2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Coordinated Attacks through Graph Autoencoder in Federated Learning

Shuoxiang Wang[1], Bencan Gong[2] and Youlin Huang[3]

[1,3] Hubei Key Laboratory of Intelligent Vision Monitoring for Hydropower Engineering, Three Gorges University, Yichang, China
[2] College of Computer and Information Technology, Three Gorges University, Yichang, China
gonbc@sina.com

Abstract. Federated Learning (FL) facilitates collaborative model training across distributed clients while preserving data privacy through decentralized computation. Architectural limitations render FL susceptible to adversarial attacks. In current attack methodologies, the absence of collaboration among attackers renders them more susceptible to detection. This paper proposes CAGA (Collaborative Attack via Graph Autoencoder), an innovative model poisoning framework. Attackers exploit graph-structured correlations among benign local models to infer the training data characteristics of the target model and subsequently adversarially reconstruct these correlations, aiming to significantly degrade the performance of global FL model through crafted malicious updates. Unlike conventional attack methodologies, CAGA leverages pre-trusted malicious users embedded within benign user groups to execute dual-mode attacks—combining explicit adversarial actions with implicit exploitation of internal user privileges. The experimental results demonstrate that the proposed CAGA attack is highly aggressive and difficult to detect. The attack outperforms the existing GAE attack in terms of both aggressiveness and stealth.

Keywords: Federated Learning, Collaborative Attack, Graph Autoencoder, Dual-mode Attack.

1 INTRODUCTION

Federated Learning (FL) [1] is a decentralized paradigm that enables multiple clients to collaboratively train a global model without exposing raw data, thereby preserving privacy and complying with regulations such as GDPR (General Data Protection Regulation) [2]. In a typical FL workflow, a central server distributes the current global model to selected clients, who perform local training on their private datasets; the server then aggregates these updates—commonly via FedAvg—over multiple rounds until convergence [3]. Although FL mitigates data silos and accommodates heterogeneous, non-IID (Non-Independent and Identically Distributed) data distributions to support personalized learning [4], its decentralized architecture expands the attack surface. In data poisoning, adversaries manipulate local datasets (e.g., by flipping labels or injecting malicious samples) to degrade model accuracy or implant stealthy backdoors. In contrast, model poisoning directly alters gradients or parameters to evade anomaly detectors and undermine robust aggregation schemes [5]. More recently, data-agnostic attacks exploit structural correlations among intercepted benign updates to launch highly stealthy poisoning campaigns without accessing raw training data [6]. However, the attack is usually executed by a single malicious client, neglecting the potential for coordinated efforts among multiple malicious clients.

In this paper, we propose CAGA, a model-poisoning attack on federated learning that fully exploits both malicious clients and their colluding peers. CAGA acquires benign local models through interactions with seemingly benign yet malicious models. This attack leverages both the data of local benign models and the structural correlations between local and global benign models to maximize its effectiveness. The attacker first constructs a graph of structural relationships by extracting feature correlations from the benign local models and the underlying data characteristics that give rise to these models. Next, with the objective of maximizing federated learning training loss, the adversary reconstructs this graph using a graph autoencoder architecture, preserving essential structural features while introducing adversarial perturbations. The resulting adversarial graph is integrated with benign feature vectors to construct malicious local models. Prior to aggregation, these models engage in coordinated behavior by embedding adversarial information into ostensibly benign counterparts, thereby facilitating collaboration among malicious clients. This strategy leads to a substantial degradation in the performance of the aggregated global model. In wireless federated settings, the broadcast nature of wireless channels amplifies the impact of CAGA, enabling propagation across multiple clients and posing severe security threats. CAGA achieves strong attack potency while maintaining high stealth, subtly skewing the global model at each aggregation round and influencing even honest participants.

The contributions of this paper are summarized as follows:

1. Collaborative attack framework: We propose CAGA, a model poisoning attack against federated learning (FL). Unlike traditional attack methods, CAGA exploits trusted peers within the benign environment to coordinate collaborative attacks, thereby achieving a significantly enhanced attack effectiveness.
2. Structural correlation-based poisoned model generation: CAGA exploits genuine benign model updates and the intrinsic correlations among local models to construct adversarial models through sub-gradient descent optimization. This process manipulates inter-model relationships while preserving stealth during aggregation, ensuring that malicious updates remain undetected.
3. Empirical validation of attack impact: Experimental results demonstrate that CAGA significantly degrades FL model accuracy, bypasses the detection of poisoning defense mechanisms, and propagates adversarial effects among benign clients, leading to severe performance deterioration in FL systems.

2 RELATED WORK

2.1 Data Poisoning Attacks and Defenses

Data poisoning attacks manipulate clients' local training datasets to embed malicious behavior into the global model. Studies indicate that such attacks can significantly undermine model integrity, even when only a small fraction of clients are compromised [7]. Bagdasaryan et al. [8] first demonstrate backdoor poisoning in FL, where malicious clients inject trigger-specific samples during local training, causing the global model to behave normally on benign inputs but misclassify inputs containing secret triggers. Follow-up studies, such as "Towards Practical Backdoor Attacks on Federated Learning Systems", optimize trigger patterns to evade detection while maximizing backdoor effectiveness across diverse FL settings [9]. Zhang et al. [10] introduce a poisoning technique that induces catastrophic forgetting in global models to impair specific learned behaviors. Recent studies have also explored label-free backdoor attacks [11] and broader security challenges in vertical FL settings [12].

To mitigate data poisoning, researchers have proposed defense strategies such as robust loss functions and the use of external validation datasets. Wang et al. [13] propose a trimmed-mean aggregation rule that discards extreme update values before averaging, effectively reducing the impact of poisoned gradients under non-IID client distributions. Purohit et al. [14] develop DataDefense, which utilizes a small clean validation set to train a detector that assigns credibility scores to client updates, enabling weighted aggregation that suppresses poisoned contributions.

2.2 Model Poisoning Attacks and Defenses

In recent years, federated learning (FL) has become increasingly susceptible to model poisoning attacks. Xie et al. proposed Poisoned FL, a method that enhances attack effectiveness by maintaining consistency in malicious client updates across multiple training rounds, thereby bypassing various defenses and exposing persistent vulnerabilities in FL systems [2]. To counter these threats, Yaldiz [15] et al. introduced CosDefense, a defense mechanism that uses cosine similarity between the global model and individual client updates to identify and filter potentially malicious clients without requiring additional information, thus improving FL security. Wan et al. [16] examined the vulnerabilities of split federated learning (SFL) through the MISA attack, which simultaneously poisons both top and bottom models, significantly degrading global performance and challenging the assumed robustness of SFL architectures. In federated recommender systems, Yin et al. [17] developed PoisonFRS, an attack that leverages fake users to inject malicious updates, effectively promoting targeted items without access to genuine user data or server aggregation rules, increasing the attack's stealth and effectiveness. Additionally, Wang et al. [18] investigated evasion-based poisoning attacks on FL-based signal classifiers, demonstrating that even a single malicious client can introduce adversarial perturbations that substantially degrade global model performance, highlighting serious real-world security concerns for FL systems.

Traditional model poisoning attacks in FL often overlook structural correlations among local models, making them vulnerable to detection by recent graph-based defenses that utilize probabilistic modeling or structural anomaly detection. Furthermore, perturbations applied independently to individual model updates are often diluted during aggregation, resulting in detectable differences between malicious and benign updates. In contrast, this paper proposes a novel model poisoning attack CAGA that constructs a graph representing structural dependencies derived from benign local models and reconstructs it using a graph autoencoder to inject adversarial perturbations at the relational level. This approach preserves the statistical and structural integrity of poisoned models, allowing them to remain indistinguishable from benign updates while progressively degrading the global model's performance over multiple communication rounds.

3 SYSTEM MODEL

We investigate a realistic and impactful adversarial scenario where malicious clients participate in FL with the intention of poisoning the global model. Unlike conventional model poisoning attacks, this approach exploits malicious peers that have gained the trust of benign users. It leverages the broadcast nature of federated learning and the collaboration among colluding clients to extract and exploit information from benign participants. Specifically, the adversary can obtain benign local model updates by impersonating a legitimate participant. After collecting these model parameters, the attacker analyzes their statistical and structural properties, focusing on the correlation between benign local models and the aggregated global model. This analysis guides the generation of malicious model updates designed to disrupt the learning process while evading detection.

As illustrated in Fig. 1, the model poisoning attack unfolds as follows. In the t-th communication round, the malicious client obtains local model updates by leveraging interactions with seemingly benign peers during communication with the server. Based on this information, the attacker crafts poisoned updates that closely resemble the distribution of benign models, increasing their likelihood of being accepted during aggregation. These updates are then securely distributed to colluding peers, who embed the malicious payloads into their local models, resulting in poisoned updates. Finally, the attacker and its accomplices jointly upload these malicious updates to the central server.

An attacker detection model that utilizes Euclidean distance [19] is introduced as a statistical anomaly metric within a server-side detection framework. The key idea is to evaluate how far each client's local update deviates from the current global model. The assumption is that honest clients' updates are similar and hence lie in a compact region of the model parameter space, whereas malicious clients' updates tend to exhibit greater divergence.

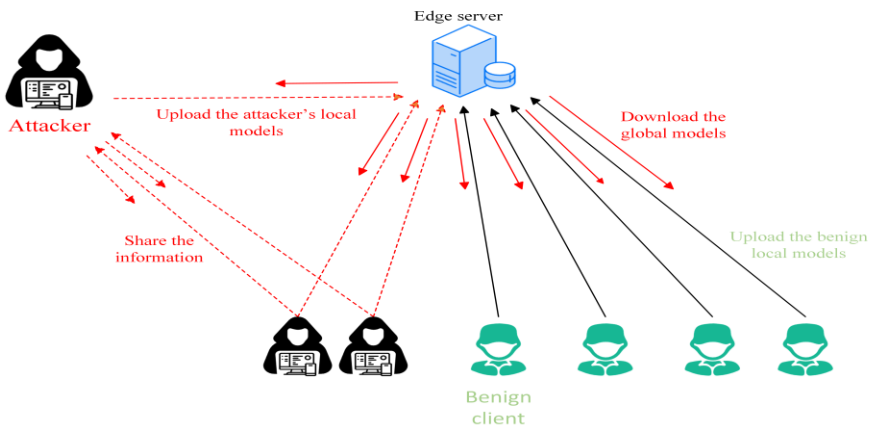

Edge server
Attacker
Upload the attacker's local models
Download the global models
Share the information
Upload the benign local models
Benign client

Fig. 1. System model.

We consider a centralized FL system comprising K benign and J malicious clients, all coordinating under a central server's orchestration. For each client $c \in \{1, \dots, K\}$, the server computes the Euclidean distance between the local model parameters θ_c and the global model parameters θ_g as follows:

$$D_c = \|\theta_c - \theta_g\|_2 \tag{1}$$

where $\|\cdot\|_2$ denotes the L2 norm (Euclidean distance). After computing these distances, the server compares each D_c with a pre-defined threshold τ. A model update is flagged as anomalous if:

$$D_c > \tau \tag{2}$$

Such updates are then either discarded or down-weighted during aggregation. This approach ensures that the influence of poisoned updates on the global model is minimized, thereby improving the robustness of FL systems.

Suppose that each benign client holds $S_c(t)$ training samples at t-th round. Denote the i-th sample's feature vector by $x_{c,i}(t)$ and its ground-truth label by $y_{c,i}(t)$, with $i \in \{1, \dots, S_c(t)\}$. We define the per-sample loss $\ell_{c,i}(\theta_c(t); x_{c,i}(t), y_{c,i}(t))$ to quantify the prediction error of the local model parameter $\theta_c(t)$ on $(x_{c,i}(t), y_{c,i}(t))$.

Each client minimizes a regularized empirical risk:

$$F_c(\theta_c(t)) = \frac{1}{S_c(t)} \sum_{i=1}^{S_c(t)} \ell_{c,i} + \alpha R(\theta_c(t)) \tag{3}$$

where $R(\cdot)$ is a regularizer and $\alpha \in [0, 1]$ its weight.

Clients perform one or more local gradient-based updates:

$$\theta_c(t+1) = \theta_c(t) - \eta \nabla F_c(\theta_c(t)) \tag{4}$$

where η is the learning rate.

After each local iteration T_{iter}, benign clients upload their updated model parameters to the central server, which aggregates them using a weighted average based on the clients' data sizes. The resulting global model is then distributed to all clients.

The cumulative amount of local training data declared by all participating clients to the server is expressed as follow:

$$S(t) = \sum_{k=1}^{K} S_k(t) + \sum_{j=1}^{J} S_j^a(t) \tag{5}$$

where $S_j^a(t), j \in [1, J]$ represents the number of local data samples claimed by the *j*-th malicious participants at *t*-th round.

Unaware of the adversarial activity, the server aggregates all submitted model updates via weighted averaging according to the reported data sizes, producing a compromised global model:

$$\theta_g^a(t) = \sum_{k=1}^{K} \frac{S_k(t)}{S(t)} \theta_k(t) + \sum_{j=1}^{J} \frac{S_j^a(t)}{S(t)} \theta_j^a(t) \tag{6}$$

where $\theta_j^a(t)$ denotes the model parameters of the *j*-th malicious user.

Subsequently, the server broadcasts the compromised global model $\theta_g^a(t)$ to all participating clients. This model is derived through the federated learning (FL) training process, which aims to optimize the global model by minimizing a global loss function calculated across all participating user devices, including the adversary's claimed dataset:

$$\min_{\theta_g^a(t)} F(\theta_g^a(t)) = \sum_{k=1}^{K} \frac{S_k(t)}{S(t)} F_k(\theta_g^a(t)) + \sum_{j=1}^{J} \frac{S_j^a(t)}{S(t)} F_j^a(\theta_g^a(t)) \tag{7}$$

where $F^a(\cdot)$ denotes the local loss function claimed by the attacker, which is stated to comply with (3).

To achieve its malicious intent, the attacker seeks to maximize $F(\theta_g^a(t))$ while ensuring that the crafted adversarial model remains undetected by the server. In the federated learning training process, the server typically assesses the similarity among local models at each communication round or periodically after several rounds, thereby filtering out potentially malicious models. The attacker's optimization problem can be formally expressed as follows:

$$\underset{\theta_j^a(t), \beta_{k,c}(t)}{max} F(\theta_g^a(t)) \tag{8a}$$

$$s.t. \, d(\theta_j^a(t), \theta_g^a(t)) \le d_\tau \tag{8b}$$

$$\sum_{k=1}^{K} \beta_{k,j}(t) \, d(\theta_c(t), \theta_g(t)) \le D_\tau \tag{8c}$$

$$\beta_{k,j}(t) \in \{0, 1\} \tag{8d}$$

where $d(\cdot,\cdot)$ is computed based on (1), and d_τ is a predetermined threshold. Constraint (8b) requires the adversary's malicious local model to remain close to the global model in Euclidean space, thereby maintaining the attack's subtlety and evading detection.

Constraint (8c) enforces that the Euclidean distance between each selected surrogate model and the aggregated malicious model does not exceed a predefined threshold, ensuring consistency among malicious updates. Collectively, these constraints regulate spatial relationships between models, enabling the attack to bypass detection mechanisms such as Krum and Multi-Krum, which utilize distance-based anomaly detection to identify and exclude outliers.

4 DESIGN DETAILS

In this section, we present the CAGA attack and its operational workflow. The adversary trains a malicious local model on private data, optimizing both task-specific performance and adversarial objectives. Leveraging graph autoencoding, the attacker crafts poisoned updates that subtly manipulate gradient distributions to evade detection. These updates are disseminated among colluding peers, whose coordinated efforts amplify the attack's impact during global aggregation.

4.1 Processing Procedure of the CAGA

The CAGA optimization problem in (8) is a non-convex combinatorial challenge, difficult to solve with standard methods. We address this by decomposing it into two subproblems—adversarial model generation and bandwidth allocation—using the Lagrangian dual method [20]. We propose an iterative framework that updates adversarial local models via contrastive graph autoencoding and sub-gradient descent, as shown in Fig. 2.

The proposed CAGA attack exploits the correlations learned among model parameters $\theta_c(t)$ during federated learning to generate the adversarial parameter $\theta_j^a(t)$. A graph autoencoder is utilized to jointly learn the adjacency matrix and the feature matrix, producing a matrix Z, which represents the encoder's output. The decoder then reconstructs the correlation matrix by multiplying Z with its transpose and applying a sigmoid activation function. The CAGA attack iteratively updates the adversarial parameter $\theta_j^a(t)$ to maximize the reconstruction loss, thereby enhancing the effectiveness of the attack.

We denote λ and ρ as the dual variables associated with the Lagrangian function. The Lagrangian corresponding to (8) is defined as follows:

$$L(\beta_{k,j}(t), \lambda, \rho) = F\left(\theta_g^a(t)\right) + \lambda\left(d_\tau - d\left(\theta_j^a(t), \theta_g^a(t)\right)\right)$$
$$+\rho(D_\tau - \textstyle\sum_{k=1}^{K} \beta_{k,j}(t)\, d(\theta_c(t), \theta_g(t))) \tag{9}$$

The dual function associated with the Lagrangian is

$$g(\lambda, \rho) = \max_{\theta_j^a(t), \beta_{k,j}(t)} L(\beta_{k,j}(t), \lambda, \rho) \tag{10}$$

The dual problem associated with the primal formulation in (5) is expressed as follows:

$$\min_{\lambda,\rho} g(\lambda,\rho) \tag{11}$$

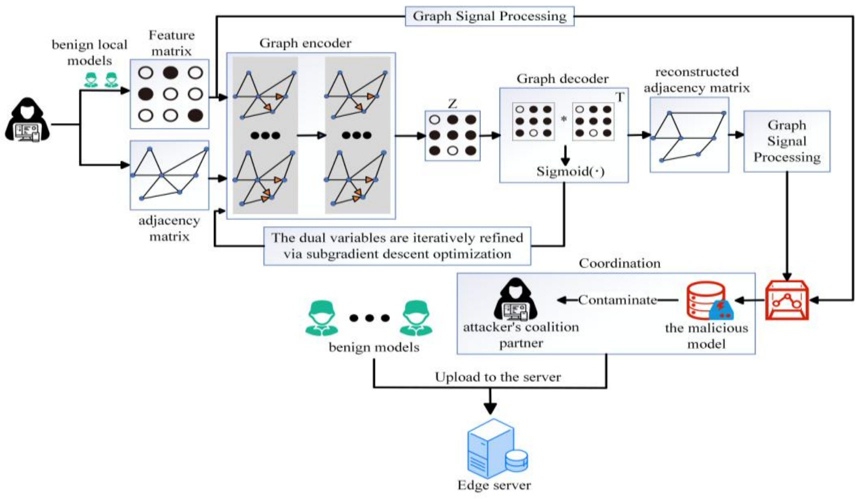


Fig. 2. Processing procedure of the CAGA.

At the *t*-th communication round, given the dual variables $\lambda = \lambda(t)$ and $\rho = \rho(t)$, the primary variable $\beta_{k,j}(t)$, which governs bandwidth allocation, is optimized by solving the following problem:

$$\beta_{k,j}(t)^* = \underset{\beta_{k,j}(t)}{argmin}\{\textstyle\sum_{k=1}^{K} \beta_{k,j}(t)\, d(\theta_c(t), \theta_g(t))\} \tag{12}$$

where the variable $\beta_{k,j}(t)$ satisfies constraint (8d). This optimization problem is a standard 0/1 knapsack problem and can be efficiently solved using dynamic programming.

Given the adversarial model $\theta_g(t)$ and the optimized bandwidth-selection variable $\beta_{k,j}(t)^*$, the dual variables $\lambda(t)$ and $\rho(t)$ are updated using sub-gradient descent to solve the dual problem in (10). Specifically, at each iteration:

$$\lambda(t+1) = [\lambda(t) - \varepsilon(d(\theta_j^a(t), \theta_g^a(t)) - d_\tau)]^+ \tag{13a}$$

$$\rho(t+1) = [\rho(t) - \varepsilon(\textstyle\sum_{k=1}^{K} \beta_{k,j}(t)^*\, d(\theta_c(t), \theta_g(t)) - D_\tau)]^+ \tag{13b}$$

where ε denotes the step size and $[\cdot]^+ = max(0,\cdot)$. The initial values of the dual variables are set such that $\lambda(1) \geq 0$ and $\rho(1) \geq 0$ to ensure the convergence of the updates in (12).

The objective of the proposed CAGA attack is to maximize the Lagrangian function (9). Given the optimized variable $\beta_{k,j}(t)^*$, along with the dual variables $\lambda(t)$ and $\rho(t)$, we proceed to optimize $\theta_j^a(t)$ by solving the following problem:

$$\theta_j^a(t)^* = \underset{\theta_j^a(t)}{argmax}\{F(\theta_g^a(t)) - \lambda(t)d(\theta_j^a(t), \theta_g^a(t)) - \rho(t)\sum_{k=1}^{K}\beta_{k,j}(t)^*\, d(\theta_c(t), \theta_g(t))\} \quad (14)$$

Specifically, the proposed CAGA attack allows an attacker can observe local model parameters from benign clients and infer the intrinsic correlations among them. These correlations are modeled as a graph, which is then reconstructively manipulated using CAGA to generate the malicious local model $\theta_j^a(t)$. The CAGA decoder is designed to replicate the original correlation patterns, thereby satisfying constraints (8b) - (8d) and minimizing structural dissimilarities between $\theta_c(t)$ and $\theta_j^a(t)$. This strategy enables the attacker to maximize the objective in (10) while simultaneously hindering the convergence of the global model $\theta_g^a(t)$. The rationale behind the design is detailed below.

1) Construction of correlation graph and feature extraction: In CAGA, the intrinsic correlations among the parameters of federated local models $\theta_c(t)$ where $c \in [1, K]$ at communication round *t* are represented by a graph $G = (V, E, X)$ (see Fig. 2). The vertex set V consists of S feature nodes, each corresponding to a dimension of the model. The edge set E encodes the pairwise similarities between parameters, while the feature matrix $X(t) = [\theta_1(t), \cdots, \theta_K(t)]^T \in R^{K\times S}$ collects all benign local models. To quantify correlations between parameters, let $\theta^s(t) \in R^{K\times 1}$ denote the *s*-th column of $X(t)$. The *cosine similarity* between $\theta_q(t)$ and $\theta_m(t)$ is computed as

$$\delta_{q,m}(t) = \frac{(\theta^q(t))^T\theta^m(t)}{\|\theta^q(t)\|\cdot\|\theta^m(t)\|} \quad (15)$$

where $q, m \in [1, S]$. Subsequently, we define the adjacency matrix $A(t) = [\delta_{q,m}(t)] \in R^{S\times S}$, which encodes the pairwise similarities among model parameters. This matrix is used as one of the inputs to the encoder of the CAGA model at the attacker's side. Leveraging $A(t)$, the attacker reconstructs the topological structure of the correlation graph G. The feature matrix $X(t)$, which contains all benign local models, is provided as the other input to the encoder.

2) Encoder design procedure: The encoder aims to project the graph G into a lower-dimensional latent space. To accomplish this, we implement a two-layer GCN (Graph Convolutional Network) that effectively learns representations capturing the intrinsic structural features of G. The resulting low-dimensional embedding is subsequently fed into the decoder, which reconstructs the original graph from this compressed representation, ultimately producing the malicious local model. For brevity and clarity, we omit the communication round index *t* in the subsequent exposition. The encoder is formally defined as follows:

$$Z^{(1)} = f_1(X, A, |W_0) \quad (16a)$$

$$Z^{(2)} = f_2\big(Z^{(1)}, A, \big|W_1\big) \quad (16b)$$

where $f_1(\cdot)$ denotes the Rectified Linear Unit (ReLU) activation function employed in the first layer, and $f_2(\cdot)$ denotes the linear activation function used in the second layer.

The parameter set W_l corresponds to the learnable weights associated with the l-th layer of the neural network.

To improve experimental feasibility and obtain the latent representation of vertices Z in graph G, we approximate the true posterior distribution with a Gaussian distribution $N(\cdot)$ [21]. The feature matrix X and adjacency matrix A serve as inputs to the encoder, which produces a low-dimensional embedding of the graph G via the designed two-layer Graph Convolutional Network.Consequently, we obtain

$$q(Z|A,X) = \prod_{s=1}^{S} q(z_s|A,X) \tag{17a}$$

$$q(z_s|A,X) = N(z_s|\mu_s, diag(\sigma^2)) \tag{17b}$$

where $\mu = Z^{(2)}$ denotes the matrix of mean vectors, and $z_s \in R^{S\times 1}$ denotes the latent representation corresponding to the s-th node, i.e., the s-th column of the matrix Z, with each row μ_s representing the mean of the latent representation for node s. Similarly, the logarithm of the variance is computed as $log\sigma = f_2\left(Z^{(1)}, A\middle|W_1\right)$, where the weights W_1 are learned using the same parameters W_0 shared with the first layer of the GCN.

Let $E \in R^{S\times S}$ be the identity matrix. We define the normalized adjacency matrix as $\tilde{A} = A + E$, where $A_{q,m}$ denotes the (q, m)-th element. The corresponding degree matrix D is diagonal, with each diagonal entry defined as $D_{q,q} = \sum_{m=1}^{S} \tilde{A}_{q,m}$. Each layer of the GCN is then computed as:

$$f_G\left(Z^{(l-1)}, A\middle|W_l\right) = f(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}Z^{(l-1)}W_l) \tag{18}$$

where $f(\cdot)$ corresponds to a nonlinear activation function like ReLU.

3) Decoder design procedure: The output matrix Z from the encoder is utilized as the input to the decoder, which is designed to reconstruct the original adjacency matrix A. Specifically, the decoder computes the inner product between two latent variables, followed by the application of a sigmoid activation function to estimate the probability of a link between each pair of vertices, as shown in the following equation:

$$p\left(\hat{A}\middle|Z\right) = \sum_{q=1}^{S}\sum_{m=1}^{S} p\left(\hat{\delta}_{q,m}\middle|z_q, z_m\right) \tag{19a}$$

$$p\left(\hat{\delta}_{q,m} = 1\middle|z_q, z_m\right) = sigmoid(z_q^T, z_m) \tag{19b}$$

where $\hat{A} = [\hat{\delta}_{q,m}] \in R^{S\times S}$ denotes the reconstructed adjacency matrix A and $sigmoid(\cdot)$ represents the logistic sigmoid function, defined as $sigmoid(x) = \frac{1}{1+e^{-x}}$. An increase in $sigmoid(z_q^T, z_m)$ suggests a greater probability of a connection between the two latent variables.

As a result of the above operations, the decoder produces the reconstructed adjacency matrix $\hat{A}$. Furthermore, a reconstruction loss function ϑ_{loss} is introduced to quantify the discrepancy between $\hat{A}$ and the original adjacency matrix A. According to (17a) and (19a), the loss is defined as follows:

$$\vartheta_{loss} = E_{k(Z|A,X)}[logp\left(\hat{A}\middle|Z\right)] - \varphi[k(Z|A,X)|p(Z)] \tag{20}$$

where $p(Z) = \prod_s p(z_s) = \prod_s N(z_s|0, E)$ serves as a Gaussian prior, and $\varphi[k(Z|A, X)|p(Z)]$ denotes the Kullback-Leibler (KL) divergence between the approximate posterior $k(Z|A, X)$ and $p(Z)$.

4) Constructing malicious models and performing model poisoning: As illustrated in Fig. 2, the goal of graph signal processing is to extract and verify the features of matrix X. The design process begins with the construction of a Laplacian matrix derived from the adjacency matrix of a benign model, i.e., A, as formulated below:

$$L = diag(A) - A \tag{21}$$

Subsequently, singular value decomposition (SVD) is applied to the matrix L, i.e., $L = U_1 \Sigma_1 V_1^T$, resulting in a complex unitary matrix that serves as the basis for the Graph Fourier Transform (GFT). This basis enables the transformation of graph data into the spectral domain. The associated diagonal matrix Σ contains the eigenvalues of L along its main diagonal, which indicate the magnitude of each corresponding frequency component.

Through the described process, the attacker employs matrix U together with the data feature matrix X, representing the local models, to construct matrix I. Matrix I encapsulates the spectral-domain data features of all benign local models, thereby shifting the focus from inter-model correlations to the intrinsic data characteristics that authenticate each local model. Matrix I is constructed as follows：

$$I = U^{-1} X \tag{22}$$

The attacker can similarly apply this process to the adjacency matrix predicted by the CAGA model. Consequently, the following expression can be derived:

$$\hat{L} = diag(\hat{A}) - \hat{A} \tag{23}$$

We perform singular value decomposition (SVD) on matrix $\hat{\mathrm{L}}$ to obtain an alternative graph Fourier transform (GFT) basis. i.e., $\hat{L} = U_2 \Sigma_2 V_2^T$, Utilizing this GFT basis along with the matrix I derived from (22), we construct malicious local models $\hat{X}$:

$$\hat{X} = U_2 I \tag{24}$$

where $\hat{X} \in R^{K \times S}$. The vector $\theta_j^a(t)$ in $\hat{X}$ is considered a representation of a malicious local model.

We define a pollution ratio $\propto$ to regulate the proportion of malicious data generated by the attacker that is shared with its seemingly benign peers. These peers remain non-malicious until the attacker initiates the attack. The sharing mechanism is formalized as follows:

$$\theta_j^{new} = \begin{cases} \theta_j^{a,i}, & if\ i \le \propto S \\ \theta_{c,i}, & if\ i > \propto S \end{cases} for\ i = 1, \cdots, S \tag{25}$$

where θ_j^{new} denotes the model parameters of the adversary's accomplice generated through the sharing process, $\theta_j^{a,i}$ represents the first *i* data features of the model parameters θ_j^a generated by the adversary, and $\theta_{c,i}$ indicates the data features from the *i*-th to the last of the benign model parameters θ_c.Following the sharing operation, malicious clients, seemingly benign malicious clients, and benign clients jointly upload their model parameters to the server for global aggregation.

4.2 Algorithm Design of the CAGA

Algorithm 1 outlines the training procedure of the proposed CAGA attack.

Algorithm 1: The proposed CAGA attack algorithm.

```
1: Initialize: $G = (V, E, X), T_{iter}, K, \theta_j^a(t), \theta_g^a(t), \lambda(1) \geq 0, \rho(1) \geq 0.$
   % Adversarial FL:
2: for each round $t = 1,2,3,\cdots$ do
3:    for each local iteration $t_{iter} = 1,\cdots, T_{iter}$ do
4:       for each benign client $c = 1,\cdots, K$ do
5:          Train the benign local model $\theta_c(t)$.
6:       end for
7:    end for
8:    Seemingly benign malicious clients initially share their training data with the
      malicious clients. Subsequently, all user devices upload their local models
      $\theta_c(t), c = 1.\cdots, K$ to the server.
9:    The attacker conducts the CAGA attack using the acquired feature matrix $X$ and
      derives the malicious model through the following procedure.
10:      The adjacency matrix $A(t) = [\delta_{q,m}(t)] \in R^{S\times S}$ is constructed based on
         (15), and both A and the feature matrix $X$ are used as inputs to the encoder
         of the CAGA model.
11:      The CAGA model is trained to maximize the reconstruction loss
         $L(\beta_{k,j}(t), \lambda(t), \rho(t)) - \vartheta_{loss}$, thereby producing the predicted adjacency
         matrix $\hat{A}$.
12:      The matrix I is constructed according to (21) and (22), while the matrix X
         is obtained from (23) and (24). The malicious model parameters $\theta_j^a(t)$ are
         then derived accordingly.
13:   Update $\lambda(t), \rho(t)$, according to (13a) and (13b).
14:   The attacker shares its training data with the seemingly benign adversarial cli-
      ents.
15:   The seemingly benign adversarial clients update their local models using (25).
16:   All clients upload their local model parameters to the server, which aggregates
      them to generate the contaminated global model $\theta_g^a(t)$ and then broadcasts it
      to all clients.
17:   Each client updates its local model based on the contaminated global model
      $\theta_g^a(t)$. i.e., $\theta_c(t) \leftarrow \theta_g^a(t), \forall i.$
18: end for
```

Prior to launching the attack, all benign users (including the seemingly benign adversarial users) complete one training round, during which the seemingly benign adversarial users refrain from any malicious activity, as detailed in steps 3 to 7. Upon completion, the seemingly benign adversarial users share their training data with the malicious users and upload this data alongside that of the benign users to the server.

Similarly, when the server broadcasts the global model, malicious users can easily access the global model parameters due to the presence of seemingly benign adversarial clients. The training objective of the CAGA model is to fully leverage the adjacency matrix A and feature matrix X, employing the Lagrangian method to maximize the model poisoning attack problem described in (8), as detailed in steps 10 to 12. The sub-gradient descent method is used to update the dual variables $\lambda(t)$ and $\rho(t)$, as described in step 13.

Subsequently, the attacker shares the generated malicious model with the seemingly benign adversarial clients, who incorporate its parameters into their local models at a controlled ratio $\propto$. Because the malicious model exploits feature of benign models, the server is unable to detect the attacker. Upon completion of the sharing process, all users upload their training data to the server, resulting in a contaminated global model $\theta_g^a(t)$. The server broadcasts the global model $\theta_g^a(t)$ to all users, who subsequently update their respective local models based on the received global parameters.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Parameters Set

In our experimental evaluation, we assess the effectiveness of the proposed CAGA attack on three widely used image classification benchmarks—MNIST, Fashion-MNIST, and CIFAR-10—and compare its performance with the existing data-agnostic GAE-based poisoning attack under a unified federated learning framework. Table 1 presents all hyperparameter settings used in our experiments.

Table 1. Experimental parameters

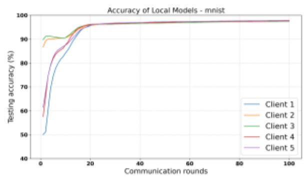
Parameter	Value	Parameter	Value
K: Number of benign devices	10-25	1st hidden layer size of GCN	32
J: Number of attackers	2-10	2st hidden layer size of GCN	16
T_{iter}: Number of local iterations	5	Learning rate	0.01,0.001
Communication rounds	100	batch size of the SVM	30

To examine the impact of network scale and attacker ratio, the number of benign clients (K) is set to 10, 15, 20, and 25, while the number of malicious clients (J) is set to 2, 4, 6, 8, and 10. Each communication round consists of five local iterations per client, and the overall training process includes 100 rounds to capture the cumulative effects of poisoning over time. Both CAGA and the baseline GAE attack [22] adopt a two-layer GCN at the server. The first hidden layer comprises 32 units, and the second comprises 16 units. We set the learning rate of the SVM (Support Vector Machine) to

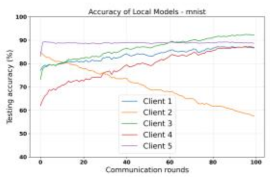
0.001 and that of CAGA to 0.01. The batch size for the SVM used in credibility assessment is fixed at 30 to ensure a balance between evaluation efficiency and classification performance.

5.2 Performance Analysis of CAGA

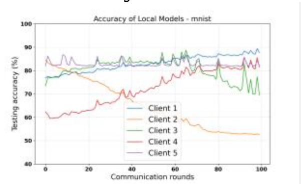
To assess the impact of the proposed CAGA attack on local clients, we configured 15 benign users and 6 malicious users, and visualized the testing accuracy of five local models over 100 FL communication rounds on the MNIST, Fashion-MNIST, and CIFAR-10 datasets under three settings: no attack, the conventional GAE attack, and the proposed CAGA attack, as illustrated in Fig. 3. When no attack is present, FL enables rapid convergence of local testing accuracy across all datasets, as shown in Fig. 3(a), (d), and (g). In contrast, Fig. 3(c), (f), and (i) indicate that the proposed CAGA attack effectively suppresses the convergence of local model accuracy.



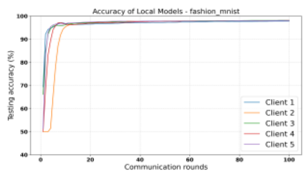
(a) No attack with MNIST.



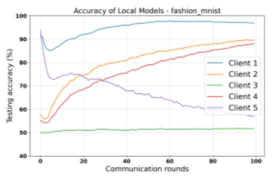
(b) GAE with MNIST.



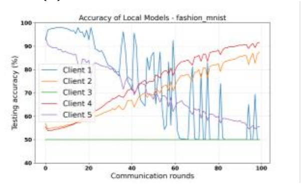
(c) CAGA with MNIST.



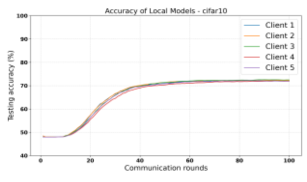
(d) No attack with FashionMNIST.



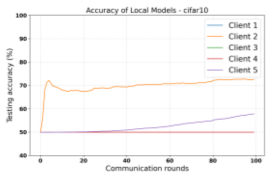
(e) GAE with FashionMNIST.



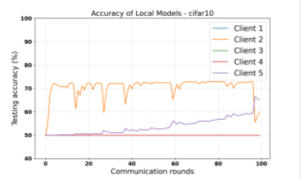
(f) CAGA with FashionMNIST.



(g) No attack with CIFAR-10.



(h) GAE with CIFAR-10.



(i) CAGA with CIFAR-10.

Fig. 3. Testing accuracy variation chart of the local client

Given 100 communication rounds and five benign clients, we evaluate the testing accuracy of their local models under three conditions: no attack, the conventional GAE attack, and the proposed CAGA attack. The experiments are conducted on three benchmark datasets: MNIST, Fashion-MNIST, and CIFAR-10.

Taking the MNIST dataset as an example, we observed that the test accuracy of Client 2 decreased to 55% at the 75th round under the CAGA attack, whereas it decreased to 58% at the 100th round under the GAE attack. The test accuracy of Client 3 began to decline after 60 rounds under the CAGA attack, while it continued to improve under the GAE attack. Client 4’s accuracy increased more slowly with additional training rounds under the CAGA attack than under the GAE attack, suggesting that the CAGA attack has a stronger ability to suppress accuracy convergence. The average test accuracy of Client 5 over 100 rounds decreased by 6% under the CAGA attack compared to the GAE attack. Although Client 1’s test accuracy slightly increased under the CAGA attack, the other clients were adversely affected. These results demonstrate that the proposed CAGA attack is more effective in degrading local model performance than the traditional GAE attack.

To further demonstrate the high level of stealthiness of our proposed CAGA attack, we used the attacker detection model introduced in Section 3.1. We set up 15 benign clients and 6 malicious clients. We randomly selected 6 benign clients to calculate the Euclidean distance between their local models and the global model, and then computed the average distance. Subsequently, we plotted the Euclidean distances between benign user models and the global model under attack scenarios on the MNIST, FashionMNIST, and CIFAR-10 datasets, as well as the Euclidean distances between malicious models and the global model under both GAE and CAGA attacks, as shown in Fig. 4, 5, and 6.

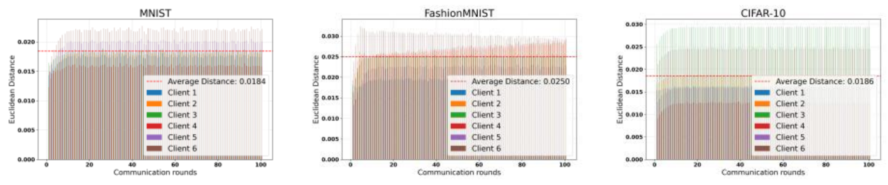


Fig. 4. Euclidean distance between benign and global models across three datasets.

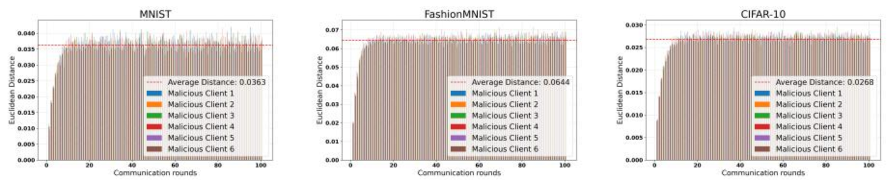


Fig. 5. Euclidean distance between malicious and global models under the GAE attack.

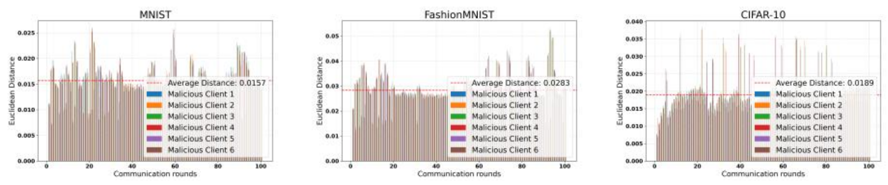
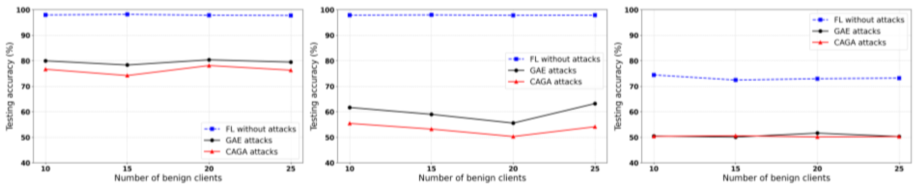


Fig. 6. Euclidean distance between malicious and global models under the CAGA attack.

Specifically, the Euclidean distance between malicious models and the global model is typically greater than that between benign models and the global model. This difference makes the method effective for identifying attackers.

Fig. 4 presents the Euclidean distances between benign models and the global model across various datasets and also shows the average Euclidean distance of the selected models. Under the proposed CAGA attack, the Euclidean distances between malicious models and the global model are comparable to or even smaller than those between benign models and the global model, as depicted in Fig. 6. Compared to the GAE attack, the Euclidean distances between malicious models and the global model are smaller under the CAGA attack. For instance, using the MNIST dataset, the average Euclidean distance between malicious models and the global model is 0.0157 under the CAGA attack, whereas it is 0.0363 under the GAE attack. This value is significantly larger than the Euclidean distance of 0.0184 between benign models and the global model. Therefore, the CAGA attack demonstrates strong stealthiness.



(a) MNIST (b) FashionMNIST (c) CIFAR-10

Fig. 7. Comparison of global accuracy under different attacks

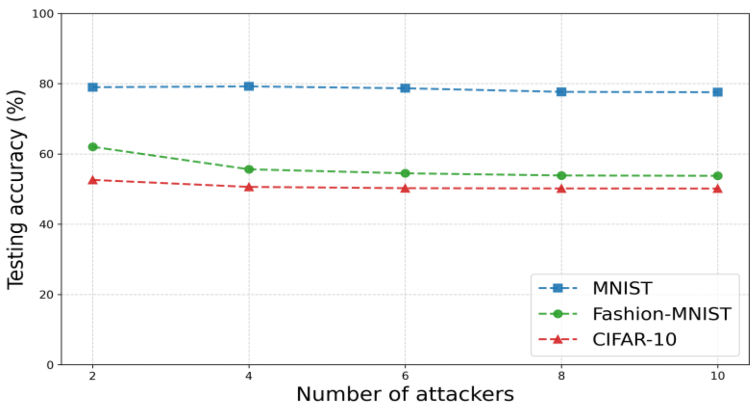


Fig. 8. Comparison of global accuracy under different numbers of attackers

Fig. 7 illustrates the global accuracy under both the GAE and CAGA attacks. The ratio of attackers to benign users was set at 2:5. To verify the effectiveness of these attacks, we also visualized the global model's accuracy in the absence of any attack. The experimental results demonstrate that, on the MNIST and FashionMNIST datasets, the average global accuracy is lower under the CAGA attack than under the GAE attack. On the CIFAR dataset, the effectiveness of the CAGA attack is comparable to that of the GAE attack. However, as shown in Fig. 6, even when the attack strength is similar, the CAGA attack demonstrates a higher level of stealthiness.

Fig. 8 presents the variation in global model accuracy as the number of malicious users increases, with the number of benign users held constant at 25. The experimental results indicate that even a small number of attackers can significantly degrade model performance. Interestingly, when the number of attackers is low, the attack can still be nearly as effective as with more attackers. However, the Euclidean distance between their local models and the global model is relatively large. As a result, when there are only a few malicious participants, the CAGA attack is not adopted. This is because the CAGA method relies on collaboration among attackers, and with too few participants, the collaborative capability is diminished, leading to reduced stealthiness.

6 CONCLUSION

In this paper, we propose a collaborative model poisoning attack against federated learning, named CAGA. This approach leverages the parameters of benign local models and their correlations with the global model to craft malicious updates. These poisoned updates are propagated among benign clients in a viral manner, substantially degrading the accuracy of the global model. We conduct extensive experiments on three benchmark datasets: MNIST, Fashion-MNIST, and CIFAR-10. The experimental results demonstrate that CAGA outperforms conventional GAE-based methods in terms of both attack effectiveness and stealthiness.

References

1. McMahan H. B., Moore E., Ramage D., Hampson S., Arcas B. A.: Communication-efficient learning of deep networks from decentralized data. In: International Conference on Artificial Intelligence and Statistics, pp.1273-1282 (2017)
2. Xie Y., Fang M., Gong N.Z.: Model poisoning attacks to federated learning via multi-round consistency. In: arXiv preprint arXiv:2404.15611 (2024)
3. Banabilah S., Aloqaily M., Alsayed E., Malik N., Jararweh Y.: Federated learning review: Fundamentals, enabling technologies, and future applications. Information processing & management **59**(6), 1-18 (2022)
4. Tan A.Z., Yu H., Cui L., Zhang X., Wang Y., Liu J.: Towards personalized federated learning. In: IEEE Transactions on Neural Networks and Learning Systems, **34**(12), pp.9587–9603 (2022)
5. Feng C., Li Y., Gao Y., Celdrán A. H., von der Assen J., Bovet G., Stiller B.: DMPA: Model poisoning attacks on decentralized federated learning for model differences. In: arXiv preprint arXiv:2502.04771 (2025)
6. Krauß T., König J., Dmitrienko A., Kanzow C.: Automatic adversarial adaption for stealthy poisoning attacks in federated learning. In: NDSS Symposium (2024)
7. Sagar S., Li C. S., Loke S. W., Choi J.: Poisoning Attacks and Defenses in Federated Learning: A Survey. In: arXiv preprint arXiv:2301.05795 (2023)
8. Bagdasaryan E., Veit A., Hua Y., Estrin D., Shmatikov V.: How to backdoor federated learning. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR **108**:2938–2948 (2020)

9. Shi C., Ji S., Pan X., Zhang X., Zhang M., Yang M., Zhou J., Yin J., Wang T.: Towards practical backdoor attacks on federated learning systems. In: IEEE Transactions on Dependable and Secure Computing, **21**(6), 5431–5447 (2024)
10. Zhang C., Zhou B., He Z., Liu Y., Wang J., Chen X.: Oblivion: Poisoning Federated Learning by Inducing Catastrophic Forgetting. In: IEEE INFOCOM 2023 – IEEE Conference on Computer Communications, pp.1–10 (2023)
11. Shen W., Huang W., Wan G., Ye M.: Label-free backdoor attacks in vertical federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, **39**(19), pp.20389–20397 (2025)
12. Ye M., Shen W., Du B., Zhang L., Li Q., Chen Y.: Vertical federated learning for effectiveness, security, applicability: A survey. In: ACM Computing Surveys, **57**(9), pp.1–32 (2025)
13. Wang T., Zheng Z., Lin F.: Federated learning framework based on trimmed mean aggregation rules. In: Expert Systems with Applications **232**, 121958 (2025)
14. Purohit K., Das S., Bhattacharya S., Rana S.: A data-driven defense against edge-case model poisoning attacks on federated learning. In: Proceedings of the 27th European Conference on Artificial Intelligence (ECAI), pp. 1234–1242 (2024)
15. Yaldiz D.N., Zhang T., Avestimehr S.: Secure federated learning against model poisoning attacks via client filtering. In: arXiv preprint arXiv:2304.00160 (2023)
16. Wan W., Ning Y., Hu S., Xue L., Li M., Zhang L.Y., Jin H.: MISA: Unveiling the vulnerabilities in split federated learning. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6435–6439 (2024)
17. Yin M., Xu Y., Fang M., Gong N.Z.: Poisoning federated recommender systems with fake users. In: Proceedings of the ACM Web Conference 2024, pp. 3555–3565 (2024)
18. Wang S., Sahay R., Brinton C.G.: How potent are evasion attacks for poisoning federated learning-based signal classifiers? In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 2376–2381 (2023)
19. Shi Z., Ding X., Li F., Chen Y., Li C.: Mitigation of a poisoning attack in federated learning by using historical distance detection. In: Annals of Telecommunications **78**, 135–147 (2023)
20. Palomar D.P., Chiang M.: A tutorial on decomposition methods for network utility maximization. In: IEEE Journal on Selected Areas in Communications, **24**(8), pp.1439–1451 (2006)
21. Li K., Yuan X., Zheng J., Ni W., Dressler F., Jamalipour A.: Leverage Variational Graph Representation for Model Poisoning on Federated Learning. In: IEEE Transactions on Neural Networks and Learning Systems, **36**(1), pp. 116–128 (2025)
22. Li K., Zheng J., Yuan X., Ni W., Akan O. B., Poor H. V.: Data-Agnostic Model Poisoning Against Federated Learning: A Graph Autoencoder Approach. In: IEEE Transactions on Information Forensics and Security **19**, 3465–3480 (2024)