

FLSSM: A Federated Learning Storage Security Model with Homomorphic Encryption

Yang Li, Chunhe Xia, Chang Li, Xiaojian Li and Tianbo Wang, *Member, IEEE*,

Abstract—Federated learning based on homomorphic encryption has received widespread attention due to its high security and enhanced protection of user data privacy. However, the characteristics of encrypted computation lead to three challenging problems: “computation-efficiency”, “attack-tracing” and “contribution-assessment”. The first refers to the efficiency of encrypted computation during model aggregation, the second refers to tracing malicious attacks in an encrypted state, and the third refers to the fairness of contribution assessment for local models after encryption. This paper proposes a federated learning storage security model with homomorphic encryption (FLSSM) to protect federated learning model privacy and address the three issues mentioned above. First, we utilize different nodes to aggregate local models in parallel, thereby improving encrypted models’ aggregation efficiency. Second, we introduce trusted supervise nodes to examine local models when the global model is attacked, enabling the tracing of malicious attacks under homomorphic encryption. Finally, we fairly reward local training nodes with encrypted local models based on trusted training time. Experiments on multiple real-world datasets show that our model significantly outperforms baseline models in terms of both efficiency and security metrics.

Index Terms—Federated Learning, Blockchain, Homomorphic Encryption, Secret Sharing

I. INTRODUCTION

Federated Learning (FL) [1], [2] has emerged as a novel machine learning paradigm designed to protect the privacy and security of user data [3]. In traditional centralized machine learning, user data must be uploaded to a central server for model training, which not only increases the risk of data breaches but may also raise user concerns about data privacy [4]. In federated learning, users only need to share model parameters while keeping their data local, effectively avoiding direct data transmission and leakage, thus attracting increasing attention [5].

Yang Li is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: johnli@buaa.edu.cn).

Chunhe Xia is with the Key Laboratory of Beijing Network Technology, Beihang University, Beijing 100191, China, and also with the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing, Guangxi Normal University, Guilin 541004, China. (e-mail: xch@buaa.edu.cn).

Chang Li is with the School of Computer Science and Technology, Zhengzhou University of Light Industry, Zhengzhou 450000, China (e-mail: 3031169424@qq.com).

Xiaojian Li is with the College of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China.

Tianbo Wang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China, and also with the Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

However, federated learning is a double-edged sword. On the positive side, keeping data on local devices significantly reduces the risk of exposure during transmission, alleviating concerns about data privacy security and increasing people’s willingness to participate in model training [6]. On the negative side, malicious attackers may still steal model parameters through man-in-the-middle attacks, inference attacks, and other methods to deduce users’ data distribution or determine whether specific data exists, causing privacy leakage and increasing fears about data security [7]. Therefore, methods that use encryption technologies to protect local model parameter privacy have become increasingly important.

Widely used encryption technologies in federated learning can be divided into homomorphic encryption, secure multi-party computation, and others [8]. Both provide high security but come with high computational complexity, requiring substantial computational resources. Secure multi-party computation requires cooperation between multiple parties and has high communication overhead [9]. Homomorphic encryption allows a single participant to compute on encrypted data locally without interacting with other parties, reducing dependency on other participants [10]. However, federated learning based on homomorphic encryption faces three unavoidable issues: encrypted model aggregation efficiency, encrypted model access control, and encrypted model contribution assessment [11].

Since homomorphic encryption encrypts local models, the aggregation server needs to perform homomorphic computations on these models, placing high demands on the server’s computational capabilities [12]. When numerous local models exist, the requirements for the aggregation server’s computational power and resources increase significantly. Since local models are encrypted before being sent to the aggregation server, which performs homomorphic computations during aggregation, the server cannot verify the encrypted local models. If malicious attackers implement model poisoning or backdoor attacks, model performance may significantly decrease, posing security threats to the global model. Assessing user contributions in federated learning is crucial for sustainable development, as fair incentive mechanisms can encourage active user participation in model training. However, in federated learning with homomorphic encryption, users share encrypted local models, making existing methods based on gradients, performance, or data quality inadequate [13]. Therefore, how to fairly evaluate user contributions without compromising data privacy becomes one of the challenges.

To address these issues, we propose A Federated Learning Storage Security Model with Homomorphic En-

encryption (FLSSM), consisting of three main components: a Homomorphic-encryption-based Aggregation Mechanism (HAM), a Model Access Control Mechanism based on Shamir’s Secret Sharing (MACM), and an Incentive Mechanism based on Trusted Time Intervals (IMTTI). In HAM, different edge nodes parallelly aggregate different slices of local models, improving aggregation efficiency under homomorphic encryption. MACM allows regulatory nodes to review local models under certain conditions, preventing malicious attacks on local models in encrypted states. IMTTI rewards users based on trusted local model training duration, proposing a new trusted model contribution assessment method without leaking model privacy. Our contributions can be summarized as follows.

- 1) We propose the FLSSM model to address the issues of aggregation efficiency, malicious attacks, and reward allocation in federated learning based on homomorphic encryption. Afterward, we design an aggregation algorithm, an access control mechanism, and an incentive mechanism to solve these problems respectively.
- 2) We present a novel aggregation algorithm that improves the aggregation efficiency of federated learning under homomorphic encryption through parallel aggregation, while establishing trusted nodes to review local models and trace malicious attacks.
- 3) We introduce a new incentive mechanism that conducts reliable assessment of model contributions through model training time, achieving fair user incentivization without compromising model privacy.
- 4) Experiments on two real-world datasets demonstrate that our model’s effectiveness.

The rest of this article is organized as follows. We first give a comprehensive review of related works in Section II. Next, we give the preliminaries for this article in Section III, then formalize the problems and present technical details of FLSSM in Section IV. After that, we conduct a series of experiments on four public datasets to evaluate FLSSM in Section V. Finally, Section VI concludes this work.

II. RELATED WORKS

A. Encrypted Model Aggregation Efficiency

Protecting federated learning model privacy with encryption technologies can enhance security but also requires extensive computation, which reduces the aggregation efficiency of federated learning. To improve the efficiency of federated learning after encryption, Chengliang Zhang et al. [14] encoded a batch of quantized gradients into a single long integer for one-time encryption, and developed new quantization and encoding schemes as well as a new gradient clipping technique. Ren-Yi Huang [15] introduced selective encryption of key model parameters to reduce computational/communication overhead, addressing the efficiency issues in HE-based federated learning. Sean Choi et al. [16] improved efficiency by offloading computationally intensive homomorphic encryption tasks to SmartNICs, reducing CPU utilization and improving resource allocation. Kai Cheng et al. [17] utilized Intel QAT hardware

accelerators, error-feedback gradient compression, and Huffman coding to simplify encryption and aggregation processes. Valentino Peluso et al. [18] introduced a Private Tensor Freezing (PTF) gating scheme that reduces the complexity of encryption/decryption, communication, and server aggregation over time. However, existing methods struggle to balance privacy and aggregation efficiency during model training, often sacrificing some privacy to improve efficiency.

B. Encrypted Model Access Control

Encrypted model parameters cannot be accessed, which creates security vulnerabilities during federated learning model aggregation, as malicious attackers may poison local models before sending them to the aggregation server. Existing model access control mechanisms can be categorized into multi-key homomorphic encryption and authentication/authorization approaches. Yuxuan Cai et al. [19] introduced “EMK-BFV,” combining multi-key homomorphic encryption with Trusted Execution Environments (TEEs) to enhance privacy, access control, and efficiency in federated learning. Jiachen Shen [20] expanded the scope of addressed privacy risks by encrypting with an aggregated public key and requiring joint decryption among participants. Xueyin Yang [21] designed a multi-private-key secure aggregation algorithm implementing homomorphic addition operations, allowing servers and clients to freely choose public-private key pairs, making it more suitable for deep models. Additionally, they compressed multi-dimensional data into one dimension, significantly reducing encryption/decryption time and ciphertext transmission communication. Caimei Wang [22] proposed a fingerprint-based subkey verification algorithm (FKM) to generate unique fingerprints for each subkey, while designing a gradient protection scheme to achieve higher security levels and reduce encryption overhead. Chun-I Fan et al. [23] proposed an identity-based multi-receiver homomorphic proxy re-encryption (IMH-PRE) scheme that utilizes homomorphic addition and re-encryption to provide improved encrypted data processing and access control. When adopting this scheme, participants can encrypt directly using public identities. Jing Wang et al. [24] proposed a hierarchical cloud-edge orchestration federated learning architecture for IoT, designing an IoT knowledge sharing method based on multi-level access control encryption to ensure knowledge confidentiality. Hui Lin et al. [25] proposed an attribute-based secure access control mechanism, discovering the relationship between users’ social attributes and their trustworthiness, where users’ trustworthiness depends on their social influence, which is then transformed into trust levels. They used federated deep learning to obtain optimal thresholds for trust levels and related access control parameters to improve access control accuracy and enhance privacy protection.

While existing federated learning model access control mechanisms improve privacy protection and security to some extent, they often involve complex key management. A key challenge is how to enhance the flexibility of control mechanisms to accommodate the dynamic nature of nodes in federated learning.

C. Encrypted Model Contribution Assessment

As local models are sent to the aggregation server in an encrypted state, it is difficult to assess the contributions of local models based on existing methods such as Shapley values or accuracy. Liangjiang Chen et al. [26] addressed encryption efficiency and contribution assessment issues by asynchronously allocating gradient weights securely based on user data quality. Ruizhe Yang et al. [27] proposed a federated learning method combining blockchain, homomorphic encryption, and reputation. Edge nodes with local data can train encrypted models using homomorphic encryption, and their contributions to aggregation are assessed through a reputation mechanism. Both models and reputations are recorded and verified on the blockchain through a consensus process, with rewards determined according to incentive mechanisms. Longyi Liu et al. [28] integrated federated learning into blockchain consensus protocols, proposing an FRConsensus algorithm based on model evaluation and stake election to overcome problems of passive participation and resource waste during model training. Additionally, they introduced model watermarking and ECC public key encryption mechanisms to protect parameter transmission. Biwen Chen et al. [29] first constructed an efficient non-interactive designated decryptor functional encryption scheme that protects training data privacy while maintaining high communication performance. Then, by combining this framework with a carefully designed blockchain, they proposed a blockchain-based federated learning framework providing fair compensation for medical image detection. Guilin Guan et al. [30] encrypted weights using multi-key homomorphic encryption to resist data recovery attacks launched by malicious edge servers and devices. Meanwhile, based on marginal loss techniques, they detected malicious clients or those uploading low-quality contributions, and assessed edge devices' contribution levels based on Shapley techniques to distribute rewards to participants. Ke Geng et al. [31] utilized arithmetic sharing to achieve submodel reconstruction and utility evaluation required in gradient Shapley under privacy protection, using shuffling and asymmetric encryption to ensure the privacy of test data collected from participating clients. Yingxin Li et al. [32] proposed a personalized residual federated secure learning scheme (PRFSL) based on homomorphic encryption and edge computing to guarantee security, timeliness, integrity of task data, and privacy needs of group workers, thereby improving encryption efficiency. Finally, they proposed a personalized privacy incentive mechanism based on evolutionary game theory to improve overall service utility.

Existing research has made some progress in incentive mechanisms for encrypted models, but in practical applications, due to encryption and privacy protection constraints, it remains challenging to balance privacy protection and the verifiability of contribution assessment.

III. PRELIMINARIES

A. Homomorphic Encryption

Homomorphic encryption refers to encryption algorithms where the ciphertexts satisfy homomorphic operational proper-

ties, meaning that encrypting data after performing addition or multiplication operations is equivalent to performing addition or multiplication operations on already encrypted data [33]. For any encryption function ϑ , if for any data A and data B, it satisfies $Dec_{\vartheta}(Enc_{\vartheta}(A) \odot Enc_{\vartheta}(B)) = A \oplus B$, then ϑ is considered to have homomorphism. Based on this, homomorphic encryption can be classified according to the supported operations into additive homomorphism and multiplicative homomorphism. Additive homomorphism indicates that encrypted ciphertexts support addition operations, while multiplicative homomorphism indicates that encrypted ciphertexts support multiplication operations. Encryption functions that only support addition, only support multiplication, or support a limited number of addition or multiplication operations are known as semi-homomorphic or partially homomorphic encryption; while encryption functions that satisfy unlimited addition or multiplication operations are fully homomorphic encryption.

Existing mainstream homomorphic encryption algorithms include CKKS, BFV, and Paillier algorithms. The CKKS algorithm is based on the Ring Learning With Errors problem (RLWE) and supports approximate homomorphic encryption. It has the advantage of supporting efficient floating-point operations, but also suffers from accumulated errors and reduced computational efficiency due to ciphertext expansion. The BFV algorithm is also based on the Ring Learning With Errors problem (RLWE) and supports fully homomorphic encryption. It has the advantage of supporting precise integer operations, but also has higher computational complexity and significant ciphertext expansion, leading to increased storage and transmission costs. The Paillier algorithm is based on the large number factorization problem and supports partially homomorphic encryption. It has the advantages of simple implementation and support for additive homomorphism, but only supports additive homomorphism and not multiplicative homomorphism, limiting its application scenarios.

B. Shamir's Secret Sharing

Shamir's Secret Sharing is a threshold scheme based on polynomial interpolation, allowing a secret S to be divided into n shares, where at least t shares are required to recover the secret [34]. The main steps are as follows:

- 1) **Initialization.** Suppose the data to be encrypted is s , which will be divided into n shares, with at least t shares required for decryption. Therefore, n is the number of participants, and t is the threshold in Shamir's secret sharing. Randomly select a large prime number p such that $(p > s)$.
- 2) **Encryption.** Arbitrarily select $t-1$ random numbers (x_1, \dots, x_n) , construct a polynomial $f(x)$ of degree $t-1$, where $f(0) = s$:

$$f(x) = s + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1} \pmod{p} \quad (1)$$

Hence, $(a_1, a_2, \dots, a_{t-1})$ are random coefficients $(0 \leq a_i < p)$. Arbitrarily select n different numbers $(x_1, \dots, x_n) \in F^+$, and calculate the corresponding $y_i = f(x_i) = s + a_1x_i + a_2x_i^2 + \dots + a_{t-1}x_i^{t-1} \pmod{p}$.

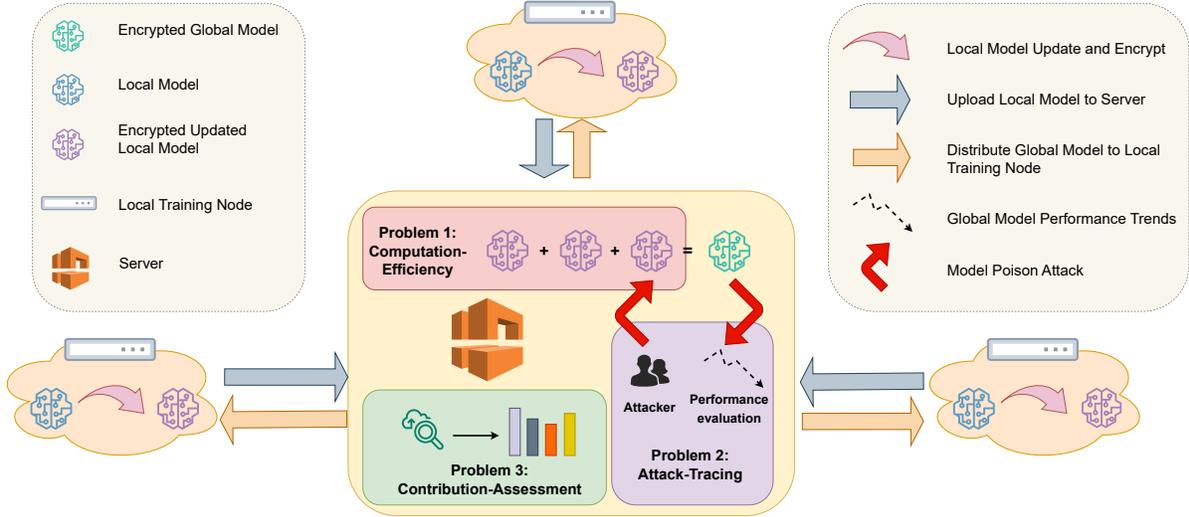


Fig. 1: Motivation for this work. Local training nodes update their local models, encrypt them using homomorphic encryption, and send them to the server. The server performs homomorphic computation, which draws three critical problems in global model aggregation: computation-efficiency, attack-tracing, and contribution-assessment. Computation-efficiency implies that homomorphic computation requires substantial computational resources. Attack-tracing refers to the covert nature and difficulty in tracing attacks initiated by local models in an encrypted state. Contribution-assessment refers to the challenge of calculating the contributions made by encrypted local models and providing fair rewards.

Each x_i and its corresponding y_i form a secret share, and (x_i, y_i) is distributed to the corresponding n participants.

- 3) **Decryption.** Collect at least t secret shares (x_i, y_i) , and use Lagrange interpolation to calculate $s = f(x = 0)$. The Lagrange basis polynomials $L_i(x)$ and the calculation of s are shown as follows:

$$L_i(x) = \prod_{\substack{1 \leq j \leq t \\ j \neq i}} \frac{x - x_j}{x_i - x_j} \quad (2)$$

$$s = \sum_{i=1}^t y_i L_i(0) \pmod{p} \quad (3)$$

C. Trusted Timestamps

Trusted timestamps are defined in RFC 3161 [35], essentially binding the hash value of a user’s electronic data with an authoritative time source. Based on this, the timestamp server generates a signature, producing an unforgeable timestamp file that proves a specific file or data existed before a certain time. Through hash values and digital signatures, trusted timestamps ensure that files have not been tampered with after the timestamp generation. Trusted timestamps rely on trusted Timestamp Authorities (TSAs), which are typically recognized by governments or industries and have high credibility. Anyone can verify the validity of a timestamp using the TSA’s public key, ensuring the transparency and fairness of the process. The main steps are as follows:

- 1) **Generate Digest.** Calculate a hash of the data for which a timestamp is to be generated, producing a data digest h :

$$h = H(D) \quad (4)$$

where H is the hash operation, D is the data.

- 2) **Initiate Timestamp Request.** The user sends a request to the Time Stamping Authority (TSA), including the hash value h of the data, to generate a timestamp σ :

$$\sigma = \text{Sign}_{TSA}(T) \quad (5)$$

Hence, $T = (h, \text{time})$ where h is the hash value of the data, and time is the current time, Sign_{TSA} is TSA signature the time and data digest to generate σ .

- 3) **Return Timestamp.** The TSA returns (T, σ) to the user.
- 4) **Verify Timestamp.** After receiving the timestamp (T, σ) returned by the TSA, the user can verify whether the signature σ is authentic using the TSA’s public key.

$$\text{Verify}_{TSA}(T, \sigma) = \text{True} \quad (6)$$

IV. METHODOLOGY

In this section, we will provide a detailed introduction of our proposed model. First we would like to present a statement regarding the practical FL privacy security problem. Then, we will provide an overview of the proposed FL framework. Our method consists of three main parts: Hierarchical Aggregation Mechanism Based on Homomorphic Encryption (HAM), Model Access Control Mechanism Based on Shamir’s Secret Sharing (MACM) and Incentive Mechanism Based on Trusted Time Intervals (IMTTI). Each module used in the framework will be described in separate subsections below.

A. Problem Statement

Our goal is to address a federated learning privacy security problem in a common scenario. This scenario involves two different organizations: regulatory authorities and participating

clients. Both organizations seek to collaboratively complete model training, but regulatory authorities need to oversee local models of participating clients to prevent malicious behavior. Therefore, how to balance the security, privacy, and efficiency of participating clients' local models becomes a key issue. Figure 1 depicts the federated learning training process under homomorphic encryption. To further elaborate, we can summarize this scenario into 3 key problems:

- 1) Encryption technology efficiency problem. The most effective existing method for protecting local model privacy and security is through encryption technologies, such as homomorphic encryption and secure multi-party computation, where local models are encrypted before transmission and aggregation. However, encryption technologies require substantial computational resources, placing high demands on the aggregation server's computational capabilities.
- 2) Historical model access control problem. Encryption technologies like homomorphic encryption can effectively protect the privacy of local models, but they also provide opportunities for model attacks launched by malicious nodes. Once encrypted, models cannot be accessed or supervised. Existing secure aggregation techniques struggle to filter out malicious models after local model encryption, allowing attackers to potentially launch poisoning attacks or backdoor attacks that disrupt the federated learning training process.
- 3) Incentive mechanism fairness and privacy problem. Federated learning requires multiple clients to jointly train models in an untrusted environment, necessitating incentive mechanisms to maintain the sustainability of federated learning model training. Fair incentive mechanisms can better motivate clients to train models on an individual basis. Existing research on incentive mechanism contribution fairness in federated learning can be categorized into approaches based on Shapley values, marginal contributions, gradient contributions, etc. However, the assessment process may require exposing some data or model information of the participants, potentially causing model privacy leakage issues.

TABLE I: List of Notations

Notations	Descriptions
FL	Federated Learning
h_g	Hash value of the global model
\mathcal{L}_κ	Loss function of Ln_κ
lm_λ^κ	Local model parameters of Ln_κ at the start of round λ
$lm_{u;\lambda}^\kappa$	Updated local model parameters of Ln_κ after round λ
$lm_{u;\lambda}^{\kappa;\Lambda}$	Parameters of the Λ -th shard of $lm_{u;\lambda}^\kappa$
$En_l^{\lambda;\Lambda}$	En_l responsible for aggregating the Λ -th shard of all local model parameters in round λ

B. Model Overview

As shown in Figure 2, the proposed model is a federated learning framework designed to address the aforementioned three issues.

This framework primarily comprises three modules, which will be elaborated upon in detail below. In our proposed distributed aggregation mechanism based on homomorphic encryption, the parameters of different slices of the local models are distributed to different edge aggregation nodes for aggregation, thereby achieving parallel aggregation of local models under homomorphic encryption and improving the efficiency of global model aggregation. This mechanism is described in Section IV-D. The historical model access control mechanism authorizes multiple trusted nodes to jointly hold the homomorphic encryption keys of local models. This approach ensures the security of local model storage while enabling model monitoring and malicious behavior traceability. This content is described in Section IV-E. Furthermore, the fairness of incentive mechanisms in federated learning is crucial for its sustainable and healthy development. However, existing privacy-preserving methods based on encryption technology in federated learning is hard to accurately evaluate model contributions. We have designed a trustworthy node participation activeness evaluation method to reward nodes, thereby providing fair incentives to participating entities while protecting the privacy of local models. This content is described in Section IV-F.

C. Initialization

In FLSSM, we denote all nodes' set is $N = n_1, n_2, \dots, n_\theta, \theta \in \mathbb{N}^+$ is the number of all nodes. N contains global aggregation nodes set, edge aggregation nodes set, local training nodes set. There has three type of nodes in FLSSM:

- 1) Global aggregation node(Gn): The set of global aggregation nodes is denoted as $GN = \{Gn_1, Gn_2, \dots, Gn_\vartheta\}$, where $\vartheta \in \mathbb{N}^+$. These nodes aggregate the various slices of local models, which have already been aggregated by the edge aggregation nodes, into the final global model.
- 2) Edge aggregation node(En): The set of edge aggregation nodes is denoted as $EN = \{En_1, En_2, \dots, En_\iota\}$, where $\iota \in \mathbb{N}^+$. These nodes receive the encrypted parameters of a specific layer from the local models, perform an initial aggregation of the local model parameters of the same layer, and then send them to Gn for final aggregation.
- 3) Local training node(Ln): The set of local training nodes is denoted as $LN = \{Ln_1, Ln_2, \dots, Ln_\kappa\}$, where $\kappa \in \mathbb{N}^+$. The local training nodes are responsible for training the local models, encrypting different slices of the local models, and sending the encrypted slices to different En for initial aggregation.
- 4) Supervise nodes(Sn): The set of supervise nodes is denoted as $SN = \{Sn_1, Sn_2, \dots, Sn_\eta\}$, where $\eta \in \mathbb{N}^+$. When the model may be under attack and local models need to be verified and reviewed, Gn will initiate a inspection request to Sn . Upon receiving the inspection request, Sn obtains the encryption keys of the local models to review them.

There are three types of models in FLSSM:

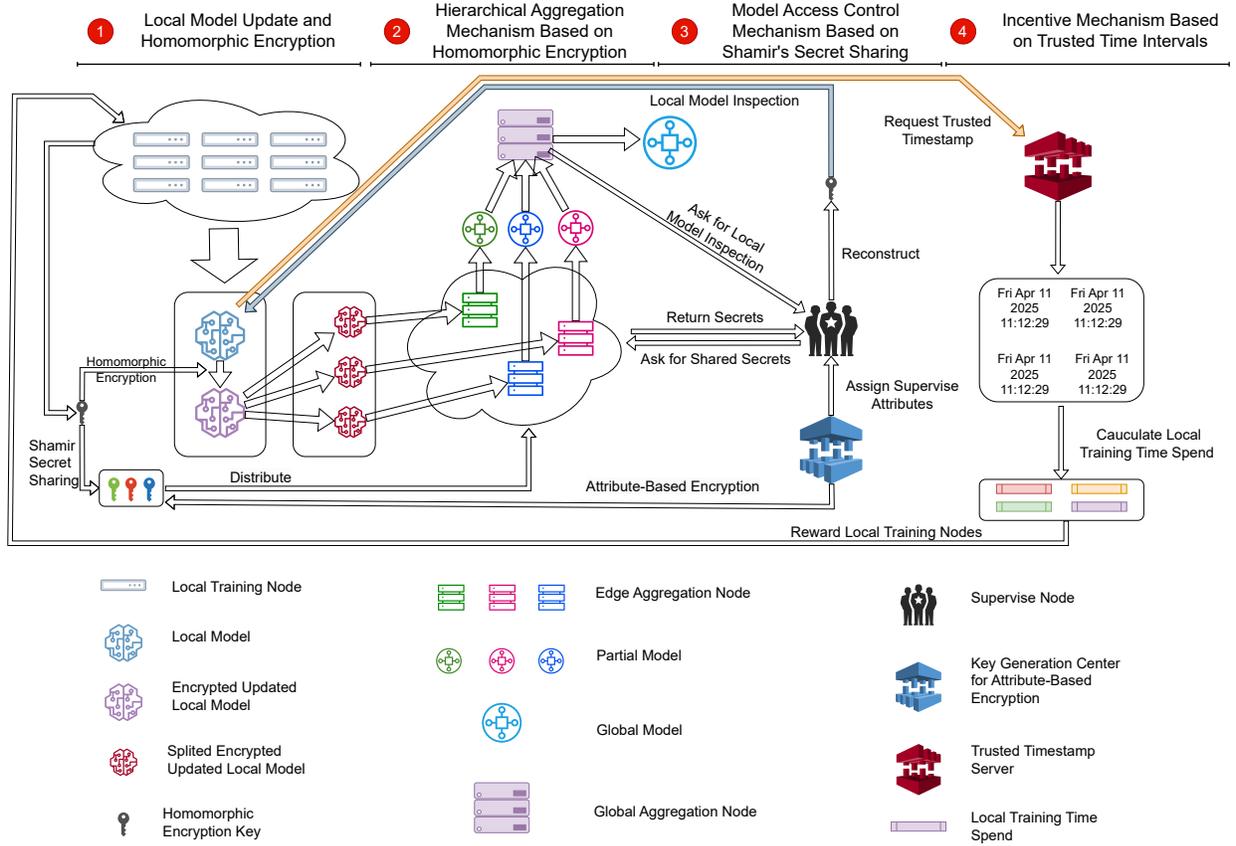


Fig. 2: Overview of FLSSM. Federated learning can be broadly divided into two parts: model training and model aggregation. Our research focuses on the second part, where we introduce edge aggregation nodes, supervise nodes, and trusted timestamp servers to improve model aggregation efficiency, enhance security, and provide reliable evidence for incentive mechanisms.

- Local model (lm): The local model set is denoted as $LM = \{lm_{\lambda}^1, lm_{\lambda}^2, \dots, lm_{\lambda}^{\kappa}\}$, where $\lambda \in \mathbb{N}^+$ represents the global training round, and κ represents the node id. lm_{λ}^{κ} represents the local model updated by Ln_{κ} in round λ .
- Partial model (pm): The partial model set is denoted as $PM = \{pm_{\lambda}^1, pm_{\lambda}^2, \dots, pm_{\lambda}^{\Lambda}\}$, where $\Lambda \in \mathbb{N}^+$ represents the Λ -th shard after the local model is evenly divided. After En receives the homomorphically encrypted parameters of a certain layer of the local models, it aggregates the received parameters of that layer through homomorphic computation to obtain pm . pm is sent to Gn for aggregation to obtain the global model of the λ -th round in ciphertext.
- Global model (gm): The global model set is denoted as $GM = \{gm_1, gm_2, \dots, gm_{\mu}\}$, where $\mu \in \mathbb{N}^+$ represents the global round number. The global model is aggregated by Gn from the collected pm .

The overall model workflow is as follows:

- 1) The aggregation node initializes the global model and distributes it to the local training nodes. Simultaneously, it calculates the hash value of the global model and sends a start timestamp request to the trusted timestamp server.
- 2) The local training nodes update the model using their local data, obtaining the updated local models.
- 3) The local training nodes homomorphically encrypt the local model according to different slices and send them to the corresponding edge aggregation nodes. Concurrently, they calculate the hash value of the local model and send an end timestamp request to the trusted timestamp server.
- 4) The local training nodes utilize Shamir's secret sharing scheme to perform attribute-based encryption on the homomorphic encryption keys, dividing them into multiple parts and distributing them to the supervise nodes (trusted nodes, such as representative nodes), stipulating that only entities with the 'supervise node' attribute can decrypt the keys.
- 5) The edge aggregation nodes aggregate the received local model slices to obtain pm .
- 6) The edge aggregation nodes send pm to Gn , and Gn aggregates the received pm to obtain gm .
- 7) The incentive mechanism calculates the time spent by each local training node on training based on the start and end times received by the trusted timestamp server and rewards the local training nodes accordingly based on their training time.
- 8) Gn distributes gm to Ln , and the above steps are repeated until the global model converges.

D. Hierarchical Aggregation Mechanism Based on Homomorphic Encryption (HAM)

In FLSSM, we propose a Hierarchical Aggregation Mechanism Based on Homomorphic Encryption (HAM) to protect the privacy and security of local models using encryption technology. Existing methods for protecting the privacy of federated learning models using encryption technology include homomorphic encryption, secure multi-party computation, and differential privacy. Among these, methods using homomorphic encryption technology offer the highest level of security but require significant computational resources, severely impacting the training efficiency of federated learning. In this work, we propose a distributed aggregation mechanism.

In HAM, we choose CKKS to encrypt the model parameters because CKKS supports efficient floating-point arithmetic, making it suitable for scenarios requiring high-precision calculations such as machine learning and deep learning. Although there is a certain issue of error accumulation, it can be controlled through reasonable parameter settings in many practical applications. The core idea of the Cheon-Kim-Kim-Song (CKKS) algorithm [36] is to perform homomorphic encryption over complex numbers, allowing direct execution of addition and multiplication operations on ciphertexts without decryption. This characteristic makes the CKKS algorithm particularly suitable for processing numerical data, such as floating-point and complex numbers, and thus it has a wide range of applications in machine learning, data analysis, and privacy protection.

We flatten the local models and then evenly distribute the flattened parameters according to the number of different edge aggregation nodes. Each part is homomorphically encrypted separately, and the encrypted model slices are sent to different edge aggregation nodes. These slices are aggregated in parallel as pm at the edge aggregation nodes. This approach improves the efficiency of homomorphic encryption while ensuring model privacy. This method enhances the model aggregation efficiency under homomorphic encryption and reduces the time required for homomorphic encryption computations during model aggregation. The algorithm details are shown in Algorithm 1.

The main steps are as follows:

- 1) Ln performs local training to obtain the updated local model. Assuming the current Ln id is ρ , and the global model training round is λ , the updated local model of Ln can be denoted as $lm_{u;\lambda}^\rho$;

$$lm_{u;\lambda}^\rho = lm_\lambda^\rho - \nabla \mathcal{L}_\rho(lm_\lambda^\rho) \quad (7)$$

Here, ∇ is the learning rate, and \mathcal{L}_ρ is the loss function of Ln_ρ :

$$\mathcal{L}_\rho = \frac{1}{|\mathcal{D}_\rho|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_\rho} \ell(lm_\lambda^\rho; \mathbf{x}, y), \quad (8)$$

Here, \mathcal{D}_ρ is the dataset of Ln_ρ , (\mathbf{x}, y) are the data points in \mathcal{D}_ρ , and $\ell(lm_\lambda^\rho; \mathbf{x}, y)$ is the loss value of lm_λ^ρ on the data \mathbf{x}, y .

- 2) Ln_ρ flattens the local model and divides it equally into several shards, then homomorphically encrypts the

different model shards. We adopt the CKKS algorithm as the homomorphic encryption algorithm in FLSSM. Assuming the encryption parameters of Ln_ρ include the polynomial modulus q_ρ , polynomial degree N_ρ , scaling factor Δ_ρ , and public-private key pair (pk_ρ, sk_ρ) , the process of homomorphically encrypting the Λ -th shard $(lm_{u;\lambda}^{\rho;\Lambda})$ of $lm_{u;\lambda}^\rho$ can be represented as:

$$[lm_{u;\lambda}^{\rho;\Lambda}] = \text{CKKS.Encrypt}(pk_\rho, \Delta_\rho \cdot lm_{u;\lambda}^{\rho;\Lambda}) \quad (9)$$

- 3) Ln sends the encrypted model parameters of different slices to the corresponding En . Assuming $[lm_{u;\lambda}^{\rho;\Lambda}]$ corresponds to $En_i^{\lambda;\Lambda}$, meaning that in the λ -th round, En_i is responsible for aggregating the model parameters of the Λ -th shard of all local models, we have:

$$Ln \rightarrow En_i^{\lambda;\Lambda} : [lm_{u;\lambda}^{\rho;\Lambda}], \quad (10)$$

- 4) En performs homomorphic computation on the received model parameters of the same shard from different Ln to obtain $[pm]$;

$$[pm]_\lambda^\Lambda = \text{CKKS.Add} \left(\frac{1}{\kappa}, \sum_{\rho=1}^{\kappa} [lm_{u;\lambda}^{\rho;\Lambda}] \right) \quad (11)$$

- 5) En sends pm to Gn , and Gn concatenates the received $[pm]$ to obtain the ciphertext of the global model $[gm]$.

$$[gm]_\lambda = \text{CKKS.Concat}(\{[pm]_\lambda^\Lambda\}_{\Lambda=1}^{\varrho}) \quad (12)$$

Here, ϱ is the total number of model slices, which also the En number.

- 6) Gn distributes $[gm]$ to Ln , which decrypts it for the next round of model training.

$$Gn \xrightarrow{[gm]} L_i : \text{Train}(\text{CKKS.Decrypt}(sk_i, [gm])), \quad (13)$$

$$\forall i \in \{1, 2, \dots, \kappa\}$$

In HAM, we homomorphically encrypt the local model according to different model slices and send them to designated edge aggregation nodes. The edge aggregation nodes aggregate these and send the result to the global aggregation node. Compared to the traditional aggregation mechanism where a single server performs homomorphic encryption and aggregation computation, this reduces the required time and alleviates the computational burden on the server during the aggregation process, thereby improving computational efficiency while ensuring the security of model parameters.

E. Model Access Control Mechanism Based on Shamir's Secret Sharing (MACM)

In FLSSM, we propose a model access control mechanism based on Shamir's secret sharing algorithm. Protecting the privacy and security of local models through encryption effectively prevents data leakage but also creates opportunities for malicious nodes to exploit. Since the server cannot access local models and can only aggregate encrypted local models, we introduce a supervisory node to inspect and validate local models. For malicious local models, the supervisory node can exclude them from the current round of global model aggregation and penalize the corresponding training node.

Algorithm 1 Hierarchical Aggregation Mechanism Based on Homomorphic Encryption (HAM)

Require:

- 1: Ln_ρ : Local nodes with id ρ
- 2: En_i : Edge aggregation nodes
- 3: Gn : Global aggregation node
- 4: λ : Current training round
- 5: κ : Number of local nodes
- 6: ϱ : Total number of model slices

Ensure: Global model for next round training

- 7: **function** LOCALTRAINING(Ln_ρ, λ)
 - 8: Update local model: $lm_{u;\lambda}^\rho \leftarrow lm_\lambda^\rho - \nabla \mathcal{L}_\rho(lm_\lambda^\rho)$
 - 9: Flatten and split $lm_{u;\lambda}^\rho$ into ϱ shards
 - 10: **for** each shard $\Lambda \in \{1, \dots, \varrho\}$ **do**
 - 11: $[lm_{u;\lambda}^{\rho;\Lambda}] \leftarrow \text{CKKS.Encrypt}(pk_\rho, \Delta_\rho \cdot lm_{u;\lambda}^{\rho;\Lambda})$
 - 12: Send $[lm_{u;\lambda}^{\rho;\Lambda}]$ to corresponding $En_i^{\lambda;\Lambda}$
 - 13: **function** EDGEAGGREGATION(En_i, λ, Λ)
 - 14: Collect encrypted shards from all local nodes
 - 15: $[pm]_\lambda^\Lambda \leftarrow \text{CKKS.Add}(\frac{1}{\kappa}, \sum_{\rho=1}^{\kappa} [lm_{u;\lambda}^{\rho;\Lambda}])$
 - 16: Send $[pm]_\lambda^\Lambda$ to Gn
 - 17: **function** GLOBALAGGREGATION(Gn, λ)
 - 18: Collect all $[pm]_\lambda^\Lambda$ from edge nodes
 - 19: $[gm]_\lambda \leftarrow \text{CKKS.Concat}(\{[pm]_\lambda^\Lambda\}_{\Lambda=1}^{\varrho})$
 - 20: **for** each $Ln_i, i \in \{1, \dots, \kappa\}$ **do**
 - 21: Send $[gm]_\lambda$ to Ln_i
 - 22: **function** LOCALDECRYPTION($Ln_i, [gm]_\lambda$)
 - 23: $gm_\lambda \leftarrow \text{CKKS.Decrypt}(sk_i, [gm]_\lambda)$
 - 24: Begin next round training with gm_λ
-

When the global model is under attack, the supervisory node can access historical local models to trace the originator of the malicious behavior.

To protect the privacy of local models, we apply homomorphic encryption. To prevent the supervisory node from arbitrarily accessing local models and causing privacy breaches, we split the homomorphic encryption key of Ln into multiple shares using Shamir's secret sharing technique. These shares are distributed among local training nodes, edge aggregation nodes, global aggregation nodes, and the supervisory node. When the model is subject to a malicious attack, the supervisory node sends queries to nodes holding key shares to request the homomorphic encryption key. Shamir's secret sharing enables a (t,n) access control mechanism, allowing the supervisory node to decrypt the key using shares from other nodes if some devices are offline.

However, storing the homomorphic encryption key across multiple nodes introduces the risk of malicious nodes colluding to reconstruct the key. To mitigate this, after splitting the homomorphic encryption key at local nodes, we further encrypt the shares using attribute-based encryption, ensuring that only the supervisory node can decrypt them. Even if malicious nodes obtain key shares, they cannot reconstruct the homomorphic encryption key, thereby safeguarding the privacy and security of local models.

MACM consists of two main modules: key distribution and

key reconstruction. In the key distribution module, Ln splits and encrypts the homomorphic encryption key of the local model and distributes the shares to trusted nodes. In the key reconstruction module, when the model is potentially under a malicious attack, Sn initiates key reconstruction requests to multiple trusted nodes, collects the key shares, decrypts, and reconstructs them to obtain the homomorphic encryption key for reviewing the local model.

The main steps of key distribution are as follows:

- 1) Ln_ρ splits the homomorphic encryption key sk_ρ into multiple shares using Shamir's secret sharing mechanism. According to Section III-B, assuming the total number of shares is n , the threshold is t , and the random number is p , a polynomial $f(x)$ is constructed as follows:

$$f(sk_\rho) = sk_\rho + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1} \pmod{p} \quad (14)$$

Select n distinct numbers $x_i, i = \{1, \dots, n\}$, to obtain the secret shares (x_i, y_i) .

- 2) Initialize the attribute-based encryption system to generate a master key Amk and a public key Apk . Generate a specific private key Ask for each node based on its attributes. Ln_ρ encrypts the key shares using attribute-based encryption, configured such that only the supervisory node can decrypt them. Assuming the supervisory node's attribute set is S , the encryption is:

$$CT_i = \mathcal{E}(Apk, (x_i, y_i), S), i \in \{1, \dots, n\} \quad (15)$$

Thus, CT_i is the ciphertext of (x_i, y_i) after attribute-based encryption.

- 3) Ln distributes the encrypted key shares CT_i to the representative nodes, aggregation server, and supervisory node.

When the global model is suspected of being compromised by poisoning or backdoor attacks, leading to performance degradation, Gn requests the supervisory node to validate the local models. The supervisory node can request homomorphic encryption key shares from nodes holding them to review the local models stored in En . The homomorphic encryption key shares are protected by attribute-based encryption, ensuring that only the supervisory node can decrypt them. Additionally, the key is split into multiple shares, and the supervisory node can only reconstruct the homomorphic encryption key if a sufficient proportion of nodes consent. This prevents the supervisory node from arbitrarily accessing local models and causing privacy breaches. The process of the supervisory node reconstructing the homomorphic encryption key to review local models is as follows:

- 1) Sn sends key reconstruction requests to the representative nodes, aggregation server, and supervisory node holding the homomorphic encryption key shares to obtain the encrypted shares CT_i .
- 2) Sn collects key shares meeting the (t,n) threshold and decrypts them using attribute-based encryption:

$$(x_i, y_i) = \mathcal{D}(CT_i, SK_M) \quad (16)$$

Algorithm 2 Model Access Control Mechanism Based on Shamir's Secret Sharing (MACM)

Require:

- 1: Ln_ρ : Local node with id ρ
- 2: Sn : Supervisory node
- 3: n : Total number of key shares
- 4: t : Threshold value
- 5: p : Large prime number
- 6: S : Attribute set of supervisory node

Ensure: Secure model access control

```

7: function KEYSHARING( $sk_\rho, n, t, p$ )
8:   Generate random coefficients  $a_1, \dots, a_{t-1}$ 
9:   Construct polynomial  $f(x) = sk_\rho + \sum_{i=1}^{t-1} a_i x^i$ 
   (mod  $p$ )
10:  for  $i \leftarrow 1$  to  $n$  do
11:    Select unique  $x_i$ 
12:    Compute  $y_i = f(x_i)$ 
13:    Store share pair  $(x_i, y_i)$ 
   return  $\{(x_i, y_i)\}_{i=1}^n$ 
14: function ATTRIBUTEBASEDENCRIPTION( $shares, S$ )
15:  Generate  $(Amk, Apk) \triangleright$  Master key and public key
16:  for each share pair  $(x_i, y_i)$  in  $shares$  do
17:     $CT_i \leftarrow \mathcal{E}(Apk, (x_i, y_i), S)$ 
18:  Distribute  $CT_i$  to trusted nodes return  $\{CT_i\}_{i=1}^n$ 
19: function KEYRECONSTRUCTION( $Sn$ )
20:   $shares \leftarrow \emptyset$ 
21:  for  $i \leftarrow 1$  to  $t$  do
22:    Request  $CT_i$  from trusted nodes
23:     $(x_i, y_i) \leftarrow \mathcal{D}(CT_i, SK_M) \triangleright$  Decrypt using ABE
24:     $shares \leftarrow shares \cup \{(x_i, y_i)\}$ 
25:   $sk_\rho \leftarrow \sum_{i=1}^t y_i \prod_{j \neq i} \frac{-x_j}{x_i - x_j} \pmod{p}$  return  $sk_\rho$ 
26: function MODELINSPECTION( $sk_\rho$ )
27:  Request encrypted local model  $[m]$  from edge nodes
28:   $m \leftarrow \mathcal{D}([m]) \triangleright$  Decrypt using reconstructed key
29:  if IsModelMalicious( $m$ ) then
30:    DeductStake( $Ln_\rho$ )
31:    RemoveFromGlobalModel( $m$ )
32:  if IsRepeatOffender( $Ln_\rho$ ) then
33:    AddToBlacklist( $Ln_\rho$ )

```

- 3) Sn reconstructs the homomorphic encryption key from the key shares using Shamir's secret sharing technique:

$$sk_\rho = \sum_{i=1}^t y_i \prod_{j \neq i} \frac{-x_j}{x_i - x_j} \pmod{p} \quad (17)$$

- 4) Sn sends an access request to the edge aggregation node to obtain the local model shares and decrypts them using the homomorphic encryption key:

$$m = \mathcal{D}([m]) = \frac{L([m]^\varpi \pmod{\sigma^2})}{L(g^\varpi \pmod{\sigma^2})} \pmod{\sigma} \quad (18)$$

Upon detecting a malicious node, the supervisory node deducts its stake and removes its local model from the global model aggregation, redistributing the updated global model.

Nodes that repeatedly engage in malicious behavior are added to a blacklist and barred from participating in local training.

F. Incentive Mechanism Based on Trusted Time Intervals (IMTTI)

Incentive mechanisms are a critical component of federated learning, fostering its sustainable development. Existing studies on federated learning incentives can be categorized into contribution-based, reputation-based, cryptocurrency-based, or blockchain-based approaches. However, these mechanisms primarily target traditional federated learning, evaluating local models in plaintext. In FLSSM, we propose a novel incentive mechanism to reliably assess the participation enthusiasm of local training nodes operating under encrypted conditions, thereby promoting the sustainability of federated learning. Specifically, we leverage a trusted timestamp server to measure the time taken by local training nodes to complete model training. Nodes that complete training faster are considered to have invested more computational resources and are thus rewarded more generously. To prevent nodes from falsely reporting training completion times to gain higher rewards, we mandate that the global aggregation node, responsible for distributing the global model, sends a training start timestamp request to the trusted timestamp server. Local training nodes send an end timestamp request upon completing training. Each timestamp request must include the model's hash value, which is verified against the hash of the aggregated local model during reward allocation to ensure the trustworthiness of the training time.

The main steps of IMTTI are as follows:

- 1) The global aggregation node distributes the global model to Ln , computes the hash value h_g of the global model, and sends h_g to the trusted timestamp server to mark the training start time:

$$h_g = H(gm) \quad (19)$$

$$T = (h_g, time) \quad (20)$$

$$\sigma_g = Sign_{TSA}(T) \quad (21)$$

Hence, $Sign_{TSA}$ denotes the trusted timestamp server, σ_g represents the start time of local model training, and $time$ is the current time.

- 2) Upon receiving gm , the local model Ln_ρ conducts training. After training is complete, it follows step 2 in Section IV-D to shard and encrypt the local model. It then computes the hash values of the encrypted local model and sends these hash values to the trusted timestamp server to initiate an end timestamp request:

$$h_l^\rho = H(lm_\lambda^\rho) \quad (22)$$

- 3) Upon receiving the hash values of the local model from Ln_ρ , the trusted timestamp server generates a trusted timestamp h_l^ρ and calculates the time difference between the end and start times, producing a trusted time interval σ_d^ρ :

$$\sigma_l^\rho = Sign_{TSA}(h_l^\rho) \quad (23)$$

Algorithm 3 Incentive Mechanism Based on Trusted Time Intervals (IMTTI)

Require:

- 1: Gn : Global aggregation node
- 2: Ln_ρ : Local node with id ρ
- 3: TSA : Trusted timestamp server
- 4: gm : Global model
- 5: R_t : Total reward per round
- 6: κ : Number of local nodes

Ensure: Reward distribution based on training time

```

7: function INITIATETRAINING( $Gn, gm$ )
8:    $h_g \leftarrow H(gm)$   $\triangleright$  Compute hash of global model
9:    $time \leftarrow \text{CurrentTime}()$ 
10:   $T \leftarrow (h_g, time)$ 
11:   $\sigma_g \leftarrow \text{Sign}_{TSA}(T)$   $\triangleright$  Get start timestamp
12:  for each  $Ln_\rho$  do
13:    Send  $(gm, \sigma_g)$  to  $Ln_\rho$ 
14:    return  $\sigma_g$ 
15: function LOCALTRAINING( $Ln_\rho, gm$ )
16:  Train local model  $lm_\lambda^\rho$  using  $gm$ 
17:  Shard and encrypt local model
18:   $h_l^\rho \leftarrow H(lm_\lambda^\rho)$   $\triangleright$  Compute hash of local model
19:  Send  $h_l^\rho$  to  $TSA$ 
20:   $\sigma_l^\rho \leftarrow \text{Sign}_{TSA}(h_l^\rho)$   $\triangleright$  Get end timestamp return
     $(lm_\lambda^\rho, h_l^\rho, \sigma_l^\rho)$ 
21: function CALCULATETIMEINTERVAL( $TSA, \sigma_g, \sigma_l^\rho$ )
22:   $\sigma_d^\rho \leftarrow \sigma_l^\rho - \sigma_g$   $\triangleright$  Calculate time interval return  $\sigma_d^\rho$ 
23: function CALCULATEREWARDS( $\{\sigma_d^\rho\}_{\rho=1}^\kappa, R_t$ )
24:  for each  $Ln_\rho$  do
25:     $\mathfrak{C}_\rho \leftarrow e^{-0.1\sigma_d^\rho}$   $\triangleright$  Calculate contribution
26:   $total\_contribution \leftarrow \sum_{\rho=1}^\kappa \mathfrak{C}_\rho$ 
27:  for each  $Ln_\rho$  do
28:     $R^\rho \leftarrow \frac{\mathfrak{C}_\rho}{total\_contribution} R_t$   $\triangleright$  Calculate reward
29:  return  $\{R^\rho\}_{\rho=1}^\kappa$ 
30: function VERIFYANDDISTRIBUTEREWARDS( $Gn$ )
31: for each  $Ln_\rho$  do
32:  Verify  $h_l^\rho$  matches received encrypted model
33:  if verification successful then
34:    Distribute reward  $R^\rho$  to  $Ln_\rho$ 
35:  else
36:    Skip reward for  $Ln_\rho$ 

```

$$\sigma_d^\rho = \sigma_l^\rho - \sigma_g \quad (24)$$

- 4) The incentive mechanism rewards local models based on their trusted time intervals. For Ln_ρ , the total reward per round R_t is fixed, and the contribution of Ln_ρ is calculated as:

$$\mathfrak{C}_\rho = e^{-0.1\sigma_d^\rho} \quad (25)$$

The reward received by Ln_ρ is then expressed as:

$$R^\rho = \frac{\mathfrak{C}_\rho}{\sum_{\rho=1}^\kappa \mathfrak{C}_\rho} R_t \quad (26)$$

V. PERFORMANCE EVALUATION

In this section, we conduct experiments and ablation studies on two public datasets to validate our model. We first introduce the experimental environment and datasets in Sections V-A and V-B. Next, we report the experimental results for the three modules in Sections V-C and V-D.

A. Environment

Our experiments were conducted on a CentOS server, using PyTorch version 2.3.0, CUDA version 12.4, with a system equipped with 396GB of RAM and a 1.8TB hard disk, the GPUs in the server are Quadro RTX 5000 (with a VRAM of 16GB). We use dirichlet distribution to split dataset, the alpha is set to 0.5.

TABLE II: Simulation Parameters

Parameters	Value
Learning rate	0.001
Batchsize	64
Alpha of data Non-IID distribution	0.5
Number of Communication Round	100
Number of Supervise Node	1
Epoch in each round	5
Total reward in one round (R_t)	10
Threshold of Shamir Secret Shares	3
Number of Shamir Secret Shares	5
Malicious attack	Sign_Flipping Attack

B. Datasets

- CIFAR-10 [37]: The CIFAR-10 dataset contains 3-channel RGB color images across 10 categories, with each image sized at 32×32 pixels. Each category includes 6,000 images, resulting in a total of 50,000 training images and 10,000 test images.
- Fashion-MNIST [38]: The Fashion-MNIST dataset consists of 70,000 frontal images of various fashion products, categorized into 10 classes. All images are grayscale with dimensions of 28×28 pixels. The training set contains 60,000 images, while the test set contains 10,000 images.

C. Comparative Experiment

1) *Computation-efficiency Experiment:* Figure 3 evaluates the relationship between aggregation time and model accuracy under two public datasets, CIFAR10 and Fashion-MNIST, comparing our proposed HAM scheme with a baseline that aggregates directly without edge aggregation nodes. Here, ‘‘CKKS’’ refers to the CKKS homomorphic encryption algorithm, where aggregation is performed directly by the global aggregation node (Gn) without edge aggregation nodes. ‘‘HAM’’ denotes the HAM aggregation algorithm, with ‘‘HAM-3’’ indicating 3 edge aggregation nodes, ‘‘HAM-5’’ indicating 5 edge aggregation nodes, and ‘‘HAM-10’’ indicating 10 edge aggregation nodes. Lower model aggregation time indicates higher computational efficiency. When the number of training nodes is the same, a greater number of edge aggregation nodes results in shorter aggregation times and higher aggregation efficiency. Furthermore, the results demonstrate that for both

datasets, our proposed scheme effectively enhances the homomorphic computation efficiency of encrypted models by utilizing edge aggregation nodes. This improvement is attributed to the fact that encrypted local model parameters are evenly distributed across different edge aggregation nodes; thus, more edge aggregation nodes mean fewer parameters for each node to aggregate. When the number of training nodes varies, the aggregation time distribution remains similar. For the same number of training nodes, model accuracy is comparable. However, when the number of training nodes increases, model accuracy slightly decreases. This indicates that our proposed scheme improves the computational efficiency of encrypted models without significantly impacting model performance. Additionally, edge aggregation nodes enable model parameters to be stored separately rather than on a single central server, enhancing the storage security of model parameters.

D. Ablation Experiment

1) *Attack-tracing Experiment*: Figure 4 illustrates the relationship between the supervisory node’s review of local models under our proposed MACM mechanism and model performance when varying proportions of malicious nodes are present among local training nodes (L_n), with the number of L_n set to 10. Figure 4a shows the impact of the supervisory node’s review on model performance in the CIFAR10 dataset when the proportions of malicious nodes are $\{0.1, 0.2\}$. When the supervisory node does not review local models ($S=0$), model performance rapidly drops to between 0.1 and 0.2, with higher proportions of malicious nodes leading to lower performance. When the supervisory node reviews local models ($S=1$), model performance shows significant improvement compared to the case without reviews. Figure 4b demonstrates the supervisory node’s impact on model performance in the Fashion-MNIST dataset. Consistent with Figure 4, the supervisory node’s review of local models effectively enhances model performance. However, in both Figures 4 and 4a, the proportion of malicious nodes still affects model performance. This may be due to training data being allocated to malicious nodes during dataset distribution. When the supervisory node identifies and excludes malicious nodes from further participation in federated training, the amount of data available for training decreases. Consequently, a higher proportion of malicious nodes results in lower model performance.

2) *Contribution-assessment Experiment*: Figure 5 illustrates the attack and defense mechanisms involving trusted timestamps in the incentive mechanism. Figure 5a displays the trusted timestamp received by a malicious node and its corresponding time. If a malicious node attempts to tamper with the trusted timestamp to reduce its reported training time and thereby gain higher rewards, it may choose to advance its training completion time. We simulated the process of a malicious node tampering with the trusted timestamp. After modifying the timestamp, the node sends it to the trusted timestamp server, which verifies whether the timestamp has been altered. In Figure 5b, the malicious node tampers with the received trusted timestamp, changing the time to 3 seconds earlier. When this is sent to the trusted timestamp server for

verification, the tampering is detected, and the server returns a verification failure result. Moreover, tampering with the time causes changes in the timestamp, which we have highlighted with a red box.

Figure 6 illustrates the reward mechanism of our proposed IMTTI incentive mechanism, with the number of L_n set to 10. Figures 6a, 6d, 6g, and 6j depict the stake changes of local training nodes in the CIFAR10 and Fashion-MNIST datasets when the proportions of malicious nodes are $\{0.1, 0.2\}$, respectively. Figures 6b, 6e, 6h, and 6k show the reward changes for each local training node across training rounds. Meanwhile, Figures 6c, 6f, 6i, and 6l illustrate the training time of each local training node per round. The pairs 6b-6c, 6d-6f, 6g-6i, and 6j-6l represent different experimental metrics within the same training setup.

For malicious nodes, their stake rapidly drops to 0 as they are identified by the supervisory node and barred from participating in local training. Examples include $device_0$ in 6a and 6g, and both $device_0$ and $device_1$ in 6d and 6j. Since the total reward per round is fixed, according to Equation 26, the reward each training node receives is inversely proportional to its training time. Figure 6 effectively validates this relationship.

VI. CONCLUSION

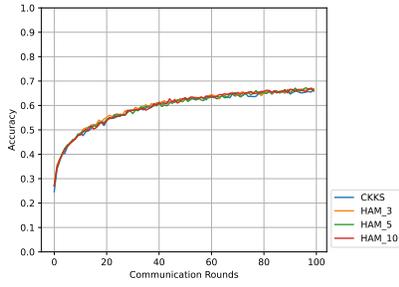
This article proposes FLSSM, a framework for encrypted federated learning models, to tackle the challenges of “computation-efficiency,” “attack-tracing,” and “contribution-assessment.” First, we employ edge aggregation nodes to enhance the aggregation efficiency of encrypted models, addressing the “computation-efficiency” issue. Second, we introduce a model access control mechanism, where a trusted third-party node reviews encrypted models to detect malicious attacks, resolving the “attack-tracing” problem. Finally, we propose an incentive mechanism that leverages a trusted timestamp server to ensure the reliability of local model training times, enabling fair contribution assessment of local training nodes under homomorphic encryption, thus addressing the “contribution-assessment” challenge. Experiments on two real-world public datasets validate the feasibility of our approach. The results demonstrate that our model not only improves aggregation efficiency and enables tracing of malicious nodes but also effectively evaluates node contributions.

VII. ACKNOWLEDGMENTS

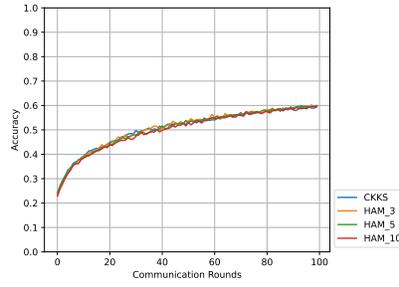
This work was supported by the National Natural Science Foundation of China under Grant No. 62272024.

REFERENCES

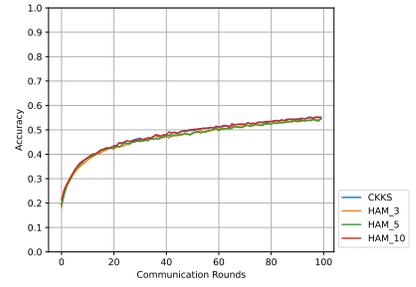
- [1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. B. Charles, G. Cormode, R. Cummings, R. G. L. D’Oliveira, S. Y. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, O. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. X. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and



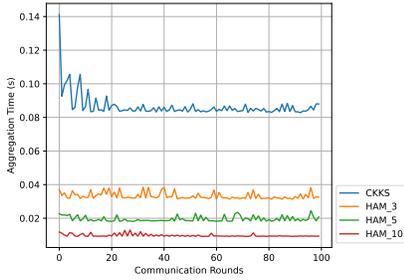
(a) Global Model Accuracy, CIFAR10, $L_n=10$



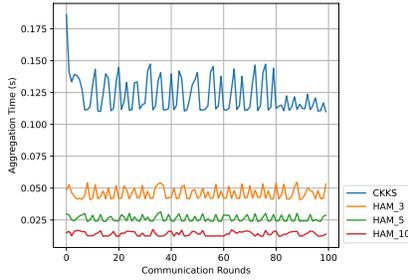
(b) Global Model Accuracy, CIFAR10, $L_n=20$



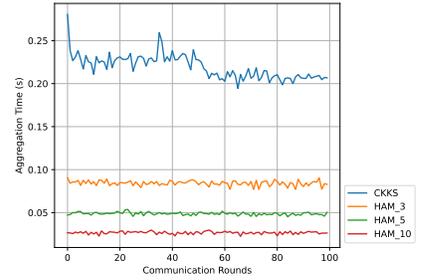
(c) Global Model Accuracy, CIFAR10, $L_n=50$



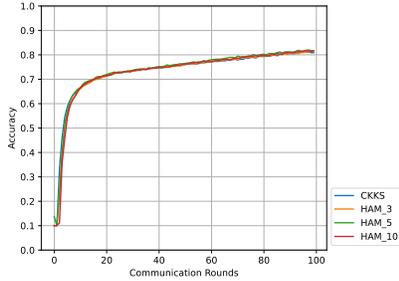
(d) Aggregation Time, CIFAR10, $L_n=10$



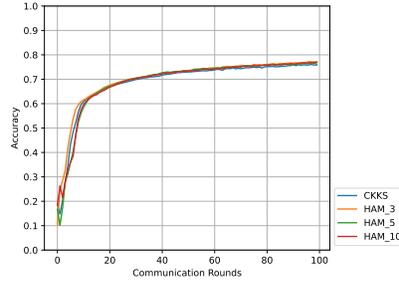
(e) Aggregation Time, CIFAR10, $L_n=20$



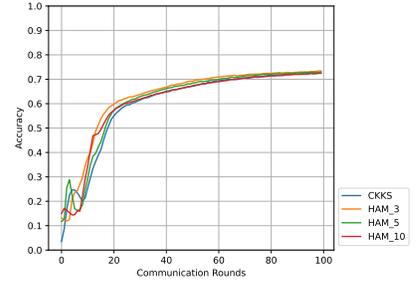
(f) Aggregation Time, CIFAR10, $L_n=50$



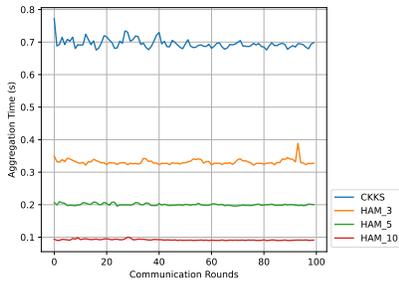
(g) Global Model Accuracy, Fashion-MNIST, $L_n=10$



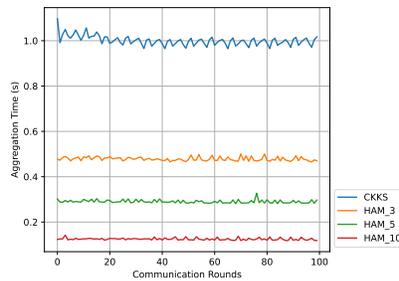
(h) Global Model Accuracy, Fashion-MNIST, $L_n=20$



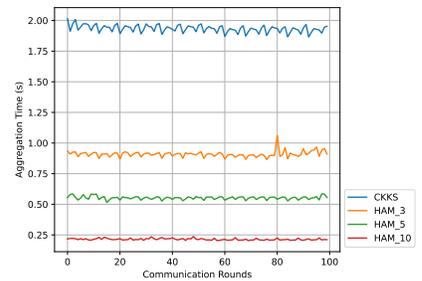
(i) Global Model Accuracy, Fashion-MNIST, $L_n=50$



(j) Aggregation Time, Fashion-MNIST, $L_n=10$

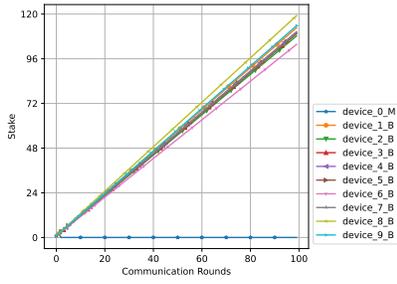


(k) Aggregation Time, Fashion-MNIST, $L_n=20$

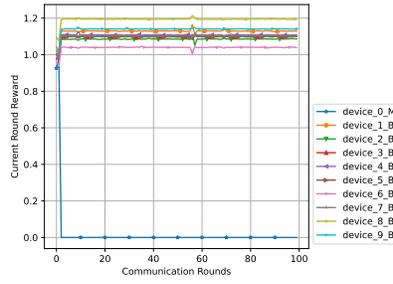


(l) Aggregation Time, Fashion-MNIST, $L_n=50$

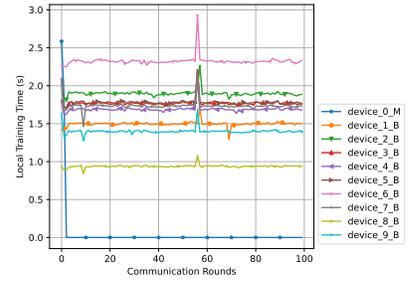
Fig. 3: Accuracy of HAM and CKKS algorithms for CIFAR10 and Fashion-MNIST classification tasks. Figures 3a-3c present the global model accuracy on CIFAR10 dataset for local training node (L_n) counts in $\{10, 20, 50\}$ and Edge Aggregation Nodes (E_n) counts in $\{3, 5, 10\}$. Figures 3d-3f show the global model aggregation time on CIFAR10 dataset with the same parameter settings. Similarly, Figures 3g-3i illustrate global model accuracy on the Fashion-MNIST dataset, and Figures 3j-3l present the global model aggregation time on Fashion-MNIST dataset under the same parameter configurations.



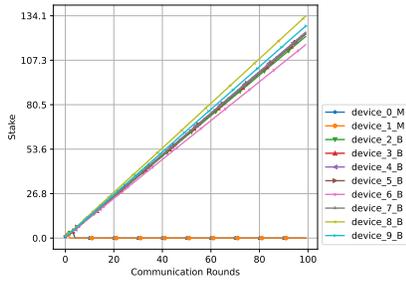
(a) Stake, CIFAR10, $L_n=10, M = 0.1$



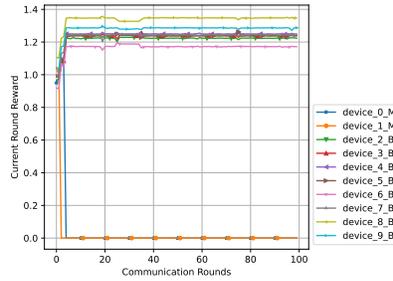
(b) Reward in One Round, CIFAR10, $L_n=10, M = 0.1$



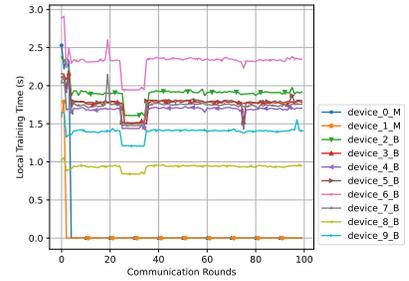
(c) Local Training Time, CIFAR10, $L_n=10, M = 0.1$



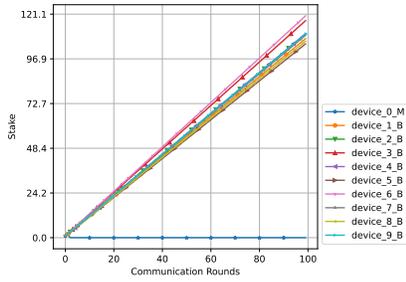
(d) Stake, CIFAR10, $L_n=10, M = 0.2$



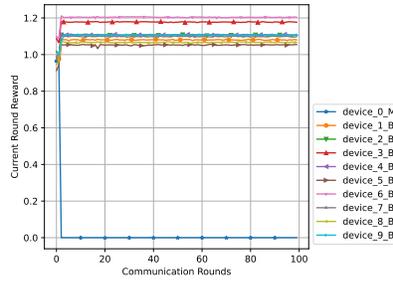
(e) Reward in One Round, CIFAR10, $L_n=10, M = 0.2$



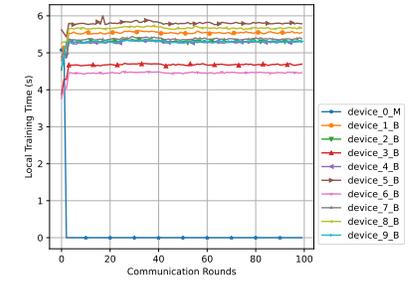
(f) Local Training Time, CIFAR10, $L_n=10, M = 0.2$



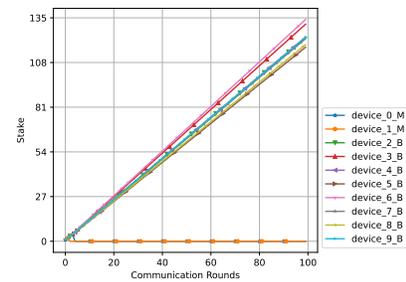
(g) Stake, Fashion-MNIST, $L_n=10, M = 0.1$



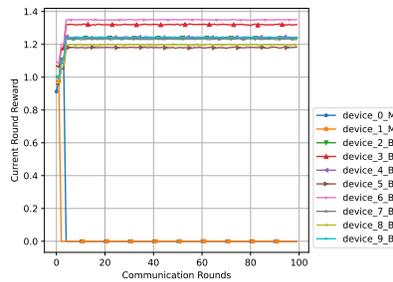
(h) Reward in One Round, Fashion-MNIST, $L_n=10, M = 0.1$



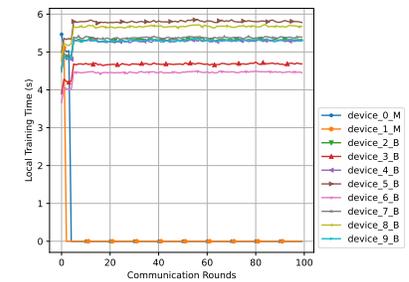
(i) Local Training Time, Fashion-MNIST, $L_n=10, M = 0.1$



(j) Stake, Fashion-MNIST, $L_n=10, M = 0.2$



(k) Reward in One Round, Fashion-MNIST, $L_n=10, M = 0.2$



(l) Local Training Time, Fashion-MNIST, $L_n=10, M = 0.2$

Fig. 6: Different devices' stake, reward and local training time of HAM algorithm under Malicious Attack for CIFAR10 and Fashion-MNIST classification tasks. "M" in the legend follows the L_n idx means this node is malicious node, and "B" in the legend means this node is benign node. "m" in the caption represents the malicious nodes ratio in local training nodes.

- tions on Big Data, 2022.
- [12] Q. Xie, S. Jiang, L. Jiang, Y. Huang, Z. Zhao, S. Khan, W. Dai, Z. Liu, and K. Wu, "Efficiency optimization techniques in privacy-preserving federated learning with homomorphic encryption: A brief survey," *IEEE Internet of Things Journal*, vol. 11, no. 14, pp. 24 569–24 580, 2024.
- [13] S. A. Rieyan, M. R. K. News, A. M. Rahman, S. A. Khan, S. T. J. Zaarif, M. G. R. Alam, M. M. Hassan, M. Ianni, and G. Fortino, "An advanced data fabric architecture leveraging homomorphic encryption and federated learning," *Information Fusion*, vol. 102, p. 102004, 2024.
- [14] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," pp. 493–506, 2020.
- [15] R.-Y. Huang, D. Samaraweera, and J. M. Chang, "Toward efficient homomorphic encryption-based federated learning: A magnitude-sensitivity approach," pp. 7810–7821, 2024.
- [16] S. Choi, D. Patel, D. Z. Tootaghaj, L. Cao, F. Ahmed, and P. Sharma, "Fednic: enhancing privacy-preserving federated learning via homomorphic encryption offload on smartnic," 2024.
- [17] K. Cheng, Z. Zou, Z. Wang, J. Yang, and S. Chen, "Qfl: Federated learning acceleration based on qat hardware accelerator," 2024.
- [18] V. Peluso, E. Malan, A. Calimera, and E. Macii, "Private tensor freezing for an efficient federated learning with homomorphic encryption," pp. 308–315, 2024.
- [19] Y. Cai, W. Ding, Y. Xiao, Z. Yan, X. Liu, and Z. Wan, "Secfed: A secure and efficient federated learning based on multi-key homomorphic encryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, pp. 3817–3833, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:266065945>
- [20] J. Shen, Y. Zhao, S. Huang, and Y. Ren, "Secure and flexible privacy-preserving federated learning based on multi-key fully homomorphic encryption," 2024.
- [21] X. Yang, Z. Liu, X. Tang, R. Lu, and B. Liu, "An efficient and multi-private key secure aggregation scheme for federated learning," pp. 1998–2011, 2024.
- [22] C. Wang, Z. Sun, and J. Lu, "A secure and efficient federated learning scheme based on homomorphic encryption and secret sharing," pp. 1170–1175, 2024.
- [23] C.-I. Fan, Y.-W. Hsu, C.-H. Shie, and Y.-F. Tseng, "Id-based multireceiver homomorphic proxy re-encryption in federated learning," *ACM Trans. Sen. Netw.*, vol. 18, no. 4, Nov. 2022. [Online]. Available: <https://doi.org/10.1145/3540199>
- [24] J. Wang, X. Lin, J. Wu, Q. Mao, B. Pei, J. Li, S. Guo, and B. Zhang, "Multi-level ace-based iot knowledge sharing for personalized privacy-preserving federated learning," in *2023 19th International Conference on Mobility, Sensing and Networking (MSN)*, 2023, pp. 843–848.
- [25] H. Lin, K. Kaur, X. Wang, G. Kaddoum, J. Hu, and M. M. Hassan, "Privacy-aware access control in iot-enabled healthcare: A federated deep learning approach," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 2893–2902, 2023.
- [26] L. Chen, J. Wang, L. Xiong, S. Zeng, and J. Geng, "A privacy-preserving federated learning framework based on homomorphic encryption," pp. 512–517, 2023.
- [27] R. Yang, T. Zhao, F. R. Yu, M. Li, D. Zhang, and X. Zhao, "Blockchain-based federated learning with enhanced privacy and security using homomorphic encryption and reputation," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 21 674–21 688, 2024.
- [28] L. Liu, Y. Hu, Y. Zhao, X. Zhang, Y. Ma, and G. Chang, "A novel federated learning system with privacy protection and blockchain consensus incentive mechanisms in cloud-edge collaboration scenarios," in *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2024, pp. 3111–3118.
- [29] B. Chen, H. Zeng, T. Xiang, S. Guo, T. Zhang, and Y. Liu, "Esbfl: Efficient and secure blockchain-based federated learning with fair payment," *IEEE Transactions on Big Data*, vol. 10, no. 6, pp. 761–774, 2024.
- [30] G. Guan, T. Zhi, H. Cai, Y. Cao, and H. Xie, "Hierarchical federated learning privacy protection framework with enhanced privacy and resistance to byzantine attacks," in *2024 IEEE 7th International Conference on Computer and Communication Engineering Technology (CCET)*, 2024, pp. 250–256.
- [31] K. Geng, L. Wang, Z. Zhang, Z. Lu, and M. Huang, "Ppce: Privacy-preserving contribution evaluation for fairness-aware federated learning," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 474–480.
- [32] Y. Li, W. Wang, Y. Wang, T. Xiangrong, P. Duan, and Z. Cai, "Personalized privacy protection incentive mechanism for mobile crowdsourcing based on homomorphic encryption and edge computing," in *2024 IEEE International Conference on Web Services (ICWS)*, 2024, pp. 1371–1376.
- [33] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.
- [34] A. Beimel, "Secret-sharing schemes: A survey," in *International conference on coding and cryptology*. Springer, 2011, pp. 11–46.
- [35] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Rfc3161: Internet x.509 public key infrastructure time-stamp protocol (tsp)," 2001.
- [36] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in cryptology—ASIACRYPT 2017: 23rd international conference on the theory and applications of cryptology and information security, Hong kong, China, December 3-7, 2017, proceedings, part i 23*. Springer, 2017, pp. 409–437.
- [37] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [38] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.