SecureGraphFL: A Privacy-Preserving and Attack-Resilient Federated Learning Framework for Traffic Prediction

Guihong Chen, *Member, IEEE*, Mingyong Tong, *Student Member, IEEE*, Jiao Yin, *Member, IEEE*, Maojie Wang, *Student Member, IEEE*, Jinli Cao, *Member, IEEE*, Hua Wang, *Senior Member, IEEE*

***Abstract*—Traffic prediction is a critical research topic with significant transportation management and planning implications. Graph Neural Networks (GNNs) have gained widespread use in traffic prediction due to their ability to capture relationships between different traffic sensors effectively. However, traditional centralized learning approaches struggle to meet the increasing demand for user data privacy. While federated learning (FL) addresses these privacy concerns, it introduces new challenges, particularly when one or more clients are compromised by malicious attacks or single-point failures, leading to significant degradation in global model performance. To address these issues, we propose SecureGraphFL, a federated learning framework based on graph convolutional networks (GCNs) and actor-critic networks that ensures security while maintaining high predictive accuracy. SecureGraphFL introduces three key innovations: (1) incorporating a parallel speed difference prediction task to enhance prediction performance, (2) utilizing a cosine similarity-based graph construction method to more effectively capture patterns from similar traffic sensors, and (3) designing an actor-critic network-based client selection mechanism to exclude malicious clients, ensuring the stability of the global model. Experiments on two open-source datasets, PEMS-D7 and METR-LA, demonstrate that in the absence of malicious or failed clients, SecureGraphFL matches state-of-the-art centralized methods and significantly outperforms traditional FL algorithms like FedAVG and FedATT. Moreover, in scenarios involving malicious clients, our client selection mechanism effectively mitigates their impact, maintaining performance levels comparable to those without any malicious activity. The code for this paper is available at: https://github.com/cry-C/SecureGraphFL.**

***Index Terms*—Federated learning, traffic prediction, graph convolutional networks, data privacy, attack resilience.**

I. Introduction

The work is supported in part by Key Research Projects of Universities in Guangdong Province under Grant 2022ZDZX1011, Guangdong Provincial Natural Science Fund Project under Grant 2023A1515011084 and Doctoral Program Construction Unit Research Capability Enhancement Project at Guangdong Polytechnic Normal University under Grant 22GPNUZDJS27. (*Corresponding author: Jiao Yin.)*

Guihong Chen is with the School of Cyber Security, Guangdong Polytechnic Normal University, Guangzhou, 510635, China, and also with the School of Automation Science and Engineering, South China University of Technology, Guangzhou, 510641, China (e-mail: chenguihong@gpnu.edu.cn).

Mingyong Tong and Maojie Wang are with the School of Cyber Security, Guangdong Polytechnic Normal University, Guangzhou, 510635, China (e-mail: mingyongtong@outlook.com; maojie_wang@foxmail.com).

Jiao Yin and Hua Wang are with the Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne, VIC 3083, Australia (e-mail: {jiao.yin, hua.wang}@vu.edu.au).

Jinli Cao is with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3083, Australia (e-mail: j.cao@latrobe.edu.au).

THE evolution from the Internet of Things (IoT) to the Internet of Everything (IoE) has revolutionized urban traffic management by connecting sensors, devices, and data in real-time [1]. This interconnected network forms the foundation of smart transportation systems, enabling the optimization of traffic signals, flow distribution, and the prediction and mitigation of congestion and accidents, thereby improving efficiency and safety [2], [25]. These advancements also support sustainable urban development by enhancing the travel experience for citizens [3], [23]. Central to the success of smart transportation within the IoE framework is traffic flow prediction, which involves real-time and historical data to anticipate and respond to traffic conditions. This predictive capability empowers cities to implement proactive measures, optimize public transportation, and improve road usage efficiency.

Relying primarily on speed time series data from traffic sensors, accurate traffic prediction remains a significant challenge due to several inherent complexities. Traditional models like the autoregressive integrated moving average (ARIMA) [4] and the vector auto-regressive (VAR) model [12] are designed to handle simple linear relationships but struggle with the dynamic and complex correlations that characterize real-world traffic data. Although more recent deep learning methods, such as those combining convolutional neural networks (CNNs) [5] and recurrent neural networks (RNNs), have demonstrated better performance in capturing these complex correlations, they often only handle grid-like structured data, neglecting the underlying spatial topology of traffic networks.

Given that traffic networks are inherently graph-structured, graph neural networks (GNNs) have gained popularity in recent years for traffic prediction due to their ability to process graph-structured data effectively [32] [33]. Models like diffusion convolutional recurrent neural network (DCRNN) [6], spatio-temporal graph convolutional networks (STGCN) [7], and GraphWavenet [8] have shown notable improvements in prediction accuracy. However, these models typically rely on distance-based graph construction methods, where adjacency matrices are formed by calculating physical distances between sensors. Although intuitive, this approach assumes that physical proximity directly translates to traffic correlation, which can be overly simplistic in heterogeneous traffic networks. In practice, two geographically close sensors may be affected by different road conditions or traffic control policies, while distant sensors connected by major traffic flows or shared bottlenecks may show strong behavioral synchrony. To better reflect such semantic or functional similarity, we instead adopt a similarity-based construction using the cosine similarity of speed time series. This allows the graph to capture latent correlations driven by real traffic dynamics, rather than just geographical topology.

Additionally, traffic data is often collected by various agencies, and centralizing this data for model training raises significant privacy concerns [9]. To address these privacy issues, Federated Learning (FL) was introduced by Google in 2017 [10], enabling decentralized data training across multiple clients while preserving data locality. In recent years, various FL-based methods have been proposed to address real-world constraints in intelligent transportation and vehicular edge computing. For instance, MOB-FL [34] introduces a mobility-aware optimization mechanism to cope with unstable connectivity caused by vehicle movement in IoV scenarios, while Fed-HGT [35] integrates Transformer-based architectures with FL to capture long-range dependencies and heterogeneous graph structures. These studies highlight key advancements in adapting FL to dynamic environments and complex spatio-temporal data. However, these methods still have limitations in secure deployment. Mobility-aware FL approaches primarily improve training stability and convergence under client churn but often overlook the impact of adversarial clients on model integrity. On the other hand, Transformer-based FL frameworks generally assume well-behaved clients and stable coordination, limiting their applicability in realistic edge computing environments with privacy threats and data heterogeneity.

To address the challenges identified, we propose a novel framework, SecureGraphFL, specifically tailored for traffic prediction. This framework is designed to improve prediction accuracy while ensuring robust data privacy and resilience against attacks. To better capture temporal dynamics within traffic networks, we introduce speed difference time series as an additional feature alongside traditional speed time series data. We also employ a cosine similarity-based method for constructing graphs between traffic sensors, which enhances the spatial correlations captured by the GCN model used in each FL client. Furthermore, SecureGraphFL incorporates an actor-critic network-based client selection mechanism to manage the presence of malicious or failing clients effectively. This ensures the stability and reliability of the global model, even in challenging scenarios. The main contributions of this paper are as follows:

- We develop the SecureGraphFL framework, specifically designed for traffic flow prediction, which not only enhances predictive accuracy but also safeguards data privacy and defends against attacks.
- We introduce two key innovations to improve prediction accuracy: (1) incorporating a parallel speed difference prediction task, and (2) utilizing a cosine similarity-based graph construction method to capture spatial patterns among traffic sensors more effectively. Our experiments on two open-source datasets, PEMS-D7 and METR-LA, demonstrate that in the absence of malicious or failing clients, SecureGraphFL matches state-of-the-art centralized methods and significantly outperforms traditional FL algorithms like FedAVG and FedATT.
- We design an actor-critic network-based client selection mechanism that excludes malicious clients, ensuring the stability and robustness of the global model. In scenarios involving malicious clients, this mechanism effectively mitigates their impact, maintaining performance levels comparable to those in secure environments.

The remainder of this paper is organized as follows: Section II reviews related work, Section III covers preliminary concepts, Section IV details the proposed model framework, Section V discusses the experimental results, and Section VI concludes the paper.

II. Related Work

Traffic flow prediction is a crucial challenge in intelligent transportation systems, and various traditional and deep learning approaches have been explored to address this issue.

The ARIMA model [4], as a classical time series prediction model, combines auto-regressive (AR), integrated (I), and moving average (MA) components to capture temporal dependencies in traffic data. While ARIMA is effective for stationary time series, its performance declines with nonlinear and complex spatiotemporal dependencies, which are prevalent in real-world traffic data. The VAR model extends the AR model to handle multivariate time series, capturing interdependencies between multiple traffic variables [12]. However, VAR struggles with high-dimensional and nonlinear data, limiting its applicability to traffic flow prediction. Similarly, the k-nearest neighbors (KNN) algorithm [11] predicts traffic flow by identifying the most similar historical data points. While effective for simple patterns, KNN becomes computationally expensive and less effective when dealing with high-dimensional and large-scale traffic data.

Deep learning methods have demonstrated superior predictive capabilities, particularly in handling complex dependencies in traffic flow data. Feedforward neural networks (FNNs) [13] leverage multi-layer perceptrons to achieve nonlinear mappings of input data. However, FNNs, even when enhanced with feature engineering, are limited in their ability to model temporal dependencies effectively. Fully connected long short-term memory networks (FC-LSTM) [14] capture long-term dependencies in traffic flow data through the gating mechanisms of LSTM networks, incorporating fully connected layers to enhance nonlinear expression and temporal dependency modeling. Despite their strong predictive power, FC-LSTM models are computationally intensive and require long training times.

In recent years, GNNs have gained popularity for traffic prediction and various applications due to their ability to model the inherently graph-structured nature of traffic networks [24]. The STGCN model [7] combines GCN with temporal convolutional networks (TCN) to simultaneously capture spatiotemporal dependencies in traffic networks, significantly improving prediction accuracy. Furthermore, the attention-based spatiotemporal graph convolutional network (ASTGCN) [15] extends STGCN by incorporating self-attention mechanisms, allowing the model to focus more flexibly on critical spatiotemporal features, thereby further enhancing prediction accuracy. However, these GNN-based methods typically construct adjacency matrices based on the physical distances between traffic sensors. This distance-based approach may fail to capture the actual connections and dynamic relationships within complex road networks, limiting the predictive performance of these models. To address this limitation, our method proposes using a similarity-based approach for graph construction, which better captures the relationships between traffic sensors.

While the methods above focus on centralized learning, they do not address the growing need for data privacy and confidentiality. Federated Learning (FL) [10] has emerged as a solution, enabling decentralized model training across multiple clients while protecting sensitive data. A critical challenge in FL is the effective selection of participating clients, as variations in data quality, computational capabilities, and network conditions can severely affect convergence speed and model robustness.

To address these issues, researchers have proposed various client selection strategies. Li et al. [17] introduced a data quality-based selection method that involves complex evaluation processes and high computational overhead. Chen et al. [18] proposed a strategy prioritizing network conditions to reduce communication delay and overhead, though this could affect model fairness. Wang et al. [19] focused on incorporating resource constraints in client selection, which enhanced model representativeness while reducing computational and communication costs. Liu et al. [20] introduced dynamic adjustments in client participation strategies to balance model performance and computational efficiency in heterogeneous environments. Zhang et al. [21] developed an adaptive client selection method that improves model adaptability and generalization by adjusting strategies according to dynamic environmental changes. Recent research has also explored mobility-aware FL [34] and Transformer-based FL frameworks [35] to improve adaptability and temporal modeling in dynamic, decentralized environments. While these approaches offer important advancements, they often assume trustworthy participants and do not address robustness against malicious or unreliable clients, especially in adversarial or non-IID settings.

Building on these insights, our work proposes a client selection method based on actor-critic networks, focusing on excluding malicious/failed clients to ensure robust and secure federated learning.

III. Preliminary

A. Graph Modeling in Traffic Prediction

In smart transportation systems, traffic sensors are placed along the roadside to periodically collect speed data. In this paper, the graph between traffic sensors is defined as an undirected graph$\mathbf{G} = (\mathcal{V}, \mathbf{E})$, where $\mathcal{V}$ represents the set of nodes, and each node $u_i \in \mathcal{V}$ (with $1 \leq i \leq N$) corresponds to a traffic sensor, with N being the total number of nodes. The edge set $\mathbf{E}$ represents the connections between nodes and is defined by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. An edge $e_{(i,j)} \in \mathbf{E}$ (with $1 \leq i, j \leq N$) exists between nodes u_i and u_j when $[\mathbf{A}]_{i,j} > 0$.

B. Traffic Prediction Problem

For all traffic sensors in the traffic network, historical speed records collected from each sensor u_i are denoted as $\mathbf{v}_i = [v_i^{(1)}, v_i^{(2)}, \ldots, v_i^{(T)}]$, where T is the total number of time steps recorded. At each time step t, the feature set $\mathbf{x}^{(t)} = \{\mathbf{v}_1^{(t)}, \mathbf{v}_2^{(t)}, \ldots, \mathbf{v}_N^{(t)}\}$ is constructed from the most recent d speed records as shown in Equation (1).

$$\mathbf{v}_i^{(t)} = \left[v_i^{(t-d+1)}, v_i^{(t-d+2)}, \ldots, v_i^{(t)}\right]. \tag{1}$$

Here, $v_i^{(t)}$ denotes u_i's observed speed at time step t.

The traffic prediction problem can be formulated as the task of training a model f to predict the speed of traffic sensors at a future time step $t+k$, denoted as $\hat{\mathbf{y}}^{(t+k)}$, where $k \in \mathbb{R}_{>0}$, as shown in Equation (2).

$$\hat{\mathbf{y}}^{(t+k)} = \{\mathbf{v}_1^{(t+k)}, \mathbf{v}_2^{(t+k)}, \cdots, \mathbf{v}_N^{(t+k)}\} = f\left(\mathbf{x}^{(t)}, \boldsymbol{\omega}\right). \tag{2}$$

Here, $\boldsymbol{\omega}$ is the set of trainable parameters of the model f.

C. Federated Learning Pipeline

In a typical FL setup, there are a total of M clients collaborating with a central server to train a shared global model, ensuring data privacy by not sharing raw data. The m-th client ($1 \leq m \leq M$) maintains its local dataset, denoted as $\mathbf{D}_m$, and contributes to the FL process by performing local model training and updates. The parameters of the global model, aggregated from the clients' updates, are denoted by $\boldsymbol{\omega}_g$. The FL pipeline can be summarized in the following steps:

1) Initialization: The central server initializes the global model parameters $\boldsymbol{\omega}_g^{(0)}$ and broadcasts them to all M clients.

2) Local training: Each client independently trains its local model on its own dataset $\mathbf{D}_m$ by updating its local model parameters $\boldsymbol{\omega}_m$. The update rule for the m-th client is given by:

$$\boldsymbol{\omega}_m = \boldsymbol{\omega}_m - \eta \nabla f(\boldsymbol{\omega}_m), \tag{3}$$

where η is the learning rate and $f(\boldsymbol{\omega}_m)$ is the local loss function for client m.

3) Aggregation: After local training, each client sends its updated model parameters $\boldsymbol{\omega}_m^{(l)}$ to the central server. The server aggregates these parameters using an aggregation function $\mathcal{F}(\cdot)$, updating the global model parameters as Equation (4)

$$\boldsymbol{\omega}_g^{(l)} = \mathcal{F}\left(\{\boldsymbol{\omega}_m^{(l)}\}_{m=1}^{M}\right), \tag{4}$$

where l denotes the communication round. The specific aggregation function $\mathcal{F}(\cdot)$ depends on the FL algorithm employed (e.g., federated averaging or attention-based aggregation).

4) Iteration until convergence: Steps 2 and 3 are repeated iteratively until the global model $\boldsymbol{\omega}_g$ converges to a satisfactory level of accuracy, minimizing the global loss function $F(\boldsymbol{\omega}_g)$.

D. Actor-Critic Network

The actor-critic network is a reinforcement learning architecture comprising two main components: the actor network and the critic network.

1) Actor network: The actor network is responsible for determining the actions $a^{(t)}$ that should be taken at each episode step t, based on the current state $s^{(t)}$. The policy, denoted as $\pi(a^{(t)} \mid s^{(t)}; \phi)$, is parameterized by ϕ, which represents the set of trainable parameters within the actor network.

The parameters ϕ of the actor network are updated using policy gradient methods, which optimize the expected return $J(\phi)$. The gradient of the objective function $J(\phi)$ is computed by Equation (5).

$$\nabla_\phi J(\phi) = \mathbb{E}_\pi \left[\nabla_\phi \log \pi(a^{(t)} \mid s^{(t)}; \phi) Q(s^{(t)}, a^{(t)}; \omega) \right], \quad (5)$$

where $\nabla_\phi J(\phi)$ represents the gradient of the expected return with respect to the policy parameters. $\log \pi(a^{(t)} \mid s^{(t)}; \phi)$ is the log-probability of selecting action $a^{(t)}$ in state $s^{(t)}$ under the current policy. $Q(s^{(t)}, a^{(t)}; \omega)$ is the action-value function, as estimated by the critic network.

The actor network updates its parameters by ascending the gradient of the expected return $J(\phi)$, guided by the critic network's estimation of the action-value function $Q(s^{(t)}, a^{(t)}; \omega)$. This encourages the selection of actions that maximize the cumulative expected reward over the long run.

The actor's parameters are updated by Equation (6).

$$\phi \leftarrow \phi + \alpha \nabla_\phi J(\phi), \quad (6)$$

where α denotes the learning rate.

2) Critic network: The critic network evaluates the quality of the actions taken by the actor network by estimating an action-value function $Q(s^{(t)}, a^{(t)}; \omega)$. This function represents the expected cumulative reward when starting from state $s^{(t)}$ and taking action $a^{(t)}$, with the critic network's parameters denoted by ω.

To minimize the temporal difference (TD) error, the critic network updates its parameters ω. The TD error $\delta^{(t)}$ is computed by Equation (7).

$$\delta^{(t)} = r^{(t)} + \gamma Q(s^{(t+1)}, a^{(t+1)}; \omega) - Q(s^{(t)}, a^{(t)}; \omega), \quad (7)$$

where $r^{(t)}$ is the immediate reward received at episode step t. γ is the discount factor, determining the importance of future rewards. $Q(s^{(t+1)}, a^{(t+1)}; \omega)$ is the predicted value for the next state-action pair. The TD error $\delta^{(t)}$ quantifies the discrepancy between the critic's estimated action-value $Q(s^{(t)}, a^{(t)}; \omega)$ and the actual observed reward plus its estimate of future rewards. Minimizing this discrepancy helps the critic network improve its accuracy in evaluating actions, thereby providing increasingly precise feedback for the actor's updates. Through iterative minimization of this TD error, the critic network gradually refines its estimation capability, enhancing the overall stability and convergence of the actor-critic mechanism.

The critic's parameters are then updated by minimizing the TD error in Equation (8).

$$\omega \leftarrow \omega + \beta \delta^{(t)} \nabla_\omega Q(s^{(t)}, a^{(t)}; \omega), \quad (8)$$

where β represents the learning rate, and $\nabla_\omega Q(s^{(t)}, a^{(t)}; \omega)$ is the gradient of the action-value function with respect to ω.

It is important to note that in the context of the actor-critic network, all references to t indicate episode steps within the reinforcement learning process and are distinct from the time steps used in traffic prediction.

IV. METHODOLOGY

We present the design of SecureGraphFL for traffic flow prediction, as depicted in Fig. 1. The lower part of the figure illustrates the enhanced client-side traffic prediction with speed difference time series and the similarity-based graph construction process. The upper portion highlights the actor-critic network-based client selection mechanism and the global model aggregation process. Each client is responsible for a total of C_m traffic sensors, where the total number of sensors across all clients sums to the total number of sensors in the entire system, denoted as N, such that $\sum_{m=1}^{M} C_m = N$.

In the framework, each client constructs a graph using an adjacency matrix based on cosine similarity from traffic sensor data. This graph is processed by two separate GCNs: one handling the original speed data and the other processing the speed difference time series. After local training, clients send their updated model parameters to the server. The server uses these parameters to compute the local model loss on a global test dataset. This loss, along with the training time, forms the state input to the actor-critic network. The network then selects trustworthy clients for global model aggregation. The selected clients' contributions are weighted through an attention mechanism during aggregation, leading to a robust global model update. The updated global model's performance is monitored by calculating a global loss, which is then used to compute the reward. This reward, along with the selected state-action pairs and the next state, is stored in the experience replay buffer for continuous learning.

A. Similarity-based Graph Construction within a Client

In SecureGraphFL, we adopt a similarity-based graph construction method to build a graph $\mathbf{G}_m = (\mathcal{V}_m, \mathbf{E}_m)$ for the sensors within the coverage area of client m. Specifically, the cosine similarity between two nodes (traffic sensors) u_i and u_j within client m is calculated by Equation (9).

$$\cos_{(i,j)} = \frac{\mathbf{v}_i \mathbf{v}_j^\top}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}, \quad 1 \le i, j \le C_m, \quad (9)$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ denote the historical speed vectors, and $\|\cdot\|$ denotes the Euclidean norm.

The adjacency matrix $\mathbf{A}_m \in \mathbb{R}^{C_m \times C_m}$ for client m is constructed based on these similarities as Equation (10).

$$[\mathbf{A}_m]_{i,j} = \begin{cases} \cos_{(i,j)}, & \text{if } \cos_{(i,j)} > \tau, \\ 0, & \text{otherwise}, \end{cases} \quad (10)$$

where the threshold τ determines the sparsity of the adjacency matrix, ensuring meaningful edges are retained.

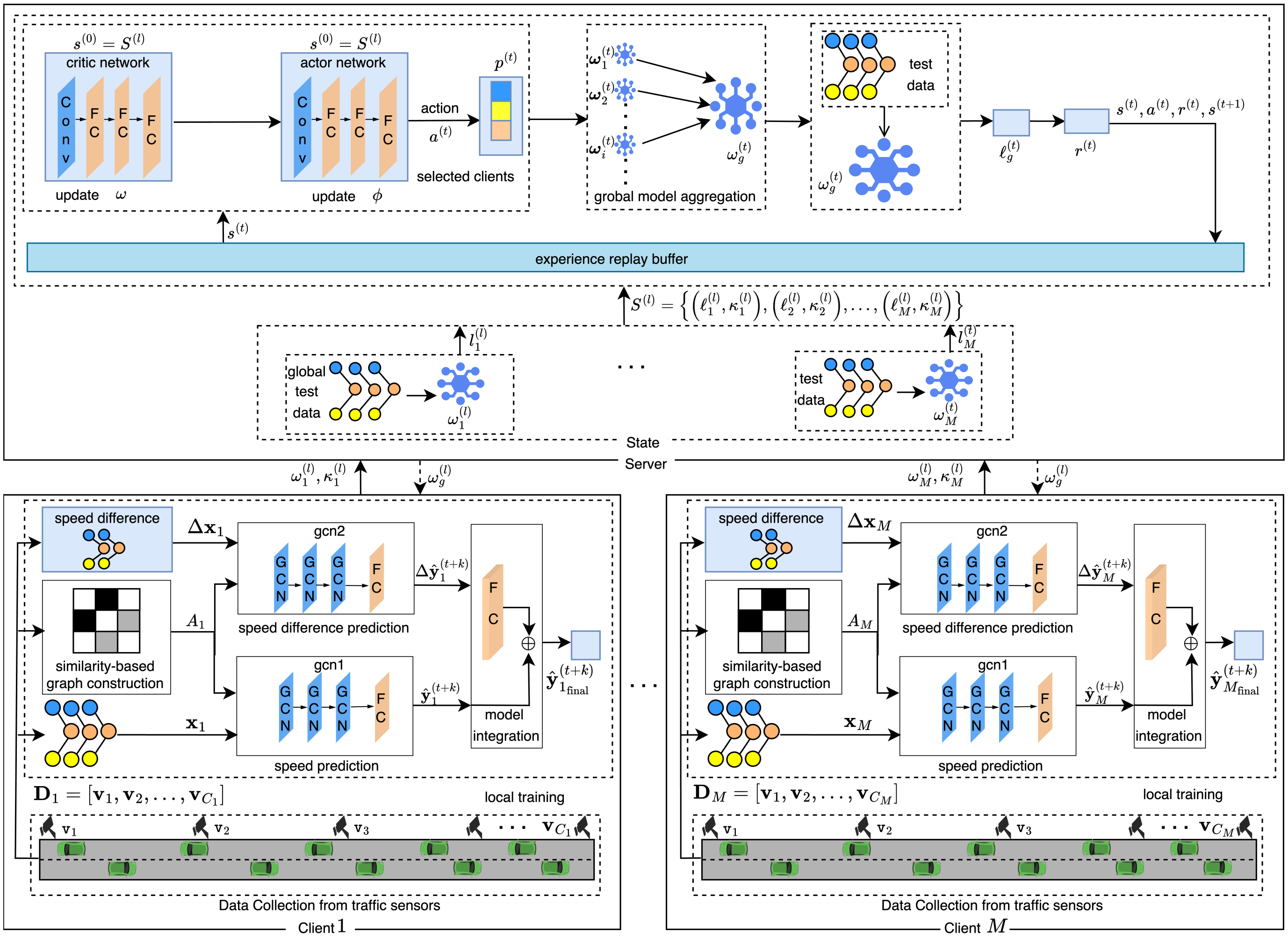

Fig. 1. The overall architecture of SecureGraphFL.

B. Enhanced Client-Side Traffic Prediction with Speed Difference Time Series

To improve traffic prediction performance, we introduce a speed difference time series vector for each client m, denoted as $\Delta\mathbf{x}_m^{(t)} = \{\Delta\mathbf{v}_1^{(t)}, \Delta\mathbf{v}_2^{(t)}, \cdots, \Delta\mathbf{v}_{C_m}^{(t)}\}$, which serves as an additional feature extracted from historical speed records. The speed difference feature vector for each sensor covered by the m-th client, denoted as $\Delta\mathbf{v}_i^{(t)}$ ($i \in \{1, 2, \cdots, C_m\}$), is calculated using Equation (11).

$$\Delta\mathbf{v}_i^{(t)} = \left[\Delta v_i^{(t)}, \Delta v_i^{(t-1)}, \ldots, \Delta v_i^{(t-d+1)}\right], \tag{11}$$

where $\Delta v_i^{(t)} = v_i^{(t)} - v_i^{(t-1)}$.

To improve the prediction accuracy of each client m, we employ two separate GCN models, denoted as gcn1 and gcn2. The gcn1 model predicts the sensors' speed in client m at the future time step $t+k$ based on $\mathbf{x}_m^{(t)} = \{\mathbf{v}_1^{(t)}, \mathbf{v}_2^{(t)}, \cdots, \mathbf{v}_{C_m}^{(t)}\}$, where k represents the number of time steps ahead of the current time t. The predicted speed is denoted as $\hat{\mathbf{y}}_m^{(t+k)} = \{\hat{v}_1^{(t+k)}, \hat{v}_2^{(t+k)}, \cdots, \hat{v}_{C_m}^{(t+k)}\}$. The gcn2 model predicts the sensors' speed difference in client m at the future time step $t+k$, denoted as $\Delta\hat{\mathbf{y}}_m^{(t+k)} = \{\Delta\hat{v}_1^{(t+k)}, \Delta\hat{v}_2^{(t+k)}, \cdot, \Delta\hat{v}_{C_m}^{(t+k)}\}$. These processes are described by Equations (12) and (13).

$$\hat{\mathbf{y}}_m^{(t+k)} = f_{\text{gcn1}}\left(\mathbf{x}_m^{(t)}, \boldsymbol{\omega}_{\text{gcn1}}\right), \tag{12}$$

$$\Delta\hat{\mathbf{y}}_m^{(t+k)} = f_{\text{gcn2}}\left(\Delta\mathbf{x}_m^{(t)}, \boldsymbol{\omega}_{\text{gcn2}}\right), \tag{13}$$

where f_{gcn1} and f_{gcn2} are the mapping functions of the respective GCN models, and $\boldsymbol{\omega}_{\text{gcn1}}$ and $\boldsymbol{\omega}_{\text{gcn2}}$ represent their trainable parameter sets.

The final speed prediction $\hat{\mathbf{y}}_{m_{\text{final}}}^{(t+k)}$ for the sensors of the m-th client is obtained by combining $\hat{\mathbf{y}}_m^{(t+k)}$ with the output of a fully connected neural network that processes $\Delta\hat{\mathbf{y}}_m^{(t+k)}$.

C. Actor-Critic Network-Based Client Selection for Global Model Aggregation

The actor-critic network dynamically selects optimal subsets of clients for global model aggregation. Specifically, the actor network selects subsets of clients by optimizing the expected cumulative reward, while the critic evaluates the quality of these selections by estimating the action-value function. The critic provides feedback through TD errors, enabling the actor to iteratively improve its client-selection decisions. This process involves evaluating client models, forming state representations, operating the actor-critic network, and updating network parameters based on the observed rewards at each communication round.

1) Client Model Evaluation and State Formation: During each communication round l, after clients complete their local model training, they upload their trained model parameters $\boldsymbol{\omega}_m^{(l)}$ and training times $\kappa_m^{(l)}$ to the server. The server evaluates each client's model on a global test dataset, calculating the corresponding loss $\ell_m^{(l)}$. This information is compiled into a communication state set $S^{(l)}$, as shown in Equation (14), which serves as the initial state for the actor-critic network in the l-th communication round.

$$S^{(l)} = \{(\ell_m^{(l)}, \kappa_m^{(l)}) \mid m = 1, 2, \ldots, M\}. \tag{14}$$

This state information, reflecting the performance and efficiency of each client, is also stored in the experience replay buffer B for use during the actor-critic network training process.

2) Actor-Critic Network-Based Client Selection Algorithm: In SecureGraphFL, the actor-critic network is responsible for dynamically selecting clients for global model aggregation during each communication round l. This process involves state observation, action selection, reward computation, and network parameter updates, all aimed at optimizing the global model's performance in a federated learning environment, as shown in Algorithm 1.

Initially, the server observes the communication state $S^{(l)}$ and sets it as the initial actor-critic network training state $s^{(0)}$. At each episode step t, the actor-critic network observes $s^{(t)}$, and guided by the policy $\pi(a^{(t)} \mid s^{(t)}; \phi)$, the actor network selects a subset from M clients, denoted as $p^{(t)}$, for aggregation. This selection balances exploration and exploitation: with probability ϵ, the network explores by selecting clients randomly, encouraging diversity in participation; with probability $1-\epsilon$, it exploits the current policy to select clients based on their predicted contribution to the global model's performance, as stated in Lines 6-8.

In Line 9, the server aggregates model parameters from the selected clients in $p^{(t)}$ to obtain a global model; then computes the global model loss $\ell_g^{(t)}$, which reflects how well the updated global model performs. Using this loss, the immediate reward $r^{(t)}$ is calculated according to Equation (15), guiding the actor-critic network in selecting clients that enhance model performance. Finally, the environment transitions to the next state $s^{(t+1)}$, capturing the impact of the updated global model.

$$r^{(t)} = \begin{cases} -e^{-\ell_g^{(t)}}, & \text{if } e^{-\ell_g^{(t)}} > \xi, \\ e^{-\ell_g^{(t)}}, & \text{otherwise.} \end{cases} \tag{15}$$

Here, the reward function uses the exponential form $e^{-\ell_g^{(t)}}$ to reflect the quality of the global model. This choice ensures a smooth, bounded, and monotonically decreasing reward with respect to the validation loss $\ell_g^{(t)}$: smaller losses yield rewards closer to 1, while larger losses produce rewards closer to 0.

Algorithm 1 Actor-Critic Network-Based Client Selection for the l-th Communication Round in SecureGraphFL

1: **Input:** State $S^{(l)} = \{(\ell_m^{(l)}, \kappa_m^{(l)}) \mid m = 1, 2, \ldots, M\}$
2: **Output:** Selected clients set $P^{(l)}$ for aggregation
3: **Initialization:** Initialize $s^{(0)} = S^{(l)}$, ϕ, ω, B, γ, α, β, ϵ, and b
4: **for** each episode **do**
5:   **for** each step t in the episode **do**
6:     Observe the current state $s^{(t)}$
7:     Select an action $a^{(t)}$ using policy $\pi(a^{(t)} \mid s^{(t)}; \phi)$
8:     Generate $p^{(t)}$ by balancing exploration and exploitation with probability ϵ
9:     Aggregate clients' models in $p^{(t)}$, compute $\ell_g^{(t)}$, then $r^{(t)}$ using Equation (15), and generate $s^{(t+1)}$
10:     Store the transition $(s^{(t)}, a^{(t)}, r^{(t)}, s^{(t+1)})$ in the experience replay buffer B
11:     **if** $z < b$ **then**
12:       Use all z experience tuples in B for training
13:     **else**
14:       Randomly sample b experience tuples from B for training
15:     **end if**
16:     Calculate the TD error $\delta^{(t)}$ via Equation (7)
17:     Update the critic network parameters ω based on $\delta^{(t)}$ via Equation (8)
18:     Update the actor network parameters ϕ using the policy gradient calculated via Equation (6)
19:     Update $s^{(t)} = s^{(t+1)}$
20:   **end for**
21: **end for**
22: $P^{(l)} = p^{(t)}$

This encourages the actor network to favor client selections that improve global performance.

The threshold parameter ξ introduces a reward flipping mechanism to enhance exploration. Specifically, when the global model is already performing well ($e^{-\ell_g^{(t)}} > \xi$), a negative reward is assigned to discourage over-reliance on currently preferred clients. This penalization prevents the policy from becoming overly greedy and promotes diversity in client participation. A smaller ξ leads to more selective aggregation (favoring low-loss clients), while a larger ξ relaxes the selection to allow broader exploration. The value of ξ is determined empirically based on validation performance.

The transition $(s^{(t)}, a^{(t)}, r^{(t)}, s^{(t+1)})$ is stored in the experience replay buffer B, as shown in Line 10, enabling the actor-critic network to learn from past experiences, thereby improving the stability of the learning process. If the current number of experience tuples z in B is less than the batch size b, all available experience tuples are used for training. Otherwise, a batch of b experience tuples is randomly sampled from B, as shown in Lines 11-14. The critic network then uses this batch to calculate the TD error $\delta^{(t)}$ via Equation (7), which represents the difference between the expected and observed rewards, as depicted in Line 16. This TD error is used to update the critic network's parameters ω via Equation

(8), while the actor network's parameters ϕ are refined using the policy gradient method via Equation (6), as shown in Lines 17-18. Finally, the client set $p^{(t)}$ obtained in the last episode step is used as the selected client set for the l-th communication round between the server and clients, denoted as $P^{(l)}$, as shown in Line 22.

This iterative process ensures that the actor-critic network continuously improves client selection strategies, enhancing the overall robustness and performance of the SecureGraphFL system in federated learning environments.

D. SecureGraphFL for Traffic Prediction

This section provides a comprehensive overview of the SecureGraphFL algorithm, focusing on its application in traffic prediction. The algorithm integrates various advanced techniques, including GCNs, an actor-critic network for client selection, and an attention-based aggregation mechanism to enhance model performance and ensure robustness in a federated learning environment, as described in Algorithm 2.

1) Global Model Distribution and Local Model Update: At the beginning of each communication round l, the global model parameters $\boldsymbol{\omega}_g^{(l)}$ are distributed to all participating clients. Each client m then performs the following steps:

Firstly, as shown in Line 7, each client m constructs a similarity-based graph $\mathbf{G}_m = (\mathcal{V}_m, \mathbf{E}_m)$ using cosine similarity, as detailed in Section IV-A. In Line 8, the client m computes a speed difference time series vector $\Delta \mathbf{x}_m^{(t)}$.

Then, in Lines 9-11, the algorithm utilizes two GCN models and integrates their outputs to predict the final speed prediction $\hat{\mathbf{y}}_{m_{\text{final}}}^{(t+k)}$. The details are shown in Section IV-B.

After computing the final speed prediction, the client updates its local model and calculates the updated local model parameters $\boldsymbol{\omega}_m^{(l)}$ in Line 12. These parameters, along with the local training time $\kappa_m^{(l)}$, are then sent to the server for further processing in Line 13.

2) State Observation and Client Selection: Upon receiving the local model parameters and training times from all clients, the server constructs the state $S^{(l)}$ as shown in Line 15, representing the performance metrics of all clients. The server then uses an actor-critic network, as described in Line 16, to select the client set $P^{(l)}$ for aggregation, as detailed in Section IV-C.

3) Attention-Based Aggregation: Then, the server employs an attention-based aggregation mechanism to prioritize the most relevant contributions from the selected clients using Equation (16), as shown in Line 17.

$$\boldsymbol{\omega}_g^{(l)} = f_{\text{att}}(\{\omega_m^{(l)}\}_{m=1}^{|P^{(l)}|}, \Theta_{\text{att}}). \tag{16}$$

Here, f_{att} is a mapping function of the attention-based aggregation mechanism with parameter set Θ_{att}. $|P^{(l)}|$ represents the number of clients in the set $P^{(l)}$.

This attention-based aggregation process ensures that the most relevant client updates, i.e., those with higher attention scores, have a greater impact on the global model, thereby enhancing the robustness and accuracy of the traffic prediction model.

Algorithm 2 SecureGraphFL for Traffic Prediction

1: **Input:** Traffic flow dataset, total number of communication rounds $\mathscr{L}$, total number of clients M
2: **Output:** Trained global model for traffic prediction
3: Initialize the global model with parameters $\boldsymbol{\omega}_g^{(0)}$
4: **for** l from 1 to $\mathscr{L}$ **do**
5: $\boldsymbol{\omega}_m^{(l)} = \boldsymbol{\omega}_g^{(l-1)}, m = 1, \ldots, M$
6: **for** each client m in parallel **do**
7: Constructs a graph $\mathbf{G}_m = (\mathcal{V}_m, \mathbf{E}_m)$ using cosine similarity
8: Computes the speed difference time series vector $\Delta \mathbf{x}_m^{(t)}$ as Equation (11)
9: Predict $\hat{\mathbf{y}}_m^{(t+k)}$ using gcn1 as Equation (12)
10: Predict $\Delta \hat{\mathbf{y}}_m^{(t+k)}$ using gcn2 as Equation (13)
11: Integrate predictions to compute the final speed prediction $\hat{\mathbf{y}}_{m_{\text{final}}}^{(t+k)}$
12: Updates the local model $\boldsymbol{\omega}_m^{(l)}$ using local data
13: Sends $\boldsymbol{\omega}_m^{(l)}$ and $\kappa_m^{(l)}$ to the server
14: **end for**
15: Server observes the state $S^{(l)} = \{(\ell_m^{(l)}, \kappa_m^{(l)}) \mid m = 1, \ldots, M\}$
16: Server selects clients $P^{(l)}$ based on Algorithm 1
17: Server updates the global model parameters via Equation (16)
18: **end for**

E. Theoretical Convergence Analysis of Actor-Critic-based Client Selection

To rigorously validate the convergence of the actor-critic-based client selection mechanism with the attention-based aggregation in SecureGraphFL, we first state necessary assumptions and then present the theorem, with detailed proof deferred to the Appendix.

Assumption 1: The variance of local gradients for any client m is uniformly bounded, as shown in Equation (17).

$$\mathbb{E}\left[\|\nabla f(\boldsymbol{\omega}_m^{(l)}) - \nabla f(\boldsymbol{\omega}_g^{(l)})\|^2\right] \le \sigma^2, \quad \forall m, l, \tag{17}$$

where σ^2 is a positive constant. This ensures bounded variability in client updates.

Assumption 2: The actor network, parameterized by ϕ, produces policies π_ϕ that satisfy Lipschitz continuity, as shown in Equation (18).

$$\|\pi_{\phi_1}(a|s) - \pi_{\phi_2}(a|s)\| \le L_\pi \|\phi_1 - \phi_2\|, \quad \forall s, a, \tag{18}$$

where L_π is the Lipschitz constant. ϕ_1 and ϕ_2 represent any two distinct parameter sets of the actor network. This assumption characterizes the smoothness of changes in the policy output with respect to its parameters.

The global loss function at communication round l is defined as Equation (19).

$$f(\boldsymbol{\omega}_g^{(l)}) = \sum_{m \in P^{(l)}} \alpha_m^{(l)} f(\boldsymbol{\omega}_m^{(l)}), \quad \text{with} \quad \sum_{m \in P^{(l)}} \alpha_m^{(l)} = 1, \tag{19}$$

where $\alpha_m^{(l)}$ denotes the attention weights derived from the attention-based aggregation mechanism.

Theorem 1: Under Assumptions 1 and 2, the actor-critic-based client selection mechanism in SecureGraphFL converges to a stationary policy π^*. Consequently, the global model parameters $\boldsymbol{\omega}_g^{(l)}$ aggregated through the attention-based aggregation mechanism defined in Equation (16), converge to a stationary point. Formally, the expected squared norm of the gradient satisfies the convergence condition given in Equation (20).

$$\lim_{l\to\infty} \mathbb{E}[\|\nabla f(\boldsymbol{\omega}_g^{(l)})\|^2] = 0. \tag{20}$$

Proof: See the Appendix.

Corollary 1: As the client selection policy stabilizes to π^*, malicious or less effective clients are consistently excluded. This results in monotonically decreasing variance in the global gradient estimates, thereby accelerating the convergence of SecureGraphFL compared to random or static client selection strategies.

F. Scalability and Overhead Analysis

We analyze the scalability of SecureGraphFL by evaluating its computational and communication overhead per communication round. All complexity estimates are computed per communication round unless otherwise noted.

1) Computational Overhead: **Client-side.** Each client trains a local GCN model on a subgraph with C_m sensors. Let L denote the number of GCN layers and H the hidden dimension. The per-round training complexity at each client is as $\mathcal{O}(L \cdot C_m \cdot H^2)$. Once the GCN model at the client side is determined, both L and H become constant values. Therefore, the overall complexity reduces to Equation (21).

$$\mathcal{O}(C_m). \tag{21}$$

Server-side. The server performs three main types of computations:

Model evaluation. Each uploaded model $\boldsymbol{\omega}_m^{(l)}$ is evaluated on a validation set with C_v sensors. The total cost of model evaluation across all clients is given in $\mathcal{O}(M \cdot C_v)$.

Actor-critic client selection. The state $S^{(l)} \in \mathbb{R}^{M\times 2}$, which includes the training loss and latency of each client, is input into the actor and critic networks. As these networks have fixed architecture and process low-dimensional input, the corresponding computational complexity is $\mathcal{O}(M)$.

Model aggregation with attention. Attention weights are computed based on the norm differences between each $\boldsymbol{\omega}_m^{(l)}$ and the current global model $\boldsymbol{\omega}_g^{(l)}$. Let d be the input dimension of the attention mechanism. The cost of attention computation and weighted aggregation over the selected client set $P^{(l)}$ is given in $\mathcal{O}(|P^{(l)}|\cdot d + |\boldsymbol{\omega}_g^{(l)}|)$, where $|\boldsymbol{\omega}_g^{(l)}|$ represents the global model size (i.e., the total number of learnable parameters).

The total server-side computational overhead per round, combining the above components, is given in Equation (22).

$$\mathcal{O}(M \cdot C_v + M + |P^{(l)}| \cdot d + |\boldsymbol{\omega}_g^{(l)}|). \tag{22}$$

2) Communication Overhead: **Uplink.** In each round, all M clients upload their trained model parameters $\boldsymbol{\omega}_m^{(l)}$ along with their training time $\kappa_m^{(l)}$. The total uplink communication overhead is given in $\mathcal{O}(M\cdot(|\boldsymbol{\omega}_m^{(l)}|+1))$, where $|\boldsymbol{\omega}_m^{(l)}|$ represents the client model size.

Downlink. The server broadcasts the aggregated global model $\boldsymbol{\omega}_g^{(l)}$ to all clients. The resulting downlink communication overhead is given in $\mathcal{O}(M \cdot |\boldsymbol{\omega}_g^{(l)}|)$.

Total communication overhead. By summing the uplink and downlink costs, the total communication complexity per round is given by Equation (23).

$$\mathcal{O}\left(M \cdot \left(|\boldsymbol{\omega}_m^{(l)}| + |\boldsymbol{\omega}_g^{(l)}|\right)\right). \tag{23}$$

Since all clients and the server share the same model architecture, it holds that $|\boldsymbol{\omega}_m^{(l)}| = |\boldsymbol{\omega}_g^{(l)}|$. Therefore, Equation (23) simplifies to Equation (24).

$$\mathcal{O}(M \cdot |\boldsymbol{\omega}_g^{(l)}|). \tag{24}$$

In summary, Equations (21)–(24) confirm that the overall computational and communication overhead of SecureGraphFL scales linearly with key system factors, including the number of clients M, local sensor count C_m, and model size $|\boldsymbol{\omega}_g^{(l)}|$. This linear scalability ensures that the framework remains efficient and practical even in large-scale federated deployments with moderate model complexity.

V. Experiments

All experiments were conducted on a machine with the following specifications: Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz and an NVIDIA GeForce GTX 3090 GPU.

A. Datasets and Preprocessing

We validated our model using two real-world traffic datasets, PeMS-D7 and METR-LA, collected by the California Department of Transportation and the Los Angeles Department of Transportation, respectively. Both datasets include essential traffic attributes, geographical information, and corresponding timestamps, which are summarized below.

PeMS-D7 was collected from over 39,000 sensor stations deployed across major metropolitan areas in California's state highway system, provided by the Caltrans Performance Measurement System (PeMS) [22]. The data were aggregated from 30-second intervals into 5-minute intervals. Specifically, we used the same smaller variant, PeMS-D7(M), as in [7], which consists of data from 228 randomly selected stations in California's District 7 to reduce computational costs. The time range for PeMS-D7(M) covers weekdays in May and June 2012.

METR-LA contains traffic speed information collected from March 1, 2012, to June 30, 2012, via 207 loop detectors installed on highways in Los Angeles County. Similar to PeMS-D7(M), the METR-LA dataset records traffic data at 5-minute intervals.

This paper adopts classical regression metrics: mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean squared error (RMSE) to evaluate traffic prediction performance.

TABLE I
PERFORMANCE COMPARISON OF CENTRALIZED LEARNING AND FL ALGORITHMS ON THE PEMS-D7(M) DATASET

Model	PeMS-D7(M) (15/30/45 min)		
	MAE	MAPE (%)	RMSE
HA[1]	4.01	10.61	7.20
LSVR[1]	2.50/3.63/4.54	5.81/8.88/11.50	4.55/6.67/8.28
ARIMA[1]	5.55/5.86/6.27	12.92/13.94/15.20	9.00/9.13/9.38
FNN[1]	2.74/4.02/5.04	6.38/9.72/12.38	4.75/6.98/8.58
FC-LSTM[1]	3.57/3.94/4.16	8.60/9.55/10.10	6.20/7.03/7.51
GCCRU[1]	2.37/3.31/4.01	5.54/8.06/9.99	4.21/5.96/7.13
STGCN(Cheb)[1]	**2.25/3.03/3.57**	5.26/**7.33/8.69**	**4.04/5.70/6.77**
STGCN(1st)[1]	2.26/3.09/3.79	**5.24**/7.39/9.12	4.07/5.77/7.03
FedAVG[2]	8.64/8.65/8.65	27.89/27.94/27.96	14.18/14.20/14.20
FedATT[2]	8.64/8.65/8.65	27.90/27.94/27.96	14.18/14.20/14.21
FedProx[2]	8.65/8.66/8.66	27.96/27.89/28.00	14.22/14.16/14.23
SCAFFOLD[2]	8.65/8.66/8.66	28.06/28.07/28.10	14.29/14.29/14.31
SecureGraphFL[2]	**2.37/3.34/4.08**	**5.60/8.26/10.52**	**4.47/6.46/7.81**

[1] These are centralized learning approaches.
[2] These are federated learning approaches.

B. Results Without Malicious or Failed Clients

1) Comparison with Other Traffic Prediction Methods: We first compared the performance of SecureGraphFL with other traffic prediction methods on the PeMS-D7(M) dataset in scenarios without malicious or failed clients. For each data sample, a 60-minute historical speed time series was used to predict the speed for the next 15 minutes, 30 minutes, and 45 minutes. The experimental results are presented in Table I.

The top half of Table I lists nine centralized learning algorithms, including historical average (HA), linear support vector regression (LSVR), ARIMA, FNN, FC-LSTM, Graph Convolutional GRU (GCGRU), and two versions of STGCN. The bottom half of Table 1 presents three FL algorithms, including the proposed SecureGraphFL. All FL models use a 3-layer GCN as the base model. Among them, both FedAVG and FedATT use speed time series as features and employ a distance-based graph construction method. For the global model aggregation approach, FedAVG uses simple averaging, and FedATT employs an attention mechanism.

The results for the centralized learning algorithms were directly taken from the paper [7]. For FL algorithms, the following experimental procedure was used: The data from each traffic sensor was split into a 70% training set, a 15% validation set, and a 15% test set. The traffic sensors were evenly distributed across nine clients, with any remainder being assigned to the last client. Each client built a graph using either distance-based or cosine similarity-based methods. All training data from the traffic sensors in the same client were pooled to train the client's GCN model, and the pooled validation data was used for early stopping. The global test set was created by combining the test data from all clients to evaluate the performance of the global model.

As shown in Table I, in the absence of malicious or failed clients, our proposed SecureGraphFL significantly outperforms other federated learning algorithms, including FedAVG [10], FedATT [29], FedProx [30], and SCAFFOLD [31], across all metrics (MAE, MAPE, and RMSE) for the 15-minute, 30-minute, and 45-minute prediction intervals. For example, SecureGraphFL achieves an MAE of 2.37 for the 15-minute prediction, significantly outperforming other FL methods, including FedAVG and FedATT (both 8.64) as well as FedProx and SCAFFOLD (both 8.65). Even when compared to the complex centralized learning models like GCCRU and the two variants of STGCN, SecureGraphFL shows a very small performance gap. For instance, SecureGraphFL's MAE for the 15-minute horizon is 2.37, closely matching the best-centralized model, STGCN (1st), which has an MAE of 2.26. Similarly, SecureGraphFL's RMSE and MAPE are also competitive with these advanced centralized models. In contrast, SecureGraphFL performs better than the other five centralized learning methods (HA, LSVR, ARIMA, FNN, and FC-LSTM), indicating its strong capability even in a federated learning setting.

TABLE II
ABLATION STUDY RESULTS ON PEMS-D7(M) AND METR-LA DATASETS

Model	Setting			PeMS-D7(M)		METR-LA	
	SD	SG	AC	MAE	RMSE	MAE	RMSE
FedATT	✗	✗	✗	8.64	14.18	13.46	24.25
FedATT&SD	✓	✗	✗	8.58	14.64	13.31	24.46
FedATT&SG	✗	✓	✗	2.49	4.55	3.90	9.27
FedATT&AC	✗	✗	✓	8.65	14.22	13.46	24.17
SecureGraphFL	✓	✓	✓	2.39	4.48	3.77	9.22

2) Ablation study of SecureGraphFL: We conducted an ablation study to analyze the impact of the various components of SecureGraphFL on its performance, as shown in Table II. In this context, SD refers to the speed difference component, SG represents the similarity-based graph construction method, and AC denotes the actor-critic Network client selection mechanism.

As shown in Table II, the baseline FedATT model (without any additional components) achieved an MAE of 8.64 and an RMSE of 14.18 on the PeMS-D7(M) dataset. When the SD component is added (FedATT&SD), there is a slight performance improvement, with the MAE decreasing to 8.58 and the RMSE increasing to 14.64. On the other hand, incorporating the SG component (FedATT&SG) significantly improves performance, with the MAE dramatically reduced to 2.49 and the RMSE to 4.55. As expected, incorporating only the AC client selection mechanism (FedATT&AC) does not significantly impact performance in the absence of malicious clients, with MAE and RMSE remaining approximately unchanged at 8.65 and 14.22, respectively. Finally, SecureGraphFL, integrating all three components, achieves the best overall performance with an MAE of 2.39 and an RMSE of 4.48. Similar observations are consistent in the METR-LA dataset.

C. Results in the Presence of Malicious Clients

To validate the robustness of the AC client selection mechanism in SecureGraphFL against malicious or compromised clients, we conducted experiments comparing SecureGraphFL with two baseline models: FedATT&AC and FedATT. The performance results under scenarios involving varying numbers of malicious clients (0, 1, 2, and 3 out of 9 total clients) are summarized in Table III.

In scenarios without malicious clients, SecureGraphFL clearly outperforms FedATT and FedATT&AC. Specifically,

TABLE III
IMPACT OF MALICIOUS CLIENTS ON MODEL PERFORMANCE

Dataset	Model	Malicious clients (0/1/2/3)	
		MAE	RMSE
PeMS -D7(M)	SecureGraphFL	2.37/2.39/2.41/2.42	4.47/4.48/4.49/4.52
	FedATT&AC	8.6592/8.6582/8.6576/8.6545	14.35/14.34/14.33/14.30
	FedATT	8.65/1.66E7/4.99E6/2.19E6	14.32/2.43E7/8.18E6/3.36E6
METR -LA	SecureGraphFL	3.65/3.65/3.83/3.84	9.27/9.29/9.21/9.22
	FedATT&AC	13.47/13.53/13.47/13.47	24.06/23.79/24.05/24.14
	FedATT	13.46/4.19E8/1.06E8/1.60E7	24.09/6.18E8/1.65E8/2.85E7

on the PeMS-D7(M) dataset, SecureGraphFL achieves an MAE of 2.37 and RMSE of 4.47, compared to the significantly higher MAE and RMSE of approximately 8.65 and 14.35 for FedATT and FedATT&AC, respectively.

When malicious clients are introduced, the performance of FedATT deteriorates dramatically, with MAE increasing from 8.65 (no malicious clients) to approximately $1.66E7$, $4.99E6$, and $2.19E6$, and RMSE rising from 14.32 to around $2.43E7$, $8.18E6$, and $3.36E6$ for 1, 2, and 3 malicious clients, respectively. In sharp contrast, FedATT&AC, which integrates our proposed AC client selection mechanism, maintains stable performance across all scenarios, with MAE consistently around 8.65 and RMSE ranging from 14.30 to 14.35. This highlights the robustness and effectiveness of the AC mechanism, which can be independently integrated into existing FL frameworks to defend against malicious participants.

Similarly, SecureGraphFL, which also incorporates the AC mechanism, shows stable performance as the number of malicious clients increases, with MAE rising only slightly from 2.37 to 2.42 and RMSE from 4.47 to 4.52. A comparable trend is observed on the METR-LA dataset, further confirming the generalizability and reliability of the AC mechanism. These results collectively demonstrate that the AC module is a practical and effective solution for enhancing the robustness of federated learning systems against malicious clients.

In summary, extensive experimental results demonstrate that SecureGraphFL consistently achieves superior traffic prediction performance over existing FL methods and maintains robustness against malicious clients. These empirical findings further substantiate the theoretical convergence guarantees presented in Section IV-E, highlighting both the practical effectiveness and theoretical soundness of our proposed framework.

VI. CONCLUSION

This paper presented SecureGraphFL, an FL framework designed to protect data privacy while achieving high performance in traffic prediction. SecureGraphFL integrates speed difference prediction and similarity-based graph construction to enhance predictive accuracy. In addition, the actor-critic-based client selection mechanism provides robust resistance against malicious clients. Extensive experiments on the PeMS-D7(M) and METR-LA datasets demonstrate that SecureGraphFL outperforms traditional FL models such as FedAVG and FedATT, particularly under adversarial conditions.

Although SecureGraphFL demonstrates strong performance and robustness, there remain several limitations that warrant further investigation. First, the current client selection mechanism assumes access to a clean and representative validation set for each client, which may not always be feasible in real-world federated environments. This reliance could affect the accuracy of trust evaluation, especially in highly noisy or imbalanced settings. Second, the cosine similarity threshold parameter τ, used in graph construction, is manually tuned and may require extensive validation across different datasets to generalize effectively. Moreover, although the reward function in the actor-critic model has shown promising results, it is currently static and hand-designed; incorporating adaptive or self-tuning reward mechanisms could improve policy stability under dynamic and non-stationary conditions.

In addition to addressing these limitations, future work will include scaling SecureGraphFL to larger networks, validating its performance on more heterogeneous real-world traffic datasets, and further improving its generalizability across domains. We believe that these efforts will enhance the practical applicability of SecureGraphFL and contribute to the broader advancement of secure, privacy-preserving federated graph learning.

REFERENCES

[1] M. S. Irfan, S. Dasgupta, and M. Rahman, “Toward transportation digital twin systems for traffic safety and mobility: A review,” *IEEE Internet Things J.*, vol. 11, no. 14, pp. 24581–24603, Jul. 15, 2024.

[2] K. Leng and C.-F. Wu, “Traffic management optimization via IoT-enhanced cooperative vehicle-infrastructure systems,” *IEEE Internet Things J.*, vol. 12, no. 4, pp. 3493–3501, Feb. 15, 2025.

[3] H. Dui, S. Zhang, M. Liu, X. Dong, and G. Bai, “IoT-enabled real-time traffic monitoring and control management for intelligent transportation systems,” *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15842–15854, May 1, 2024.

[4] G. P. Zhang, “Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model,” *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[5] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting,” *ICLR*, 2018.

[7] B. Yu, H. Yin, and Z. Zhu, “Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting,” *IJCAI*, 2018, pp. 3634–3640.

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[9] M. H. Kabir and M. S. Islam, “Enhancing Traffic Flow and Reducing Congestion: A Smart City Approach with an IoT-based Intelligent Traffic Management System,” *2024 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, 2024, pp. 1–6.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” *AISTATS*, 2017.

[11] P. Zhao, J. Jin, and D. Yao, “K-Nearest Neighbor Algorithm and Its Application in Traffic Flow Prediction,” *Journal of Transportation Engineering*, vol. 140, no. 2, p. 04013011, 2014.

[12] C. W. J. Granger, “Investigating Causal Relations by Econometric Models and Cross-Spectral Methods,” *Econometrica*, vol. 37, no. 3, pp. 424–438, 1969.

[13] Y. Bengio, “Learning Deep Architectures for AI,” *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[14] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatial-temporal graph convolutional networks for traffic flow forecasting,” *Proc. AAAI Conf. Artif. Intell. (AAAI'19/IAAI'19/EAAI'19)*, 2019, Article no. 114, pp. 1–8.

[16] L. Skovajsova, L. Hluchý, and M. Staňo, "A Review of Multi-Objective and Multi-Task Federated Learning Approaches," *2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2025, pp. 000035–000040.

[17] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[18] C. Chen, J. Fan, and J. Liu, "Joint Optimization of Client Selection and Communication Resource Allocation in Federated Learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 5021–5033, 2021.

[19] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1698–1707.

[20] Q. Liu, G. Li, W. He, H. Zhang, and L. Wang, "Federated Learning on Non-IID Data Silos: An Experimental Study," *IEEE Access*, vol. 8, pp. 57089–57100, 2020.

[21] H. Zhang, Z. Xie, R. Zarei, T. Wu, and K. Chen, "Adaptive Client Selection in Resource Constrained Federated Learning Systems: A Deep Reinforcement Learning Approach," *IEEE Access*, vol. 9, pp. 98423–98432, 2021.

[22] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, "Freeway performance measurement system: Mining loop detector data," *Transp. Res. Rec.*, vol. 1748, no. 1, pp. 96–102, 2001.

[23] S. Han, Y. Chen, G. Chen, J. Yin, H. Wang, and J. Cao, "Multi-step reinforcement learning-based offloading for vehicle edge computing," in *Proc. 2023 15th Int. Conf. Adv. Comput. Intell. (ICACI)*, Seoul, Korea, Republic of, 2023, pp. 1–8.

[24] J. Yin, W. Hong, H. Wang, J. Cao, Y. Miao, and Y. Zhang, "A compact vulnerability knowledge graph for risk assessment," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 8, article 194, pp. 1–17, Sep. 2024.

[25] M. Wang, S. Han, G. Chen, J. Yin, and J. Cao, "Blockchain-empowered resource allocation and data security for efficient vehicular edge computing," *Proc. Web Inf. Syst. Eng. (WISE) 2023*, vol. 14306, Singapore: Springer, 2023, pp. 204–217.

[26] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in Neural Information Processing Systems*, vol. 12, 1999, pp. 1008–1014.

[27] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," *International Conference on Learning Representations (ICLR)*, 2020.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, 2014, pp. I–387–I–395.

[29] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning Private Neural Language Modeling with Attentive Aggregation," *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[30] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys)*, 2020, pp. 429–450.

[31] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning," *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 5132–5143.

[32] J. Yin, G. Chen, W. Hong, H. Wang, J. Cao, and Y. Miao, "Empowering vulnerability prioritization: a heterogeneous graph-driven framework for exploitability prediction," *Proceedings of the 24th International conference on web information systems engineering (WISE)*, Cham, Switzerland: Springer, 2023, pp. 289–299.

[33] E. Huang, Z. Zhao, J. Yin, et al., "Transformer-enhanced adaptive graph convolutional network for traffic flow prediction," *ACM Transactions on Intelligent Systems and Technology*, 2025.

[34] B. Xie et al. , "MOB-FL: Mobility-Aware Federated Learning for Intelligent Connected Vehicles," *ICC 2023-IEEE International Conference on Communications*, Rome, Italy, 2023, pp. 3951–3957.

[35] H. Wang, X. Zhou, S. Wang, X. Zhao, X. Deng, and W. Gong, "Full-Link Delivery Time Prediction in Logistics Using Federated Heterogeneous Graph Transformer," *IEEE Internet of Things Journal.*, vol. 12, no. 6, pp. 6775–6789, Mar. 15, 2025.

APPENDIX

PROOF OF THEOREM 1

The proof is structured into two parts, first showing the convergence of the actor-critic-based client selection policy, then demonstrating the convergence of global model parameters under the attention-based aggregation mechanism.

A. Convergence of actor-critic-based client selection policy

Under Assumption 2, the actor-critic parameters $\phi^{(l)}$ converge almost surely to a stationary point ϕ^*, following standard convergence arguments in actor-critic frameworks [26], [28]. The convergence result is given in Equation (25).

$$\lim_{l \to \infty} \|\pi_{\phi^{(l)}} - \pi_{\phi^*}\| = 0. \tag{25}$$

Therefore, the client selection policy stabilizes to π^*, effectively filtering out unreliable clients. As a result, the variance of the aggregated gradients from selected clients gradually diminishes across communication rounds.

B. Convergence of global parameters under attention-based aggregation

With a stable client selection policy π^*, the set $P^{(l)}$ and the associated attention weights $\alpha_m^{(l)}$ from the attention-based aggregation mechanism (Equation (16)) also stabilize. Under Assumption 1, the variance of aggregated global gradients monotonically decreases, as shown in Equation (26).

$$\lim_{l \to \infty} \delta_l = \lim_{l \to \infty} \mathbb{E}\left[\|\nabla f(\boldsymbol{\omega}_m^{(l)}) - \nabla f(\boldsymbol{\omega}_g^{(l)})\|^2\right] = 0. \tag{26}$$

Following standard convergence analysis in federated optimization [27], the expected gradient norm of the global model update satisfies the bound given in Equation (27).

$$\mathbb{E}\left[\|\nabla f(\boldsymbol{\omega}_g^{(l+1)})\|^2\right] \le \mathbb{E}\left[\|\nabla f(\boldsymbol{\omega}_g^{(l)})\|^2\right] - \gamma_l + \delta_l, \tag{27}$$

where $\gamma_l > 0$ denotes the expected optimization gain at round l, and δ_l captures the variance due to client model inconsistency, which vanishes as model divergence is minimized across selected clients.

According to Equation (26), the variance term δ_l vanishes as the training progresses. As $\delta_l \to 0$, we obtain the convergence result in Equation (28).

$$\lim_{l \to \infty} \mathbb{E}\left[\|\nabla f(\boldsymbol{\omega}_g^{(l)})\|^2\right] = 0. \tag{28}$$

This confirms that the global model converges to a stationary solution in expectation.

■

The proposed attention-based aggregation mechanism dynamically generates normalized client weights $\alpha_m^{(l)}$ based on the parameter differences $\|\boldsymbol{\omega}_m^{(l)} - \boldsymbol{\omega}_g^{(l)}\|$, ensuring smooth and bounded updates across rounds. This design generalizes classical weighted averaging and aligns with standard convergence assumptions in federated optimization [27]. Furthermore, Assumption 2 (Lipschitz continuity of the actor-critic update) is consistent with established convergence conditions in actor-critic methods [26], [28]. These properties together ensure that all theoretical conditions for convergence are satisfied within the SecureGraphFL framework.