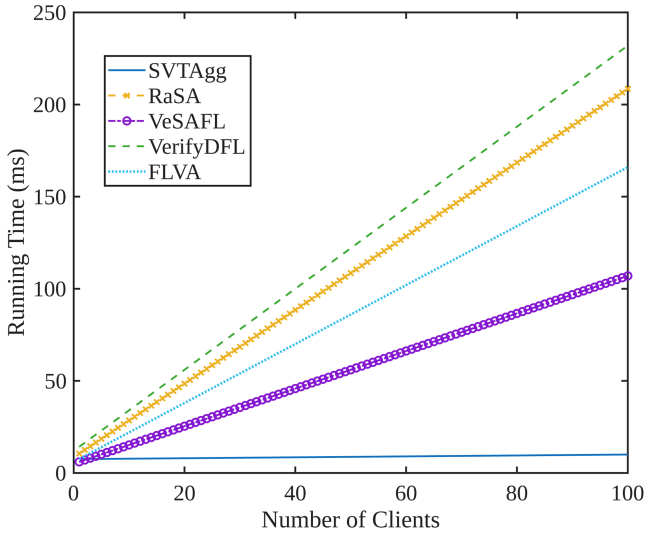



Fig. 9. Comparison on computational cost of cloud server.

3) *Performance Impact of Client Drop-out.* Client dropout affects the verification process at the cloud server, particularly in the validation of aggregated homomorphic verifiable tags. We simulate 1 to 5 edge servers coordinated by the cloud server, with each edge server initially managing 100 clients. When some clients drop out, the edge server must reconstruct and remove the corresponding mask values $r_{i,j}$ for $i \in N_{j,1} \setminus N_{j,2}$. However, this additional overhead is minimal, as the removal operation only involves simple addition operations and does not require expensive cryptographic computations. Therefore, the more noticeable performance impact of client dropout appears at the cloud server during aggregated tag verification. Fig. 8 presents the cloud-side computational cost under varying client dropout rates (0% to 20%) and different numbers of edge servers. The results reveal three key observations:

- The computational overhead increases slightly as the dropout rate grows, primarily due to the additional mask compensation terms that must be incorporated into the aggregated tag verification process.
- Even at a 20% dropout rate, the performance degradation remains limited and well within a practical range, demonstrating the robustness of the protocol under dynamic participation.
- The cloud-side runtime scales moderately with the number of edge servers, since the cloud processes only aggregated model updates and aggregated signatures from each edge server, rather than individual client submissions.

Overall, these results confirm that SVTAgg maintains both efficiency and robustness in dynamic EFL environments where client availability may fluctuate across training rounds.

4) *Efficiency on Verification.* Fig. 9 illustrates the running time of the cloud server in SVTAgg and other existing works, including RaSA [33], VeSAFL [42], VerifyDFL [40], and FLVA [44], as the number of participating clients increases from 1 to 100. The computational cost of all schemes grows with the number of clients, since individual local updates must be processed and verified. Each signature/proof in RaSA [33], VeSAFL [42], VerifyDFL [40], and FLVA [44] must be verified individually, whereas SVTAgg aggregates client signatures and performs batch verification. SVTAgg demonstrates a nearly stable growth trend with respect to the number of clients, indicating strong scalability. Compared to RaSA, VeSAFL, VerifyDFL, and FLVA, SVTAgg achieves competitive runtime performance while simultaneously providing security, verifiability, and traceability guarantees. Notably, the runtime increase remains controlled even at 100 clients, demonstrating that the integration of homomorphic signatures and verification procedures does not introduce prohibitive overhead. This confirms that SVTAgg is practical for large-scale EFL deployments.

5) *Cost of Zero-knowledge Proof.* SVTAgg employs zero-knowledge proofs to guarantee consistency between masked gradients and homomorphic signatures. This additional mechanism introduces moderate computational and communication overhead at both the client and edge-server sides. At the client side, each participant generates a proof involving a constant number of group exponentiations and hash evaluations, which scales linearly with the number of training rounds but remains independent of the model dimension. Given that local model training dominates the computational cost in EFL, the overhead of proof generation is relatively small in practice. At the edge server side, verification requires checking a fixed number of pairing-free group equations per client, resulting in a computational cost that scales linearly with the number of participating clients. Although this introduces additional verification latency, it is amortized by the batch aggregation process and does not significantly affect overall system efficiency. In terms of communication, each proof incurs a constant-size overhead proportional to a small number of group elements, which is negligible compared to the transmission of model updates. Therefore, the use of zero-knowledge proofs provides strong consistency guarantees

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

with only modest overhead, making it practical for deployment in realistic EFL environments.

### D. Communication Efficiency

In EFL, communication efficiency is a critical factor due to the hierarchical interaction among clients, edge servers, and the cloud server. In each training round, communication occurs at two primary levels: between clients and their associated edge servers, and between edge servers and the cloud server.

At the client–edge layer, each client in SVTAgg transmits its masked local model update, similar to Bonawitz et al.'s SecureAgg [12]. The dominant communication cost remains the model gradient itself, which is determined by the model architecture. For example, the parameter size of ResNet-18 is about 11.7 M. In addition to the masked model gradient, the homomorphic signature $\sigma_{i,j}$ and the zero-knowledge proof $PoK_{i,j}$. The size of $\sigma_{i,j}$ is 48 bytes for BLS12-381, and the size of $PoK_{i,j}$ is 384 bytes. At the edge–cloud layer, SVTAgg provides a more significant communication advantage compared to existing verifiable aggregation schemes. In prior works such as RaSA [33], VeSAFL [42], VerifyDFL [40], and FLVA [44], each client's signature must be individually forwarded to the cloud server for verification, resulting in communication overhead that scales linearly with the number of participating clients. In contrast, SVTAgg leverages the homomorphic property of signatures and integrates proxy re-signatures to support hierarchical aggregation. The edge server aggregates both the masked gradients and the corresponding verifiable tags before forwarding them to the cloud. Consequently, the cloud only receives a single aggregated model update (e.g., the size is 11.7 M for ResNet-18) and a single aggregated verifiable tag per edge server (the size is 48 bytes). This reduces the communication complexity between the edge and cloud from linear in the number of clients to constant per edge server.

Client dropout does not significantly affect communication overhead in SVTAgg. Although dropout requires mask removal operations, these adjustments primarily involve local noise $r_{i,j}$, whose size is 32 bytes, for each drop-out client. Therefore, the additional communication overhead grows linearly with the number of dropped-out clients; however, it remains small compared to the size of the model parameters.

## VIII. Conclusion

In this paper, we have proposed SVTAgg, a secure, verifiable, and traceable model aggregation protocol for EFL. SVTAgg addresses key privacy and integrity challenges posed by untrusted edge servers in practical EFL deployments. Through the joint use of secret sharing, double masking, homomorphic verification, and signature-based traceability, SVTAgg simultaneously ensures gradient confidentiality, aggregation correctness, and participant accountability. Security analysis demonstrates that SVTAgg provides strong privacy, verifiability, and traceability guarantees under realistic adversarial edge settings, while experimental results show that SVTAgg achieves learning performance comparable to standard EFL with low computation and communication overhead. These results confirm that SVTAgg is both secure and practical for real-world edge learning systems. Overall, SVTAgg establishes a robust aggregation foundation for trustworthy FL at the network edge, enabling secure and privacy-preserving collaboration in emerging applications such as autonomous systems, mobile health, and large-scale IoT analytics. In the future work, we will extend SVTAgg to support fully malicious edge servers and stronger Byzantine-robust aggregation strategies, as well as exploring adaptive verification mechanisms that further reduce overhead under dynamic edge participation.

## References


[1] X. Wang et al., "Empowering edge intelligence: A comprehensive survey on on-device AI models," *ACM Comput. Surv.*, vol. 57, no. 9, pp. 1–39, 2025.

[2] Q. Duan, J. Huang, S. Hu, R. Deng, Z. Lu, and S. Yu, "Combining federated learning and edge computing toward ubiquitous intelligence in 6 g network: Challenges, recent advances, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2892–2950, 2023.

[3] J. Shen, N. Cheng, W. Xu, H. Wang, W. Quan, and X. Shen, "pFedDHPO: A differentiable approach for personalized hyperparameter optimization in federated learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 12, pp. 412–426, 2026.

[4] K. Wang, Q. He, F. Chen, H. Jin, and Y. Yang, "FedEdge: Accelerating edge-assisted federated learning," in *Proc. ACM Web Conf.*, 2023, pp. 2895–2904.

[5] P. Li et al., "Filling the missing: Exploring generative AI for enhanced federated learning over heterogeneous mobile edge devices," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 10001–10015, Oct. 2024.

[6] N.-B. Nguyen, K. Chandrasegaran, M. Abdollahzadeh, and N.-M. Cheung, "Re-thinking model inversion attacks against deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16384–16393.

[7] S. Anari, S. Sadeghi, G. Sheikhi, R. Ranjbarzadeh, and M. Bendechache, "Explainable attention based breast tumor segmentation using a combination of UNet, ResNet, DenseNet, and EfficientNet models," *Sci. Rep.*, vol. 15, no. 1, 2025, Art. no. 1027.

[8] M. Mansouri, M. Önen, W. B. Jaballah, and M. Conti, "SoK: Secure aggregation based on cryptographic schemes for federated learning," *Proc. Privacy Enhancing Technol.*, vol. 2023, pp. 140–157, 2023.

[9] S. Zhang, W. Wu, L. Song, and X. Shen, "Efficient model training in edge networks with hierarchical split learning," *IEEE Trans. Mobile Comput.*, vol. 24, no. 10, pp. 10214–10229, Oct. 2025.

[10] L. Wang, M. Polato, A. Brighente, M. Conti, L. Zhang, and L. Xu, "PriVeriFL: Privacy-preserving and aggregation-verifiable federated learning," *IEEE Trans. Serv. Comput.*, vol. 18, no. 2, pp. 998–1011, Mar./Apr. 2025.

[11] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 911–926, 2020.

[12] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[13] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1253–1269.

[14] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 479–489, Mar. 2021.

[15] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning," 2020, *arXiv:2009.11248*.

[16] J. So et al., "LightSecAgg: A lightweight and versatile design for secure aggregation in federated learning," *Proc. Mach. Learn. Syst.*, vol. 4, pp. 694–720, 2022.

[17] F. Luo, S. Al-Kuwari, H. Wang, and X. Yan, "FSSA: Efficient 3-round secure aggregation for privacy-preserving federated learning," 2023, *arXiv:2305.12950*.

[18] H. Fereidooniet al., "SAFELearn: Secure aggregation for private federated learning," in *Proc. IEEE Secur. Privacy Workshops*, 2021, pp. 56–62.


Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.

[19] J. Tang, H. Xu, M. Wang, T. Tang, C. Peng, and H. Liao, "A flexible and scalable malicious secure aggregation protocol for federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 4174–4187, 2024.
[20] R. Xu, B. Li, C. Li, J. B. Joshi, S. Ma, and J. Li, "TAPFed: Threshold secure aggregation for privacy-preserving federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4309–4323, Sept./Oct. 2024.
[21] P.-C. Cheng et al., "DeTA: Minimizing data leaks in federated learning via decentralized and trustworthy aggregation," in *Proc. 19th Eur. Conf. Comput. Syst.*, 2024, pp. 219–235.
[22] A. P. Kalapaaking, I. Khalil, M. S. Rahman, M. Atiquzzaman, X. Yi, and M. Almashor, "Blockchain-based federated learning with secure aggregation in trusted execution environment for Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1703–1714, Feb. 2023.
[23] X. Yang, Z. Liu, X. Tang, R. Lu, and B. Liu, "An efficient and multi-private key secure aggregation scheme for federated learning," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 1998–2011, Sept./Oct. 2024.
[24] J. Song, W. Wang, T. R. Gadekallu, J. Cao, and Y. Liu, "EPPDA: An efficient privacy-preserving data aggregation federated learning scheme," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 3047–3057,Sep./Oct. 2023.
[25] Z. Xu et al., "A privacy-preserving data aggregation protocol for internet of vehicles with federated learning," *IEEE Trans. Intell. Veh.*, vol. 10, no. 1, pp. 217–227, Jan. 2025.
[26] S. Hou, S. Li, T. Jahani-Nezhad, and G. Caire, "PriRoAgg: Achieving robust model aggregation with minimum privacy leakage for federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 5690–5704, 2025.
[27] R. Ma, K. Hwang, M. Li, and Y. Miao, "Trusted model aggregation with zero-knowledge proofs in federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 11, pp. 2284–2296,Nov. 2024.
[28] Z. Zhang, J. Li, S. Yu, and C. Makaya, "SAFELearning: Secure aggregation in federated learning with backdoor detectability," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 3289–3304, 2023.
[29] M. Shen et al., "Secure decentralized aggregation to prevent membership privacy leakage in edge-based federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 3105–3119, May/Jun. 2024.
[30] T. Zhao, F. Li, and L. He, "DRL-based secure aggregation and resource orchestration in MEC-enabled hierarchical federated learning," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 17865–17880, Oct. 2023.
[31] Y. He, J. Zhang, P. Yang, Z. Sun, and X. Shen, "PPBR: Privacy-preserving and byzantine-robust edge-assisted hierarchical federated learning in mobile networks," *IEEE Trans. Mobile Comput.*, vol. 25, no. 2, pp. 1595–1612, Feb. 2026.
[32] S. Wang, H. Xie, Y. Guo, F. Guo, F. Jing, and R. Bie, "Privacy-preserving and efficient model aggregation in edge-assisted federated learning," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2024, pp. 511–521.
[33] L. Wang, M. Huang, Z. Zhang, M. Li, J. Wang, and K. Gai, "RaSA: Robust and adaptive secure aggregation for edge-assisted hierarchical federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 4280–4295, 2025.
[34] X. Guo et al., "VeriFL: Communication-efficient and fast verifiable aggregation for federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1736–1751, 2021.
[35] B. Buyukates, J. So, H. Mahdavifar, and S. Avestimehr, "LightVeriFL: A lightweight and verifiable secure aggregation for federated learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 5, pp. 285–301, 2024.
[36] Z. Guan, Y. Zhao, Z. Wan, and J. Han, "OPSA: Efficient and verifiable one-pass secure aggregation with tee for federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 5, pp. 5321–5334, Sep./Oct. 2025.
[37] B. Zhang, G. Lu, P. Qiu, X. Gui, and Y. Shi, "Advancing federated learning through verifiable computations and homomorphic encryption," *Entropy*, vol. 25, no. 11, 2023, Art. no. 1550.
[38] G. K. Mahato, A. Banerjee, S. K. Chakraborty, and X.-Z. Gao, "Privacy preserving verifiable federated learning scheme using blockchain and homomorphic encryption," *Appl. Soft Comput.*, vol. 167, 2024, Art. no. 112405.
[39] A. A. Bellachia, M. A. Bouchiha, Y. Ghamri-Doudane, and M. Rabah, "VerifBFL: Leveraging zk-SNARKs for a verifiable blockchained federated learning," in *Proc. IEEE Netw. Operations Manage. Symp.*, 2025, pp. 1–9.
[40] S. Wang, Y. Tian, J. Xiong, J. Ma, and Y. Zhang, "VerifyDFL: Secure aggregation for decentralized federated learning with input validation in mobile edge intelligence," *IEEE Trans. Cogn. Commun. Netw.*, vol. 12, pp. 2526–2541, 2026.
[41] Y. Zhong, W. Tan, Z. Xu, S. Chen, J. Weng, and J. Weng, "WVFL: Weighted verifiable secure aggregation in federated learning," *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19926–19936, Jun. 2024.
[42] J. Bouamama, Y. Benkaouz, and M. Ouzzif, "VeSAFL: Verifiable secure aggregation for privacy-preserving federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 6, pp. 6592–6609, Nov./Dec. 2025.
[43] Y. Siriwardhana, P. Porambage, M. Liyanage, S. Marchal, and M. Yliantila, "Shield-secure aggregation against poisoning in hierarchical federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 2, pp. 1845–1863, Mar./Apr. 2025.
[44] H. Xie, Y. Wang, Y. Ding, C. Yang, H. Zheng, and B. Qin, "Verifiable federated learning with privacy-preserving data aggregation for consumer electronics," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2696–2707,Feb. 2024.
[45] B. Libert and J.-J. Quisquater, "Improved signcryption from q-diffie-hellman problems," in *Proc. Int. Conf. Secur. Commun. Netw.*, 2004, pp. 220–234.
[46] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.
[47] C.-H. Li and J. Pieprzyk, "Conference key agreement from secret sharing," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 1999, pp. 64–76.



**Lingshuang Liu** (Student Member, IEEE) received the BEng and MS degrees from the School of Computer Science and Engineering, University of Electronic Science and Technology of China in 2014 and 2017, respectively. She is working toward the PhD degree with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include security and privacy in communication networks and artificial intelligence.



**Xiangman Li** (Student Member, IEEE) received the BS degree from the Department of Electrical and Computer Engineering, Queen's Univeristy, Canada, in 2022, where she is currently working toward the PhD degree with the Department of Electrical and Computer Engineering. Her research interests include applied cryptography and AI security.



**Xue Qin** (Student Member, IEEE) received the BEng degree from the North University of China, in 2008, and the MASc degree from the University of Windsor, Windsor, ON, Canada, in 2022. She is currently working toward the PhD degree in electrical and computer engineering with the University of Waterloo, Waterloo, ON. Her research interests include mobile edge computing, AI, and network security.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.



**Jianbing Ni** (Senior Member, IEEE) received the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2018. He is currently an associate professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON. His research interests include applied cryptography and network security, focusing on edge computing, artificial intelligence, and blockchain technology.



**Xuemin (Sherman) Shen** (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a university professor emeritus with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include network resource management, wireless network security, Internet of Things, AI for networks, and vehicular networks. Dr. Shen is a registered professional engineer of Ontario, Canada, Engineering Institute of Canada fellow, Canadian Academy of Engineering fellow, Royal Society of Canada fellow, Chinese Academy of Engineering foreign member, international fellow of the Engineering Academy of Japan, and distinguished lecturer of IEEE Vehicular Technology Society and Communications Society. He was the recipient of the "West Lake Friendship Award" from Zhejiang Province in 2023, President's Excellence in Research from the University of Waterloo in 2022, Canadian Award for Telecommunications Research from the Canadian Society of Information Theory in 2021, R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013), Excellent Graduate Supervision Award in 2006 from the University of Waterloo, and Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He was the president of IEEE Communications Society, vice president of Technical & Educational Activities, vice president of Publications, member-at-large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, and member of IEEE Fellow Selection Committee of the ComSoc. He was the editor-in-chief of *IEEE Internet of Things Journal*, *IEEE Network*, and *IET Communications*.

Authorized licensed use limited to: Nanjing University. Downloaded on October 30,2026 at 06:13:54 UTC from IEEE Xplore. Restrictions apply.