

LOGSAFE: Logic-Guided Verification for Trustworthy Federated Time-Series Learning

Dung Thuy Nguyen, Ziyang An, Taylor T. Johnson, Meiyi Ma, Kevin Leach

Department of Computer Science, Vanderbilt University, Nashville TN

{dung.t.nguyen, ziyang.an, taylor.johnson, meiyi.ma, kevin.leach}@vanderbilt.edu

Abstract—Federated Learning (FL) offers a promising solution to the privacy challenges associated with centralized Machine Learning (ML) by enabling decentralized, collaborative training across distributed agents and devices commonly found in cyber-physical systems (CPS), such as autonomous vehicle controllers, distributed sensing platforms, and industrial control systems. However, FL remains vulnerable to poisoning attacks, where adversarial clients manipulate local data or model updates to degrade global model performance or induce unsafe behaviors in CPS that depend on these models for sensing, control, and actuation. Although substantial progress has been made in developing defenses against poisoning in FL, existing robust methods are predominantly evaluated on computer vision tasks and fall short of addressing the unique characteristics of time series data that arise naturally in CPS, including temporal dependencies, heterogeneity across devices, and large-scale deployments with potentially many compromised participants.

In this paper, we present LOGSAFE, a defense mechanism designed to mitigate poisoning attacks in Federated Time Series (FTS), even under heterogeneous client behaviors and high adversarial participation, conditions frequently encountered in large CPS. Unlike traditional model-centric defenses that assess update similarity, LOGSAFE leverages logical reasoning to evaluate client reliability by aligning their predictions with global time series patterns. Our approach extracts client-specific logical reasoning properties, hierarchically infers global temporal specifications, and verifies each client against these properties to identify and exclude malicious participants during aggregation. Experimental results on two FTS datasets show that LOGSAFE consistently outperforms existing baselines; in the best case, it reduces prediction error by 93.27 percent relative to the second best method. Our code is available at <https://github.com/judydnguyen/LOGSAFE-Robust-FTS>.

Index Terms—Federated Learning, Poisoning Attacks, Signal Temporal Logic, Time Series.

I. INTRODUCTION

Federated learning (FL) has emerged as a promising paradigm for leveraging data and computing resources from multiple clients to train a shared model under the orchestration of a central server [1]. In FL, clients use their data to train the model locally and iteratively share the local updates with the server, which then combines the contributions of the participating clients to generate a global update. Recently, FL has been demonstrated to be efficient for time-series-related tasks [2]–[5], and more recently in CPS-related domains [6]–[8], enabling secure knowledge sharing across distributed systems while preserving data privacy. Although FL has many notable characteristics and has been successful in many applications [9]–[11], recent

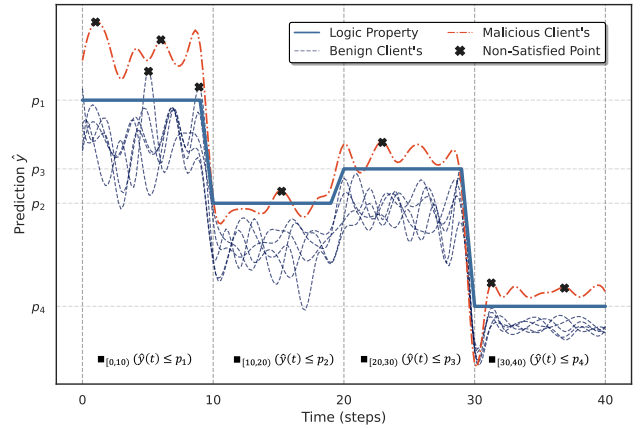


Fig. 1: Illustration of logical verification given by benign and malicious clients' predictions on the FHWA traffic dataset. Our approach learns global properties like $\hat{y}(t) \leq p_1$ for $t \in (0, 10]$, where $\hat{y}(t)$ denotes the predicted traffic flow on a highway segment. The predictions made by malicious clients often violate these properties (marked with $\times$).

studies indicate that FL is fundamentally susceptible to adversarial attacks in which malicious clients manipulate the local training process to contaminate the global model [11], [12]. In the context of AI-enabled CPS, such poisoning attacks can translate into unsafe or unstable system behaviors [13], [14]. Attacks against such systems can be broadly classified into *untargeted* and *targeted* attacks [11], [15], [16]. The former aims to deteriorate the performance of the global model on all test samples [17], [18]; while the latter focuses on causing the model to generate false predictions following specific objectives of the adversaries [11], [12].

Motivation. Prior work has attempted to curtail existing threats in FL, which we broadly classify into two directions: (1) robust FL aggregation [19]–[21] and (2) anomaly model detection [17], [22]–[24]. The former aims to optimize the aggregation function to limit the effects of polluted updates caused by attackers, whereas the latter attempts to identify and remove malicious updates. The main drawback of these methods is that polluted updates remain in the global model, reducing the model's precision while leaving the attack impact unmitigated [11]. On the other hand, most methods for identifying malicious clients proposed so far follow the majority-based paradigm in that they assume benign local model

updates are aligned and the majority compared to the malicious ones; thus, polluted updates are supposed to be outliers in the distribution of all updates. In practice, nearly all existing defenses have been designed and evaluated in computer vision contexts, such as MNIST, CIFAR-10, or ImageNet [11], [17], [19], [21], [22], [24], [25], where data modalities are spatial and static. However, these assumptions do not hold in domains such as time-series modeling or CPS, where the data are temporally correlated, non-stationary, and often highly heterogeneous across clients. Our initial investigation confirms that existing defenses are ineffective in federated time-series (FTS) settings, where temporal heterogeneity and dynamic patterns lead benign updates to diverge substantially. This behavior is much more complex than in image classification tasks, where local models typically converge toward similar feature prototypes for each class, making poisoned updates stand out as outliers. In FTS, by contrast, the lack of such alignment causes malicious updates to blend in with benign ones rather than appear as outliers. Last but not least, existing defenses lack formal verification and explainability for why a client is considered malicious, making it difficult to audit decisions and undermining trust in safety-critical CPS deployments. To this end, these limitations become especially critical in CPS, where time-series signals, including network measurements and traffic flows, play a direct role in autonomous decision-making and ensuring operational safety.

Our solution. To fill this gap, we present LOGSAFE, a defense mechanism that mitigates poisoning attacks against Federated Time Series (FTS) under especially challenging scenarios—that is, in the presence of non-IID client data and a large number of adversarial clients. Motivated by recent work demonstrating that STL (Signal Temporal Logic) is highly effective for neural networks on time-series tasks [3], [26], as it provides a formal and interpretable language to specify temporal constraints and domain knowledge that standard neural networks do not inherently capture, we design LOGSAFE to dynamically mine and verify client-specific STL properties during training, making our method practical across diverse datasets. In our setting, an STL property is a temporal specification over signals, for example, a requirement that whenever the predicted signal exceeds a safety threshold, it must return below that threshold within a fixed number of time steps. These specifications both reflect how temporal patterns are learned from the training data and provide an interpretable description of the model’s behavior over time. Our approach is orthogonal to existing model-centric defenses that detect or downweight clients based on the similarity of their local updates to one another or to a trusted reference model. Instead, LOGSAFE evaluates clients based on the satisfaction of STL properties over their predictions, focusing on temporal behavior rather than parameter space. Instead, we use STL property mining and verification to study and evaluate each client model’s dynamic behaviors, capturing traces of predictions on the given time series. The intuition behind our approach is that, after rounds of training, benign clients naturally reach a consensus of their models’ predictions, following close reasoning patterns

and sharing STL properties. Meanwhile, in the time series context, poisoning attacks disrupt the global model’s behavior, leading to abnormal logical patterns in the resulting predictions (e.g., extreme values at specific time steps). These deviations cause the corresponding logical specifications to diverge from those of the aggregated benign models (illustrated in Figure 1). First, we extract *local logical reasoning properties* — client-specific STL formulas that summarize recurring temporal patterns in the model’s predictions over time (e.g., in a traffic setting, whenever vehicle density rises above a threshold, the predicted speed drops below a safe limit within a few time steps). This is conceptually related to dynamic invariant generation in software testing (e.g., DIG [27]), but differs in that we mine STL temporal properties over prediction traces, rather than exact algebraic invariants over program executions. Then, we use unsupervised clustering to group clients based on the parameters of their extracted STL properties (e.g., thresholds, time bounds, and temporal operators). This enables us to infer *global reasoning properties* that summarize system-wide behavior from the clustered local properties in a hierarchical manner. Instead of forcing all clients to share a predefined specification, we first cluster them based on similar STL property parameters, and then perform aggregation both within and across these clusters. Using unsupervised clustering, which is commonly used to handle heterogeneous client populations in FL [3], [28], [29], helps mitigate data heterogeneity (e.g., from differences in hardware, sensors, or control algorithms) while enabling the extraction of coherent global reasoning templates. These aggregated properties are used to assess the consistency and trustworthiness of client contributions by identifying deviations from expected logical behaviors. Our empirical evaluation validates the improved effectiveness of our approach compared to existing defenses that mitigate poisoning attacks by reducing prediction error toward the vanilla case.

Contributions. We make the following contributions. (1) We present LOGSAFE — a poisoning-resistant defense for Federated Time Series applicable to AI-enabled CPSs. LOGSAFE identifies and eliminates suspicious clients that distort the global model using logical reasoning property inference and verification, and is, to the best of our knowledge, the first method to thoroughly address both targeted and untargeted poisoning attacks in FTS—even under high ratios of compromised clients and sophisticated attack strategies. (2) We systematically evaluate existing robust FL defenses in the context of federated time-series learning (FTS), identify their limitations when adapted to the time-series domain, and demonstrate that formal verification can be leveraged effectively to defend against adversarial attacks in FL. (3) We provide in-depth studies on multiple datasets, FL settings, and attack scenarios to demonstrate the superiority of LOGSAFE over state-of-the-art defense techniques, highlighting its potential as a step toward safe and verifiable learning in AI-enabled CPS.

II. RELATED WORKS

A. Poisoning Attacks in FL

Unlike traditional centralized learning, where data is aggregated at a single location, FL operates in a decentralized setting where data stays on client devices, and only model updates are communicated. This distributed nature makes FL susceptible to various types of *poisoning attacks* [30], [31]. A poisoning attack in federated learning occurs when an attacker alters the model submitted by a client to the central server during the aggregation process, either directly or indirectly, causing the global model to update incorrectly [32].

Depending on the goal of a poisoning attack, we classify poisoning attacks into two categories: (1) untargeted poisoning attacks [17], [33], and (2) targeted poisoning attacks [34]–[36]. Untargeted poisoning attacks aim to induce the learned model to exhibit a high testing error across all testing examples, thereby resulting in a denial-of-service attack. The Byzantine attack is among the most popular such attacks [17], [37]. In targeted poisoning attacks, the learned model produces attacker-desired predictions for specific test examples, e.g., predicting spam as non-spam and predicting attacker-desired labels for test examples with a particular trojan trigger (these attacks are also known as backdoor/trojan attacks [11]). In the context of a time-series task, the targeted attack can be adding imperceptible noise to the original sample such that the model predicts the incorrect class [38] or manipulating the prediction into the extreme value/specific directions [39].

To further strengthen the attack, some model poisoning attacks are often combined with data poisoning ones [11], [12]. In such attacks, adversaries intentionally manipulate their local model updates before sending them to the central server. In [40], [41], the projected gradient descent (PGD) attack is introduced to be more resistant to many defense mechanisms. In a PGD attack, the attacker projects their model on a small ball centered around the previous iteration’s global model. In addition, scaling and constraint-based techniques [18], [42] are commonly used to intensify the poisoning effect while stealthily bypassing robust aggregators.

B. Defenses against Poisoning Attacks in FL

Defending against poisoning attacks in Federated Learning requires novel approaches tailored to the unique characteristics of the FL paradigm. Existing defense strategies can be broadly categorized into (1) robust aggregation mechanisms, (2) anomaly detection, and (3) adversarial training. Robust Aggregation Mechanisms are designed to mitigate the impact of malicious updates during the model aggregation process [17], [19], [20], [25], [43]–[45]. Blanchard et al. [17] proposed the Krum algorithm, which selects updates from a majority of clients that are most similar to each other, thereby excluding potentially malicious updates. Another approach, Median-based aggregation [19], [25], takes the median of the updates from all clients, which is more resilient to outliers and adversarial manipulations. RLR [20] adjusts the global learning rate based on the sign information contained in each update per dimension and combines differential privacy noise.

The second approach is backdoor detection, which detects the backdoor gradients and filters them before aggregation [17], [21], [22], [24], [46]–[49]. To begin, Cao et al. [50] introduced FLTrust, a method that establishes a trust score for each client based on the similarity of their updates to a trusted server-side model. Updates with low trust scores are either down-weighted or excluded from the aggregation process. More recently, Zhang et al. [24] proposed predicting a client’s model update at each iteration from historical model updates and flagging a client as malicious if the received model update from the client and the predicted model update are inconsistent across multiple iterations. In addition, Nguyen et al. [22] developed an approach that combines anomaly detection with clustering, grouping similar updates and filtering out those that deviate significantly from the majority. However, the effectiveness of these methods in Federated Time Series (FTS) scenarios remains unclear, as they lack mechanisms to capture the logical reasoning properties and temporal dependencies inherent in time-series data. Existing model-centric defenses overlook these temporal relationships and dynamic patterns, limiting their ability to detect adversarial behavior accurately. Recent work has explored using formal logic to improve the interpretability and robustness of federated learning systems. An et al. [3] proposed a formal logic-enabled personalized FL framework that leverages Signal Temporal Logic (STL) to infer and enforce client-specific properties, using logical constraints as a regularization mechanism for personalization. In contrast, our approach is verification-oriented: rather than constraining local training, LOGSAFE mines global STL specifications from collective client behavior and uses their satisfaction levels as trust signals to detect and mitigate malicious updates during training. By integrating STL-based specification mining and verification directly into the learning pipeline, LOGSAFE provides an adaptive defense mechanism tailored to federated time-series learning.

III. BACKGROUND

A. Federated Time Series (FTS)

This work focuses on time series forecasting, which involves predicting future values based on historical data. Formally, let $\mathbf{X}_{1:L} = (\mathbf{x}_1, \dots, \mathbf{x}_L)^\top \in \mathbb{R}^{L \times M}$ be a history sequence of L multivariate time series, where for any time step t , each row $\mathbf{x}_t = (x_{t1}, \dots, x_{tM}) \in \mathbb{R}^{1 \times M}$ is a multivariate vector consisting of M variables or channels. Given a history sequence $\mathbf{X}_{1:L}$ with look-back window L , the goal of multivariate time series forecasting is to predict a sequence $\mathbf{X}_{L+1:L+\tau} = (\mathbf{x}_{L+1}, \dots, \mathbf{x}_{L+\tau})^\top \in \mathbb{R}^{\tau \times M}$ for the future τ timesteps. To achieve this in a decentralized and privacy-preserving manner, we consider a federated learning (FL) system where N participants collaborate to build a global model G_T that generalizes well to their future observations. In our FL system, there is a central server $\mathcal{S}$ and a client pool $\mathcal{C} = [C_1, C_2, \dots, C_N]$, where each client i has a dataset of size $|\mathcal{D}_i| = N_i$. The training procedure comprises T rounds of interaction between the server and the clients, as follows. **Step 1.** The server broadcasts the current global model G^{t-1} to

all participating clients. Note the initial model G^0 is randomly initialized. **Step 2.** Each client i updates the global model G^{t-1} using its local dataset $\mathcal{D}_i$ via mini-batch stochastic gradient descent (SGD) and sends its updated model weights back to the server. **Step 3.** The server aggregates the local updates G_i^t from all clients to generate a new global model G^t , which is then shared with the clients in the next round. This iterative process continues until the global model achieves satisfactory performance on all clients' data, ensuring it generalizes well across the diverse datasets in the federated setting.

B. Temporal Reasoning Property Inference

Signal temporal logic (STL) [51] is a precise and flexible formalism designed to specify temporal logic properties. Here, we first provide the syntax of STL, as defined below in Definition 1.

Definition 1 (STL syntax):

$$\varphi ::= \mu \mid \neg\mu \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Diamond_{[a,b]}\varphi \mid \Box_{[a,b]}\varphi \mid \varphi_1 \mathcal{U}_{[a,b]}\varphi_2$$

We denote the temporal range as $[a, b] \in \mathbb{R}_{\geq 0}$, where $a \leq b$. Additionally, let $\mu : \mathbb{R}^n \rightarrow \{\top, \perp\}$ be a signal predicate (e.g., $f(x) \geq 0$) on the signal variable $x \in \mathcal{X}$. Moreover, φ , φ_1 , and φ_2 are different STL formulas. The symbols $\Box$, $\Diamond$, and $\mathcal{U}$ are temporal logic operators, denoting “always,” “eventually,” and “until,” respectively. The “always” property requires the formula φ to be satisfied at all *future* time steps from a to b . The “eventually” property requires the formula φ to be true at some future time steps from a to b . Lastly, the “until” property requires that φ_1 is true until φ_2 becomes true.

We suggest referring to An et al. [3] for eight examples that can be expressed using STL and inferred through specification mining algorithms. However, users are not restricted to these specifications; they can define any specifications that can be parsed into STL formulas and syntax. The formal description of the logic property inference task is shown in Definition 2 below [52]. Additionally, we provide a practical example of STL property inference in Example 1.

Definition 2 (STL property inference): Given an observed fact x and a parameterized STL formula $\varphi(\alpha)$, where parameter α is unknown, the task is to find the correct value of α such that the formula φ holds for all instances of x .

Example 1 (Example of STL inference): Consider the template STL property $\varphi_m(\alpha) = \Box_{[0,5]}((x_1 \leq 0.5) \vee (\alpha \leq x_2 \leq 10))$. The task is to find a value for the unknown parameter α such that during future time stamps between 0 and 5 if the signal variable x_1 is less than or equal to 0.5, the signal variable x_2 must be greater than or equal to α and less or equal than 10.

While α can take infinitely many values, only certain ones enhance reasoning during the verification process. The choice of α influences how well a property $\mathcal{X}$ aligns with observations, shaping how accurately the STL property reflects the data. Therefore, the STL inference task can be better formulated in Definition 3, which introduces the concept of a tight bound.

Definition 3 (STL property inference under a tight bound): Given observed data $\mathcal{X}$ and a templated STL formula $\varphi(\alpha)$, where α is an unknown free parameter, the objective is to determine a value for α that yields a tightly-fitted temporal logic property. The objective is represented by the equation $\rho(\varphi, \mathcal{X}; \alpha) = \epsilon$, where a smaller positive ϵ indicates a closer alignment with the data.

The task presented in Definition 3 can be transformed into an equivalent numerical optimization problem, which can be effectively solved using both gradient-free and gradient-based off-the-shelf solvers, as demonstrated below [53].

$$\min |\epsilon| \text{ s.t. } \epsilon = p' - p \\ \text{where } \rho(\varphi(p), \mathcal{X}, t) \geq 0 \text{ and } \rho(\varphi(p'), \mathcal{X}, t) < 0$$

In a templated logic formula φ , candidate values of free parameters are denoted by p and p' and the timestamp is denoted by t . As described in the equation above, the goal is to minimize the value of $|\epsilon|$, which measures the difference between a satisfactory parameter value and an unsatisfactory one. Since the STL robustness function is non-differentiable at zero, alternatives such as the “tightness metric” in Jha et al. [53] can be utilized to address this problem. We further provide Example 2 to illustrate this definition better.

Example 2 (Example of tight-bound inference): Consider the task from Definition 3 with the example of finding α that satisfies $\varphi_m(\alpha) = \Box_{[0,5]}((x_1 \leq 0.5) \vee (\alpha \leq x_2 \leq 10))$ for the 5-step sequence $\mathcal{X} = ((0.4, 4), (0.45, 5), (0.55, 6), (0.75, 7), (1.0, 9))$. The goal is to find p to minimize the value of $|\epsilon|$.

We evaluate $p = 5$, $p = 6$, and $p' = 7$. For $\alpha = 5$ and $\alpha = 6$, the formula is satisfied at every time step. However, for $\alpha = 7$, $\rho(\varphi_m(7), \mathcal{X})$ is negative at $t = 3$, indicating a violation. To minimize the discrepancy $\epsilon = p' - p$, we seek smaller ϵ . For instance, $\epsilon = 1$ when $p = 6$ and $p' = 7$, compared to $\epsilon = 2$ for $p = 5$ and $p' = 7$. Thus, $\alpha = 6$ provides a tighter fit than $\alpha = 5$, reducing the mismatch between the STL formula and observed data, ensuring better alignment with the logic specification.

C. Threat Model

Attacker Capabilities: We consider an FL system consisting of a central server $\mathcal{S}$ and a pool of clients $\mathcal{C}$ with size N , where a fraction ϵ of these clients are malicious, i.e., $\epsilon = \frac{|\mathcal{C}_p|}{N}$. These malicious clients can operate independently (non-colluded) or be controlled by a single adversary A (colluded). The objective of the malicious clients is to compromise the global model G_T by introducing poisoned time series data or tampering with model updates. The primary goal is to manipulate the poisoned model G'_T to output the predictions in the poisoning objective that the adversaries expect. In our study, we assume that malicious clients can render *targeted* or *untargeted* attacks. In the untargeted case, the objective is to produce predictions with a high testing error indiscriminately across testing examples, which is represented as: $\max(\mathcal{L}(G_T(\mathcal{X}), \mathcal{Y}))$ where $G_T(\mathcal{X})$ denotes the predictions of the model G_T on input $\mathcal{X}$, and $\mathcal{Y}$

represents the true labels. Meanwhile, in the targeted case, the goal is to minimize the loss between the model's predictions and the adversarial target labels: $\min(\mathcal{L}(G_T(\mathcal{X}), \hat{Y}))$, where $\hat{Y}$ is the adversarial target label. In addition, we consider malicious clients with *white-box* attack capabilities. This assumption implies that these clients can manipulate their local training data, injecting poisoned time-series data $\mathcal{D}_p$ into the training set $\mathcal{D}_i^t$ of each compromised client i . They can also control the local training process and modify model updates before sending them to the server for aggregation, which is more challenging for the defender.

Defender Capabilities: Following existing FL defenses, we assume the defender, represented by the server, has the following capabilities. The defender can only access local model updates after the local training process and has no prior knowledge of the attack strategies employed by adversaries. Furthermore, the defender can access only a limited portion of unlabeled clean data, rather than the full training set. It is important to note that the assumption of clean data is not unique to our method and has been widely adopted in FL-related research fields [23], [54], [55]. Additionally, we assume the server is honest and does not engage in privacy-breaching attacks. Finally, we assume the server has sufficient computational resources, which aligns with real-world scenarios [55]. The defender's primary objective is to minimize malicious clients' influence on the global model by reducing the empirical prediction error.

IV. APPROACH

Figure 2 illustrates the four components that comprise LOGSAFE: (i) local logic inference; (ii) global logic inference; (iii) global property verification, and (iv) malicious client detection. In this section, we will present in detail how each step is performed.

A. Local Logic Inference

LOGSAFE operates in an FL scenario, where each client completes local training to compute an update that is communicated each training round with an aggregating server. During each such communication round, we mine STL specifications from each client after completing its local training. We employ stochastic gradient descent (SGD) [3] as the optimization algorithm to update the models as: $G_i^t \leftarrow \text{SGD}(G_i^t, D_i, \eta)$, where η is the learning rate.

Given a local model of the client C_i during round t , i.e., G_i^t , and $\mathcal{D}_v$ be the unlabeled validation data owned by the server, our method generates the logic property $\varphi(p^i)$ for each participating client based on their prediction $\mathcal{Y}_i^t$ on dataset $\mathcal{D}_v$. Our method uses Equation III-B to infer a logic reasoning property φ from the client prediction. Recall that any locally inferred logic reasoning property must be satisfied by every data point in the client's prediction. Moreover, any STL formula can be represented by its equivalent Disjunctive Normal Form (DNF), which typically has the form of $P \vee Q \vee R \vee \dots$. Each clause within a DNF formula consists of variables, literals, or conjunctions (e.g., $P := p_1 \wedge \neg p_2 \wedge \dots \wedge p_n$).

Essentially, the DNF form defines a range of satisfaction where any clause connected with the disjunction operator satisfies the STL formula φ . After this step, each client C_i has a logical property represented $\varphi(p^i) := \varphi_1(p^i) \wedge \varphi_2(p^i) \wedge \dots \wedge \varphi_n(p^i)$ extracted by logic inference.

B. Global Logic Inference

Given a set of local logic properties $\mathcal{P} = \{\varphi(p^1), \varphi(p^2), \dots, \varphi(p^m)\}$ at the t^{th} round, the server aggregates them to construct the global property P_g that all clients should satisfy. The core idea involves multi-level aggregation, where clients with similar properties are first clustered to derive a global property for each cluster. These cluster-level properties are then aggregated to form the final global logic properties. This approach helps mitigate targeted and untargeted attacks, as malicious clients with properties considerably different from benign ones can be isolated, reducing their impact on the global property P_g .

To categorize samples based on their properties, we use FINCH [56], a hierarchical clustering method that identifies groupings in the data based on neighborhood relationships, without requiring hyperparameters or predefined thresholds. FINCH is an unsupervised clustering algorithm particularly suitable for scenarios with an uncertain number of clusters, making it ideal for clustering client properties. Using FINCH, we group clients based on their local properties $\{P_i\}$ as follows: $\mathcal{P} \xrightarrow{\text{FINCH}} \Gamma_{L_k} = \{\xi_i\}_{i=1}^K \wedge r_{i,k} \in \mathbb{R}^{m \times K}$. Here, $\{\xi_i\}_{i=1}^K$ is a set of clusters, and $\mathbf{r}$ is the cluster assignment metric, where $r_{i,k} \in \mathbb{R}^{m \times K}$ and $r_{i,k} = 1$ if $i \in k$ else $r_{i,k} = 0$. This technique implicitly clusters clients with similar logical properties, as clients with similar temporal reasoning properties are grouped, while those with differing properties are not combined. A key advantage of the LOGSAFE framework is its dynamic assignment of client properties to clusters, allowing the aggregation process to adapt to changes in logic properties over time. Next, we discuss how to extract cluster property for each cluster ξ_i .

Cluster Property Inference. For a given cluster $\xi_i = \{\varphi(p^j) \mid \forall j \in [1, m] \wedge r^{j,i} = 1\}$, where local property $\varphi(p^j)$ is from client j^{th} in cluster ξ_i , the corresponding cluster property P_{L_i} is computed as:

$$P_{L_i} := \varphi(\bar{p}), \quad \bar{p} := \frac{1}{|\xi_i|} \sum_{j=1}^{|\xi_i|} p_j \quad (1)$$

In Equation 1, $\bar{p}$ represents the average of the properties inferred by the local clients within the cluster, and P_{L_i} denotes the cluster property expressed in the same STL (Signal Temporal Logic) syntax as the local models. The rationale behind this approach is to derive a representative property for the cluster by aggregating the individual properties of the clients. By averaging the properties, we create a cluster property that captures the commonalities among clients in the cluster, thus providing a robust and collective representation of their logic properties.

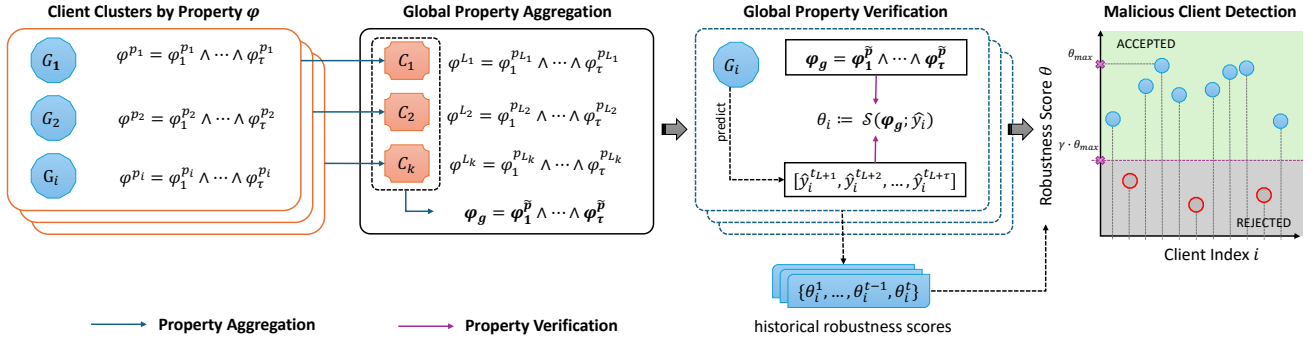


Fig. 2: Overview of LOGSAFE. For each training round, LOGSAFE first conducts local logic inference to acquire the local reasoning properties; which are then used as a criterion to cluster clients. Global property φ_g is calculated by aggregation of clustered properties. Using φ_g , the server $\mathcal{S}$ verifies the satisfaction scores θ for each client and uses them to determine the suspicious updates to be removed from the final aggregation.

Global Property Inference. Given a set of cluster properties $\{P_{L_i} = \varphi(p^{L_i}) | \forall i \in [1, K]\}$, the global property is then calculated as the median of these parameters from cluster properties, which is:

$$P_g := \varphi(\tilde{p}), \quad \tilde{p} := \text{Med}(p^{L_i}), \forall i \in [1, K], \quad (2)$$

Clustering clients is a well-established approach for handling heterogeneous and non-IID data, where local properties' parameters vary greatly [29], [57]. Inspired by this, we construct the global property by averaging cluster properties rather than individual local ones. This ensures the global property better represents each cluster's data, enhancing model generalization within groups and avoiding a uniform approach that may not fit diverse client distributions. By leveraging the median, which is less sensitive to outliers, we reduce the influence of adversarial clients with properties that deviate sharply from benign ones. This aggregation process aligns to build a global model that is resilient to both targeted and untargeted attacks, as it reduces the impact of malicious clients whose properties deviate dramatically from the benign ones.

C. Global Property Verification

Given a global property $\varphi(\tilde{p})$, this component assesses the level of satisfaction for each percentage of network predictions, denoted by $\hat{\mathcal{Y}} = (y_{L+1}, \dots, y_{L+\tau})$, with respect to a specified property φ . Given an input history $\mathcal{X} = (x_1, \dots, x_L)$, the temporal logic formula φ is defined over the resulting predicted trace $\hat{\mathcal{Y}}$, and its satisfaction value quantifies the degree to which $\hat{\mathcal{Y}}$ over the next τ time steps satisfies φ under the condition that $\mathcal{X}$ is provided as the preceding context. This framework enables us to quantify the degree to which the network's predictions align with the desired property. We formalize the concept of a robustness score θ in Definition 4.

Definition 4 (Robustness Score θ): Given a network's prediction $\hat{\mathcal{Y}} = (y_{L+1}, \dots, y_{L+\tau})$, logical property φ and scoring function $\mathcal{S}$, robustness score θ quantify the degree to which the predicted sequence $\hat{\mathcal{Y}}$ adheres to the specified property φ . Formally, $\theta := \mathcal{S}(\varphi, \hat{\mathcal{Y}}) = \{\rho(\varphi, \mathcal{Y}, t) | \forall t \in [L+1, L+\tau]\}$,

where $\rho(\varphi, x, t)$ maps an STL formula φ , a signal trace $\mathcal{Y}$ at time t , to a value.

Depending on the chosen scoring function, ρ returns either a binary satisfaction status or a real-valued satisfaction level [51], [53]. We provide a practical example of STL property verification in Example 3.

Example 3 (Example of STL verification): Given the STL property $\varphi = \Box_{[0,5)}((x_1 \geq 0.2) \wedge (x_1 \leq 2.5) \wedge (x_2 \geq 6) \wedge (x_2 \leq 10))$, we evaluated the robustness score θ using predicted sequences $\hat{\mathcal{Y}} = [(0.4, 4), (0.45, 5), (0.55, 6), (0.75, 7), (1.0, 9)]$. The scoring function $\rho(\varphi, \mathcal{Y}, t)$ calculates the degree of satisfaction at each time step by checking whether both x_1 and x_2 lie within their specified bounds and returns a boolean value. The final score is $\theta = [\perp, \perp, \top, \top, \top]$, where $\top$ means the property is satisfied, while $\perp$ means the property is violated at that time step.

For each training round, after the robustness score is calculated using Definition 4, the server collects a set of scores for each client, denoted as $\theta^t := \{\theta_1^t, \theta_2^t, \dots, \theta_K^t\}$. To convert the robustness score θ from qualitative semantics (i.e., boolean values) to numerical values, we compute the fraction of "True" signal predicates (i.e., the fraction of $\top$ in θ^t). A higher-scoring θ indicates that a client's predictions align with global properties or the majority of benign clusters, reflecting model reliability and suggesting the client is likely benign. Conversely, lower-scoring θ highlights discrepancies in the client's model or data, potentially signaling anomalies or malicious intent, warranting further investigation. Over time, we hypothesize that benign clients increasingly agree on the training objective, leading to more similar models and higher alignment in their predictions on the same validation set. As a result, these clients are expected to achieve higher θ scores, reflecting consistent adherence to the global property and reinforcing their benign behavior.

To further enhance the reliability of the robustness score, we use historical information to update scores cumulatively. Using historical information has been used in related works [21], [43], [45] and demonstrated its potential improvement. Specifically, the score for client i is updated as $\theta_i = \frac{f_i-1}{f_i} \theta_i + \frac{1}{f_i} \theta_i^t$,

where θ_i is the cumulative score, θ_i^t is the current round's score, and f_i represents the number of training rounds client i has participated in up to round t . By using cumulative scores, the method evaluates each client's overall consistency and performance across multiple rounds, reducing the impact of erratic single-round results. This trustworthy score helps identify and exclude malicious clients from aggregation during each training round.

D. Malicious Client Detection

To identify and filter out malicious clients, we use a binary mask $\mathcal{M} \in \mathbb{R}^m$, where each element $\mathcal{M}_i$ indicates whether a client i is considered malicious. The mask is defined as follows:

$$\mathcal{M}_i = \begin{cases} 1 & \text{if } \theta_i \geq \gamma \cdot \max(\theta_j), \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Here γ is a threshold parameter determining the cutoff for malicious model filtering. Note that the trustworthy scores θ are normalized before applying Equation 3. The rationale behind this approach is that models producing predictions with significantly lower robustness scores are more likely to be compromised or adversarial, especially if they are in the minority. The assumption is that the properties of poisoned models, which are often outliers or deviate from the global property, will exhibit lower scores than most benign clients. The robustness threshold for each training round is determined based on the distribution of the robustness scores across all clients. This threshold helps distinguish between reliable and unreliable client updates. To ensure the integrity of the global model, only updates from non-malicious clients are used for aggregation. The global model w_t at round t is updated as follows:

$$G_t = \mathcal{A}(G_{t-1}; \{\nabla_i \mid \forall i \in [1, m] \text{ and } \mathcal{M}_i = 1\}), \quad (4)$$

where $\mathcal{A}$ represents the aggregation function used by the server to combine the updates from the selected non-malicious clients. In this formulation, ∇_i denotes the model update from client i . By excluding updates from clients marked as malicious (i.e., those with $\mathcal{M}_i = 0$), the server can ensure that the global model w_t is not adversely affected by compromised or unreliable client contributions. We hypothesize that LOGSAFE enhances the robustness of the global model and helps maintain its performance by focusing on contributions from trustworthy clients.

V. EXPERIMENTS

In this section, we empirically evaluate the performance of LOGSAFE under various poisoning attack scenarios, including both untargeted and targeted attacks. We compare LOGSAFE against state-of-the-art FL defenses, including Krum/Multi-Krum [17], FoolsGold [21], RFA [19], ARAGG [43], RLR [20], FLAME [22], and FLDetector [24]. All experiments are conducted on an Amazon EC2 g5.4xlarge instance equipped with one NVIDIA A10G Tensor Core GPU, 16 vCPUs, 64 GB of memory, and 24 GB of dedicated GPU

memory. To thoroughly evaluate the robustness and generalization capability of LOGSAFE, we organize our study around the following research questions:

- **RQ1:** Can LOGSAFE robustly defend against both untargeted and targeted poisoning attacks?
- **RQ2:** Does LOGSAFE generalize across diverse model architectures and FL aggregators?
- **RQ3:** How does LOGSAFE behave under varying adversarial strengths, including different attack ratios and strategies?

A. Experiment Setup

Dataset descriptions. Following the previous FTS studies [2], [3], we conduct experiments using two time-series datasets, which are LTE Physical Downlink Control Channel (PDCCH) measurements [2] and Federal Highway Administration [58]. The PDCCH dataset was collected from three different base stations in Barcelona, Spain, which is publicly accessed by Perifanis et al. [2]. We further separate data from these three stations into 30 sub-clients to simulate the FL environment. Following the settings by Perifanis et al. [2], the objective is to predict the first five measurements for the next timestep using as input the observations with a window of size $\tau = 10$ per base station. The FHWA dataset is a real-world dataset of highway traffic data. We obtained a publicly available dataset from the Federal Highway Administration (FHWA 2016) and preprocessed hourly traffic volume from multiple states. After preprocessing and excluding low-quality and missing data, we continue with the final data from 15 states, this setting is followed [3]. The learning objective is to predict the traffic volume for the next $\tau = 24$ consecutive hours based on the past traffic volume at a location over the previous five days. For these two datasets, we focus on using operational range (ref. [3]) for logical reasoning inference and qualitative semantics for verification.

Attack settings. We argue that existing methods typically handle either targeted or untargeted attacks, but not both. However, from a defender's perspective, it is difficult to anticipate which type of attack an adversary will launch, making it more practical to develop robust FL methods that can effectively handle both. Therefore, we compare LOGSAFE with other defenses under both (i) *untargeted attacks* and (ii) *targeted attacks*. Regarding *untargeted attacks*, we consider Gaussian Byzantine attack followed by Fang et al. [18], where for a Byzantine client i in iteration r , the message to be uploaded is set to follow a Gaussian distribution, i.e., $\nabla_i^t \sim \mathcal{N}(0, \sigma_G^2)$. This attack randomly crafts local models, which can mimic certain real-world scenarios where compromised devices might produce erratic or unexpected updates. Regarding *targeted attacks*, we consider the case where malicious clients use a flip attack to manipulate the training data, i.e., data poisoning. Specifically, the adversary aims to disrupt a machine learning model by modifying a subset of data points to their extreme target values [39]. This attack is conducted by computing the distance of each target value $\mathcal{Y}_t \in \mathcal{Y}$ from the nearest boundary (i.e., $\mathcal{Y}_{min}, \mathcal{Y}_{max}$) of the feasibility domain, then selecting the

points with the largest distances and flipping their target values to the opposite extremes. Under this attack, the victim model shifts the target variable to the other side’s extreme value of the feasibility domain¹.

In FL, to make poisoning attacks stronger and more stealthy against defenses, we evaluate *white-box* attacks, where the adversary can combine data poisoning (as discussed above) with model poisoning techniques. Consequently, when evaluating targeted attacks, we examine four strategies: (i) Targeted Attack (without model poisoning), (ii) PGD Attack (Targeted Attack combined with PGD), (iii) Constrain-and-Scale Attack (Targeted Attack combined with Constrain-and-Scale), and (iv) Model Replacement Attack (Targeted Attack combined with Model Replacement) [11], [12].

Defense baselines. We implement seven representative FL defenses for byzantine attacks and targeted attacks, including Krum/Multi-Krum [17], RFA [19], ARAGG [43], Fools-gold [21], FLAME [22], RLR [20], and FLDetector [24].

Evaluation metrics. We evaluate the model performance using the Mean Absolute Error (MAE) and Mean Squared Error (MSE). The considered metrics are defined as follows: $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$; and $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$. The objective of the defender is to minimize both MSE and MAE.

Simulation setting. We simulate an FL system with a total of N clients; in each training round, the server randomly selects $k\%$ of clients to participate, a process known as client sampling. Unless otherwise specified, N is set to 30 and 100 for the PDCCH and FHWA datasets, respectively, with k fixed at 50%. The number of communication rounds is set to 20 for both datasets, with 3 local training epochs per round. For all experiments and algorithms, we use SGD with consistent learning rates and a batch size of 128. A vanilla RNN model is employed as the backbone network.

B. Experimental Results

1) **RQ1:** *Can LOGSAFE robustly defend against both untargeted and targeted poisoning attacks?* : First, we report the effectiveness of our defense and other baselines against both untargeted and targeted attacks, measured by MSE and MAE metrics, in Table I and Table II.

Robustness on Untargeted Attack. We first evaluate our method against Byzantine attacks and compare it with state-of-the-art defenses under full and partial participation scenarios. In the full scenario, all clients participate in training, while in the partial scenario, the server S randomly selects m clients each round. Results are shown in Table I, with the best highlighted in bold. Most defenses fail to mitigate the effect of malicious clients. Our method achieves the best performance in both settings across the PDCCH and FHWA datasets, reducing MSE by up to 78.84% compared to the second-best baseline. In contrast, other baselines, except RFA, show notable degradation on the FHWA dataset, a more practical and heterogeneous scenario. Since FTS differs from classification tasks, client data may exhibit diverse logical

patterns, and even benign clients’ gradients are not necessarily close. Moreover, methods such as Krum, RFA, FLAME, and ARAGG perform reasonably well in the full setting but degrade in partial scenarios. Overall, our method demonstrates superior generalization, maintaining robust performance while others show variability or degradation, especially when only a subset of clients participates each round.

Robustness on Targeted Attack. Table II demonstrates our method’s effectiveness under four white-box targeted attacks: Targeted, PGD, Constrain-and-scale (CnS), and Model Replacement. Our method consistently achieves the lowest MSE and MAE across all settings, outperforming the second-best baseline by 20.69% on PDCCH and 93.27% on FHWA in MSE. Existing baselines exhibit unstable performance across datasets and degrade under higher heterogeneity, as pairwise distance-based detection becomes ineffective when even benign updates deviate significantly. When adversaries combine data and model poisoning, baselines show mixed results: RLR mitigates CnS but struggles with Targeted and PGD attacks, while FLDetector handles CnS better than PGD or Model Replacement. Defenses relying on fixed metrics such as Euclidean or cosine distance remain susceptible to white-box attacks. In contrast, our method maintains robust performance across all scenarios. As shown in Figure 4, benign clients consistently achieve higher and more stable robustness scores, while malicious clients exhibit lower, fluctuating scores, a divergence that grows over training rounds, confirming the score’s effectiveness in detecting malicious clients.

RQ1 Summary. LOGSAFE shows high effectiveness, consistently surpassing existing defenses against both untargeted and targeted attacks, even when adversaries combine multiple model poisoning strategies to strengthen their impact.

2) **RQ2:** *Does LOGSAFE generalize across diverse model architectures and FL aggregators?* : To answer this question, we evaluate the stability of our method across different FL configurations by varying two key factors that have been shown to strongly influence defense performance in FL [22], [41]: (i) the model architecture and (ii) the FL aggregator, as illustrated in Figure 3 and Figure 5.

Effect of Different Model Architectures. In this experiment, we assess different model architectures to verify the applicability of our method across various settings. We evaluated three versions for each architecture: the vanilla model (no attack), the poisoned model (attack without defense), and the defended model (attack with defense LOGSAFE applied) and presented the results in Figure 3. As shown, adding LOGSAFE substantially mitigates poisoning effects caused by malicious clients, thus reducing MSE and MAE by over 99.6% and 93.3%, respectively. In the targeted attack scenario, LOGSAFE also shows its efficacy with different model architectures. Indeed, while the poisoned model exhibits performance degradation, using LOGSAFE with an LSTM backbone reduces the MSE by 66.7% and the MAE by 59.1%. In all model architectures, LOGSAFE consistently mitigates the effect of poisoning attacks as close to the one of a vanilla model. To this end, LOGSAFE can mitigate both untargeted and targeted attacks

¹We reproduce this attack using source code published by Muller et al. [39].

TABLE I: Defense efficacy under Byzantine Attacks with FHWA and PDCCH datasets.

Methods	PDCCH Dataset				FHWA Dataset			
	Full		Partial		Full		Partial	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Krum	.0066	.0455	.0066	.0478	.0139	.0741	.0107	.0661
Multi-Krum	.8706	.7428	.0105	.0910	.5991	.6316	6.326	2.062
RFA	.0053	.0389	.0071	.0694	<u>.0027</u>	<u>.0338</u>	<u>.0052</u>	<u>.0495</u>
ARAGG	.0056	.0518	.0071	.0576	.0252	.1132	.0487	.1691
FoolsGold	.0591	.1650	.0575	.2912	.7358	.7491	.4878	.6096
FLAME	<u>.0047</u>	<u>.0387</u>	<u>.0058</u>	<u>.0563</u>	.0134	.0703	.0133	.0712
RLR	33.21	5.545	.0111	.0652	.3283	.4439	1.488	.9757
FLDetector	.0168	.0944	.0078	.0750	.3959	.4986	.0295	.1159
Ours	<u>.0045</u>	<u>.0378</u>	<u>.0045</u>	<u>.0376</u>	<u>.0021</u>	<u>.0351</u>	<u>.0011</u>	<u>.0227</u>

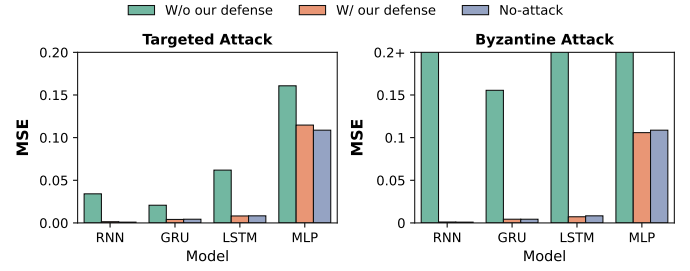


Fig. 3: Zoom-in robustness of our method when applied with different model architectures.

TABLE II: Defense efficacy under targeted attacks with different model poisoning techniques under the PDCCH and FHWA datasets.

Methods	PDCCH								FHWA							
	Targeted		PGD		CnS		MR		Targeted		PGD		CnS		MR	
	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)	MSE ($\downarrow$)	MAE ($\downarrow$)
Krum	.0066	.0478	.0066	.0478	.0107	.0854	.0066	.0472	.0257	.1021	.0257	.1021	.0049	.0475	.0226	.0882
Multi-Krum	.0105	.0910	.0105	.0910	.0058	.0550	.0105	.0909	.0279	.1377	.0279	.1377	1.001	.9080	.0287	.1396
RFA	<u>.0071</u>	<u>.0694</u>	<u>.0071</u>	<u>.0694</u>	.0059	.0581	.0072	.0695	<u>.0126</u>	<u>.0823</u>	<u>.0135</u>	<u>.0874</u>	<u>.0135</u>	<u>.0874</u>	<u>.0124</u>	<u>.0805</u>
ARAGG	.0106	.0669	.0076	.0520	.0065	.0491	.0127	.0856	.2211	.4215	.2433	.4366	.5772	.6744	.2207	.4212
FoolsGold	.0575	.2091	.0575	.2091	.0521	.2155	.0568	.0564	.1333	.5459	.1333	.2960	2.394	1.368	.1003	.2576
FLAME	.0058	.0563	.0058	.0563	.0053	.0482	<u>.0058</u>	.0563	.0297	.1221	.0297	.1221	.0358	.1397	.0297	.1221
RLR	.0111	.0652	.0111	.0652	.0069	<u>.0466</u>	.0061	.0456	.0254	.1154	.0254	.1154	1.325	1.032	.0260	.1214
FLDetector	.0078	.0750	.0078	.0750	<u>.0048</u>	<u>.0460</u>	.0080	.0783	.0233	.1239	.0233	.1239	.0194	.0971	.0395	.1647
Ours	<u>.0046</u>	<u>.0380</u>	<u>.0046</u>	<u>.0380</u>	<u>.0046</u>	<u>.0385</u>	<u>.0046</u>	<u>.0380</u>	<u>.0014</u>	<u>.0318</u>	<u>.0014</u>	<u>.0318</u>	<u>.0012</u>	<u>.0261</u>	<u>.0014</u>	<u>.0318</u>

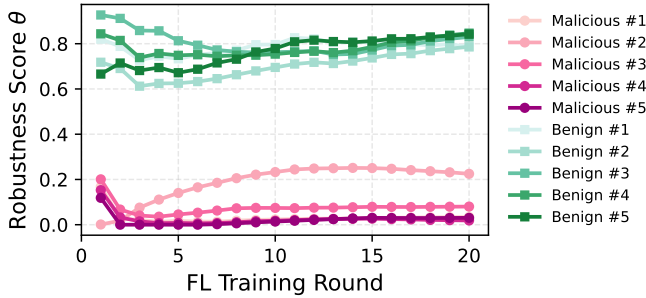


Fig. 4: Selected clients' robustness scores θ over training rounds on the FHWA dataset.

and provide robust performance improvements across various architectures. **Effect of Different FL Aggregators.** In this experiment, we evaluate the performance LOGSAFE when combined with various FL aggregators with Byzantine and targeted attacks. We select five aggregators, including FedAvg, FedProx [59], Scaffold [60], FedDyn [61], FedNova [62], and denote their LOGSAFE-enhanced versions as Aggregator+. The result is presented in Figure 5, which reveals a sustainable contrast in performance between baseline methods and their corresponding LOGSAFE-enhanced versions. When integrated with different baselines, LOGSAFE consistently mitigated errors from attacks. Across all aggregators, it reduced MSE by over 99%, yielding error levels close to the no-attack case.

RQ2 Summary. LOGSAFE can be seamlessly integrated with different architectures and aggregation strategies while maintaining stable and robust defense performance.

3) **RQ3:** How does LOGSAFE behave under varying adversarial strengths, including different attack ratios and strate-

gies?: We assess the robustness of LOGSAFE under different adversarial strengths, where the attack ratio denotes the fraction of compromised clients in the federation; a higher ratio corresponds to a stronger adversary. In addition, we study a challenging **adaptive attack** in which the adversary is assumed to know the server-side defense strategy and crafts its poisoned updates accordingly to maximize the chance of bypassing our logic-guided verification.

Effect of Different Attack Ratios. Table III demonstrates the efficacy of different defenses against various backdoor attacks under varying attack ratios $\epsilon \in [0.05, 0.1, 0.2, 0.3, 0.5]$ on both PDCCH and FHWA datasets. As ϵ increases, the scenario becomes more challenging due to the decreasing ratio of benign to malicious clients, with $\epsilon = 0.5$ being the most difficult case where benign and malicious clients are equal in number. The results reveal remarkable variations in baseline robustness: most methods deteriorate at higher ratios, with methods such as Multi-Krum and FoolsGold experiencing severe degradation, and Multi-Krum's MSE escalating more than 10 times as ϵ rises from 0.05 to 0.5. In contrast, LOGSAFE consistently achieves the lowest MSE and MAE across all attack ratios, maintaining stable performance even at $\epsilon = 0.5$, demonstrating its robustness and reliability.

Adaptive Attacks. We evaluate LOGSAFE under an adaptive attack setting, where we assume the adversary knows the defense mechanism employed by the server, enabling it to attempt to circumvent it. Since LOGSAFE's core defense lies in its ability to leverage global reasoning properties to evaluate clients' trustworthiness, we developed adaptive attacks based on the premise that adversaries could infer logical properties from the global model to obscure their suspicious behavior and

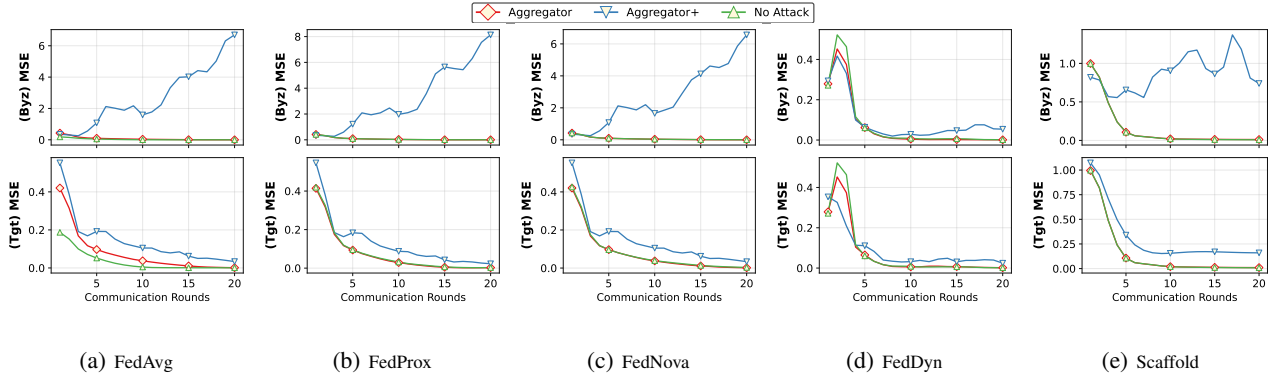


Fig. 5: Evaluation of defense efficacy when integrating our method with various aggregators. Aggregator+ indicates the combination of our method with the corresponding aggregator.

TABLE III: Comparing the performance of different defenses under various attack ratios ϵ with PDCCH and FHWA datasets.

Methods	PDCCH										FHWA									
	$\epsilon = 0.05$		$\epsilon = 0.1$		$\epsilon = 0.2$		$\epsilon = 0.3$		$\epsilon = 0.5$		$\epsilon = 0.05$		$\epsilon = 0.1$		$\epsilon = 0.2$		$\epsilon = 0.3$		$\epsilon = 0.5$	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Krum	.0078	.0505	.0070	.0485	.0051	.0435	.0072	.0496	.0068	.0425	.0268	.0965	.0090	.0619	.0170	.0661	.0031	.0356	.0065	.0526
Multi-Krum	.0044	.0379	.2987	.3869	.7665	.7005	1.178	.8686	7.643	2.181	.1347	.2897	1.277	.8875	6.326	.0622	7.730	2.227	18.05	3.321
RFA	.0048	.0388	.0051	.0392	.0053	.0398	.0052	.0395	.0048	.0388	.0045	.0458	.0036	.0382	.0052	.0495	.0037	.0426	.0069	.0547
ARAGG	.0063	.0480	.0067	.0536	.0106	.0669	.0138	1.037	.0138	.0902	.0067	.0691	.0375	.1574	.2211	.4215	.3451	.5220	.8945	.8493
FoolsGold	.0076	.0492	.0080	.0605	.0765	.1950	.0551	.1594	.0424	.1394	.0746	.2304	.3137	.4797	.4878	.6096	.6487	.7124	.9113	.8498
FLAME	.0046	.0394	.0047	.0389	.0050	.0391	.0050	.0393	.0780	.2025	.0095	.0626	.0087	.0580	.0133	.0712	.0153	.0753	5.844	1.969
RLR	.0978	.2679	2.011	1.117	2.515	1.403	2.456	1.367	39.88	5.820	.1820	.3551	1.328	.9699	1.488	.9757	7.001	2.096	3.689	1.562
FLDetector	.0047	.0393	.0043	.0408	1.862	1.170	1.788	1.156	10.97	2.650	.0026	.0347	.0930	.2117	.0295	.1159	.0612	.1813	9.460	2.426
Ours	.0047	.0384	.0047	.0387	.0045	.0376	.0045	.0382	.0047	.0395	.0014	.0267	.0017	.0285	.0011	.0227	.0011	.0243	.0019	.0324

TABLE IV: LOGSAFE on adaptive attacks. The results are obtained with the default setting we use in the main experiment.

Ratio	MSE		MAE	
	Adaptive	No-Adaptive	Adaptive	No-Adaptive
0.05	0.0017	0.0014	0.0291	0.0267
0.1	0.0015	0.0017	0.0260	0.0285
0.2	0.0014	0.0011	0.0274	0.0227
0.3	0.0023	0.0011	0.0332	0.0243
0.5	0.0026	0.0019	0.0362	0.0324

constrain their models to produce predictions that satisfy these properties. As shown in Table IV, adaptive attacks are largely ineffective against LOGSAFE. Even under the most adversarially favorable scenario — 50% of the clients are malicious — the attacks achieve only limited success in evading the defense, with errors still half those of the second-best baseline (RFA), a scenario generally considered impractical in typical federated learning systems. The failure of these adaptive attacks stems from a trade-off: as poisoned models attempt to satisfy the logical properties required for trustworthiness, they inadvertently cause malicious clients’ models to align with the behavior of benign clients, diminishing the attack’s impact on the global model. Thus, while designing an adaptive attack capable of evading LOGSAFE is theoretically possible, it simultaneously introduces errors that undermine the adversary’s objectives.

RQ3 Summary. LOGSAFE remains robust under varying adversarial strengths and adaptive settings, consistently achieving the lowest error across all attack ratios, even with 50% malicious clients. Adaptive attacks that try to bypass the server defense also fail to compromise the defense.

VI. CONCLUSION

In this study, we address a critical gap in FL robustness by focusing on the specific challenges posed by poisoning attacks within the FTS domain. Mainstream FL defenses have primarily targeted computer vision applications and showcase their limited efficacy with FTS. We introduce LOGSAFE, a defense mechanism designed to tackle these challenges effectively. LOGSAFE leverages a unique approach that combines formal logic reasoning with hierarchical clustering to evaluate client trustworthiness and align predictions with global time-series patterns. This method diverges from traditional model-centric defenses by focusing on the logical consistency of client contributions, which enhances its ability to detect and mitigate adversarial behaviors. The results from our experiments on two distinct datasets and various attack settings demonstrate that LOGSAFE outperforms existing baseline methods, confirming its efficacy in improving the robustness of FL systems for time series applications.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grant Nos. 2443803 and 2427711, the NSA Science of Security (SoS) program, and the ARPA-H DIGIHEALS program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the U.S. Government.

REFERENCES

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.
- [2] V. Perifanis, N. Pavlidis, R.-A. Koutsiamanis, and P. S. Efraimidis, "Federated learning for 5g base station traffic forecasting," *Computer Networks*, vol. 235, p. 109950, 2023.
- [3] Z. An, T. T. Johnson, and M. Ma, "Formal logic enabled personalized federated learning through property inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 10882–10890.
- [4] R. Tripathy, "A hybrid federated learning for medical cyber-physical systems," in *Proceedings of the 25th International Conference on Distributed Computing and Networking*, 2024, pp. 377–381.
- [5] S. Chen, G. Long, J. Jiang, and C. Zhang, "Federated foundation models on heterogeneous time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 15, 2025, pp. 15839–15847.
- [6] W. Marfo, D. K. Tosh, and S. V. Moore, "Federated learning for efficient condition monitoring and anomaly detection in industrial cyber-physical systems," in *2025 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2025, pp. 740–746.
- [7] M. K. Quan, P. N. Pathirana, M. Wijayasundara, S. Setunge, D. C. Nguyen, C. G. Brinton, D. J. Love, and H. V. Poor, "Federated learning for cyber-physical systems: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2025.
- [8] H. T. Truong, B. P. Ta, Q. A. Le, D. M. Nguyen, C. T. Le, H. X. Nguyen, H. T. Do, H. T. Nguyen, and K. P. Tran, "Light-weight federated learning-based anomaly detection for time-series data in industrial control systems," *Computers in Industry*, vol. 140, p. 103692, 2022.
- [9] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: challenges and applications," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.
- [10] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [11] T. D. Nguyen, T. Nguyen, P. Le Nguyen, H. H. Pham, K. D. Doan, and K.-S. Wong, "Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions," *Engineering Applications of Artificial Intelligence*, 2024.
- [12] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948.
- [13] H. Figueroa, Y. Wang, and G. C. Giakos, "Adversarial attacks in industrial control cyber-physical systems," in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2022, pp. 1–6.
- [14] A. D. M. Ibrahim, M. Hussain, and J.-E. Hong, "Deep learning adversarial attacks and defenses in autonomous vehicles: A systematic literature review from a safety perspective," *Artificial Intelligence Review*, vol. 58, no. 1, p. 28, 2024.
- [15] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021.
- [16] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Information Fusion*, vol. 90, pp. 148–173, 2023.
- [17] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.
- [18] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.
- [19] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.
- [20] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, "Defending against backdoors in federated learning with robust learning rate," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.
- [21] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.
- [22] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.
- [23] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," *ArXiv*, vol. abs/2201.00763, 2022.
- [24] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.
- [25] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5650–5659.
- [26] M. Ma, J. Gao, L. Feng, and J. Stankovic, "StlNet: Signal temporal logic enforced multivariate recurrent neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14604–14614, 2020.
- [27] T. Nguyen, D. Kapur, W. Weimer, and S. Forrest, "Dig: A dynamic invariant generator for polynomial and array invariants," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 4, pp. 1–30, 2014.
- [28] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19586–19597, 2020.
- [29] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2749–2758.
- [30] A. K. Nair, E. D. Raj, and J. Sahoo, "A robust analysis of adversarial attacks on federated learning environments," *Computer Standards & Interfaces*, vol. 86, p. 103723, 2023.
- [31] K. N. Kumar, C. K. Mohan, and L. R. Cenkeramaddi, "The impact of adversarial attacks on federated learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- [32] G. Xia, J. Chen, C. Yu, and J. Ma, "Poisoning attacks in federated learning: A survey," *IEEE Access*, vol. 11, pp. 10708–10722, 2023.
- [33] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 19–35.
- [34] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in neural information processing systems*, vol. 31, 2018.
- [35] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [36] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net, 2020.
- [37] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.
- [38] P. Rathore, A. Basak, S. H. Nistala, and V. Runkana, "Untargeted, targeted and universal adversarial attacks and defenses on time series," in *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–8.
- [39] N. Müller, D. Kowatsch, and K. Böttinger, "Data poisoning attacks on regression learning and corresponding defenses," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2020, pp. 80–89.
- [40] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of

- Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 634–643.
- [41] T. D. Nguyen, T. A. Nguyen, A. Tran, K. D. Doan, and K.-S. Wong, “Iba: Towards irreversible backdoor attacks in federated learning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
 - [42] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, “Can you really backdoor federated learning?” *ArXiv*, vol. abs/1911.07963, 2019.
 - [43] S. P. Karimireddy, L. He, and M. Jaggi, “Byzantine-robust learning on heterogeneous datasets via bucketing,” in *International Conference on Learning Representations*, 2022. [Online]. Available: <https://openreview.net/forum?id=jXKKDEi5vJt>
 - [44] Y. Allouah, S. Farhadkhani, R. Guerraoui, N. Gupta, R. Pinot, and J. Stephan, “Fixing by mixing: A recipe for optimal byzantine ML under heterogeneity,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 1232–1300.
 - [45] S. P. Karimireddy, L. He, and M. Jaggi, “Learning from history for byzantine robust optimization,” in *International Conference on Machine Learning*. PMLR, 2021.
 - [46] S. Huang, Y. Li, C. Chen, L. Shi, and Y. Gao, “Multi-metrics adaptively identifies backdoors in federated learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4652–4662.
 - [47] X. Mu, K. Cheng, Y. Shen, X. Li, Z. Chang, T. Zhang, and X. Ma, “Feddmc: Efficient and robust federated learning via detecting malicious clients,” *IEEE Transactions on Dependable and Secure Computing*, 2024.
 - [48] A. Raza, S. Li, K.-P. Tran, and L. Koehl, “Using anomaly detection to detect poisoning attacks in federated learning applications,” *arXiv preprint arXiv:2207.08486*, 2022.
 - [49] T. D. Nguyen, A. D. Nguyen, T.-H. Nguyen, K.-S. Wong, H. H. Pham, T. T. Nguyen, and P. Le Nguyen, “Fedgrad: Mitigating backdoor attacks in federated learning through local ultimate gradients inspection,” in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 01–10.
 - [50] X. Cao, M. Fang, J. Liu, and N. Z. Gong, “Fltrust: Byzantine-robust federated learning via trust bootstrapping,” *arXiv preprint arXiv:2012.13995*, 2020.
 - [51] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.
 - [52] E. Bartocci, C. Mateis, E. Nesterini, and D. Nickovic, “Survey on mining signal temporal logic specifications,” *Information and Computation*, p. 104957, 2022.
 - [53] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, “Telex: Passive stl learning using only positive examples,” in *International Conference on Runtime Verification*. Springer, 2017, pp. 208–224.
 - [54] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaning attack on neural networks,” in *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.
 - [55] C. Zhu, J. Zhang, X. Sun, B. Chen, and W. Meng, “Adfl: Defending backdoor attacks in federated learning via adversarial distillation,” *Computers & Security*, vol. 132, p. 103366, 2023.
 - [56] S. Sarfraz, V. Sharma, and R. Stiefelhofen, “Efficient parameter-free clustering using first neighbor relations,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8934–8943.
 - [57] F. Sattler, K.-R. Müller, and W. Samek, “Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints,” *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.
 - [58] FHWA, “Highway performance monitoring system field manual,” 2016 [Online]., office of Highway Policy Information. [Online]. Available: https://www.fhwa.dot.gov/policyinformation/hpms/fieldmanual/hpms_field_manual_dec2016.pdf
 - [59] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.
 - [60] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “Scaffold: Stochastic controlled averaging for federated learning,” in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.
 - [61] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, “Federated learning based on dynamic regularization,” *arXiv preprint arXiv:2111.04263*, 2021.
 - [62] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.