Lightweight Techniques for Federated Anomaly Detection in Log Data

Anastasia Pustozerova, André García Gómez, Max Landauer, Markus Wurzenberger, Florian Skopik, Edgar Weippl, Rudolf Mayer, and Andreas Ekelhart

***Abstract*—Accurately and efficiently identifying anomalies within log data is crucial for maintaining the reliability, availability, and security of modern computing systems. In interconnected environments, log data often come from distributed sources such as Internet of Things devices, industrial networks, or smart grids. Centralizing these logs for anomaly detection can be challenging due to strict confidentiality requirements, regulatory constraints, and the limited bandwidth of edge networks. Federated learning (FL) offers an alternative by enabling local model training on site, while aggregating only models instead of sensitive data, thereby preserving data confidentiality and reducing data transfer. This article develops and evaluates a federated log-anomaly detection pipeline and analyzes its components. We adapt lightweight anomaly detection techniques in a federated setting, comparing them with deep learning (DL) methods, assessing detection capabilities, computational efficiency, memory requirements, and inference time. We explore the residual privacy risks in FL within the proposed pipeline, develop a threat model and suggest mitigation strategies. Our findings indicate that lightweight anomaly detection methods can match the effectiveness of DL techniques in the FL framework, while often providing improved computational and communication efficiency. Furthermore, these techniques show better resilience to non-IID data distributions, a typical FL challenge that can severely hinder the effectiveness of traditional machine learning models. However, the memory footprint of lightweight models varies with dataset characteristics and, in some cases, may exceed that of DL models.**

***Index Terms*—Anomaly detection, federated learning (FL), log data.**

I. Introduction

DETECTING anomalies in log data is vital for maintaining the secure and reliable operation of computing systems. In various settings, including Internet of Things (IoT) environments and distributed systems, logs are generated in a decentralized manner, across numerous interconnected devices or edge nodes. Monitoring system logs and applying anomaly detection techniques is essential for identifying security threats, operational errors, and maintenance requirements [23], [24], [37], [45], [54]. However, data protection and confidentiality concerns, as well as regulatory requirements, often prevent centralized aggregation, data analysis, and training of anomaly detection models across devices or organizations. Federated learning (FL) allows training such models without centralizing sensitive data; instead, models are trained locally on the device, and only the models are shared with the server. FL has gained significant interest in the area of anomaly detection on edge devices, offering not only enhanced privacy, but also improved system efficiency and reduced latency [58].

Previous studies on anomaly detection using FL have largely concentrated on machine learning ML-based approaches, leading to significant progress in log-based anomaly detection [29]. Nonetheless, these approaches often impose substantial training overheads and exhibit inefficiencies during inference, which can degrade responsiveness and undermine reliability in time-critical applications. This issue is exacerbated on edge devices within IoT environments where computational resources are constrained.

In this research, we introduce a federated log anomaly detection framework aimed at enabling rapid and lightweight detection methods. We explore and adapt to federated settings widely used deep learning methods along with simpler, lightweight anomaly detection techniques, such as those proposed in [30]. We refer to these methods as *lightweight* because they are designed to minimize computational and communication overhead while efficiently processing large volumes of log data, even on hardware with limited resources. We provide a comprehensive experimental evaluation of these techniques in FL. This evaluation considers settings with varying numbers of clients in FL, covering both IID (independent and identically distributed) and non-IID data distributions. Imbalanced data distributions across clients pose a major challenge in FL [27], often leading to prolonged convergence and, in some cases, suboptimal model performance. We assess anomaly detection effectiveness by measuring the F1-score, precision, and recall, and evaluate efficiency in terms of data transfer volume, inference latency, and training.

While FL avoids direct data sharing, it still requires the exchange of model updates or, in the case of lightweight

Received 2 October 2025; revised 21 December 2025; accepted 21 February 2026. Date of publication 1 April 2026; date of current version 24 April 2026. This work was supported in part by European Union under European Defence Fund under Grant 101121403 - NEWSROOM, in part by Austrian FFG Kiras project ASOC under Grant FO999905301, in part by SBA Research under Grant SBA-K1 NGC, in part by COMET Center funded by BMIMI, BMWET, and in part by the Federal State of Vienna, managed by FFG. Associate Editor: H. Jiang. *(Anastasia Pustozerova and André García Gómez contributed equally to this work and share the first authorship.) (Corresponding author: Anastasia Pustozerova.)*

Anastasia Pustozerova, Edgar Weippl, Rudolf Mayer, and Andreas Ekelhart are with the University of Vienna, 1090 Vienna, Austria (e-mail: anastasia.pustozerova@univie.ac.at; edgar.weippl@univie.ac.at; rudolf.mayer@ait.ac.at; andreas.ekelhart@univie.ac.at).

André García Gómez, Max Landauer, Markus Wurzenberger, and Florian Skopik are with the AIT Austrian Institute of Technology, 1210 Vienna, Austria (e-mail: andré.garcía-góme@ait.ac.at; max.landauer@ait.ac.at; markus.wurzenberger@ait.ac.at; Florian.skopik@ait.ac.at).

Digital Object Identifier 10.1109/TR.2026.3677674

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

techniques, statistical representations, features or parts of the original data, which may still contain confidential information. Deep learning models trained with FL are also vulnerable to disclosure risks, as sensitive data can be extracted through model access using various inference attacks [13], [53]. Since preserving privacy is a core motivation for employing FL, we examine the residual confidentiality risks within the proposed lightweight federated anomaly detection framework and suggest mitigation strategies.

This work addresses the following research questions.

1) *RQ1: How do lightweight anomaly detection models trained under IID and non-IID data distributions in a federated setting compare with centralized and local models in terms of detection effectiveness?* The objective of this question is to assess the advantages of FL by contrasting the performance of the global model against: a) models trained locally by individual clients, to highlight the value of collaboration; and b) a centralized model, to demonstrate the extent to which FL can approximate the performance of a centralized baseline while preserving data privacy.
2) *RQ2: How do lightweight anomaly detection methods compare with deep learning (DL)-based approaches in FL in terms of both effectiveness and efficiency, and what are the implications for system reliability?* The objective is to evaluate whether lightweight techniques can achieve comparable detection effectiveness to DL models while offering advantages in computational and communication efficiency—factors that directly influence the responsiveness, availability, and overall reliability of resource-constrained distributed systems.
3) *RQ3: What disclosure risks arise when performing anomaly detection in federated learning, and how can these risks be mitigated?* The aim is to identify confidentiality threats in federated anomaly detection, develop a threat model, and propose mitigation strategies, thereby ensuring that privacy breaches do not undermine the trustworthiness of the system.

Our main contributions are as follows.

1) *Federated log anomaly detection pipeline:* We design and formalize a unified FL pipeline that integrates lightweight anomaly detection techniques alongside deep learning models, and we release all code publicly to support reproducibility.
2) *Systematic evaluation of lightweight methods in FL:* We provide the first comprehensive assessment of lightweight log anomaly detection methods across diverse FL scenarios–varying client counts and both IID and non-IID data distributions–evaluating detection performance, inference cost, and communication overhead.
3) *Privacy threat model and risk assessment:* We develop a dedicated threat model for federated log anomaly detection and analyze confidentiality risks across all evaluated methods, proposing practical mitigation strategies to enhance dependability and trustworthiness in real-world deployments.

The rest of this article is organized as follows. In Section II, we review recent advances in centralized and federated log anomaly detection. Section III introduces the proposed federated anomaly detection pipeline, elaborating on various design choices, potential vulnerabilities, and alternative design considerations. Section IV covers the adaptation of lightweight techniques in FL and describes the experimental setup. Section VI presents and analyzes the findings in response to RQ1 and RQ2. Section VII addresses RQ3 by analyzing confidentiality risks, presenting a threat model, and mitigation strategies. Section VIII considers potential risks that may influence the validity of our experiments. Finally, Section IX concludes this article and suggests directions for future research.

II. Related Work

The related work is organized into two parts: First, it covers log anomaly detection techniques, emerging trends, and open challenges; second, it explores the application of FL in log-based tasks and other security scenarios, along with a discussion on residual privacy risks in FL.

A. Log Anomaly Detection

In the realm of log data anomaly detection, deep learning methodologies, particularly DeepLog [10], have been of significant influence [29]. The research of Le and Zhang [33] provides an assessment of various models, focusing on both semisupervised methods (such as DeepLog, LogAnomaly [44], PLELog [63]) and supervised methods (including CNN [39], LogRobust [67]). Supervised techniques necessitate labeled data distinguishing anomalies from normalcy, whereas semisupervised approaches utilize only data from normal operations for training. Despite the robust performance of DL techniques [33], their effectiveness frequently declines in real-world scenarios in terms of accuracy and processing speed. Typically, supervised models are regarded as the more effective choice [33], [38], however, this assumption can be misleading, as it presumes that all anomalies are labeled and drawn from the same distribution - a condition rarely satisfied in real-world settings. Anomaly detection diverges from binary classification because it aims to detect outliers within the predominant distribution, which renders supervised methods less ideal even though they are extensively favored in academic discussions [64].

Recent work has explored the use of large language models (LLMs) for log anomaly detection [16]. Prominent examples include LogBERT [17] and NeuralLog [32], which implement BERT-type strategies, alongside LogGPT [19], a model based on a Generative Pretrained Transformer for predictive tasks. Techniques like PreLog [31] and BigLog [55] are LLMs trained with log data, while methods such as SuperLog [25] and LogPrompt [38] leverage prompt-based strategies. These innovations, though promising, often increase computational and memory demands, and are insufficient in addressing challenges like detection speed and privacy. There is a demand among practitioners for faster solutions and concerns about the inconsistency in the availability of the anomaly label [40]. Employing LLMs for rapid inference involves significant computational power, typically scarce in environments such as IoT, and reliance on third-party APIs can increase privacy and cost-related issues.

Recent developments in deep learning for log anomaly detection have trended toward increasingly complex architectures. However, multiple studies indicate that simpler approaches can outperform these sophisticated models. For example, Yu et al. [65] compared supervised DL methods such as LogRobust and NeuralLog with classical machine learning techniques (e.g., Decision Trees, K-nearest neighbor) and found that the latter were more effective; on the Thunderbird dataset, a simple K-nearest neighbor baseline trains roughly $1,000\times$ faster than NeuralLog while achieving an F1-score higher by 0.0625. Likewise, Landauer et al. [30] show that lightweight statistical techniques often match or exceed the performance of deep models such as DeepLog and LogAnomaly, outperforming them by 1–9 percentage points in F1-score on standard benchmarks (HDFS, BGL datasets) while requiring orders of magnitude less computation.

These reviews, however, have methodological limitations, lacking empirical evidence on computational efficiency, and not considering factors like memory usage, and provide insufficient transparency in parameter optimization, potentially leading to biased outcomes.

B. Federated Anomaly Detection

FL is widely employed for anomaly detection in distributed data sources such as industrial systems [37], [41], [57], IoT networks [8], and smart grids [26]. FL bolsters data privacy since the data stays on its original device, while only the trained models are sent to a central server for integration into a global model. This technique not only secures privacy but also minimizes bandwidth consumption by transmitting streamlined models in place of entire datasets [26], [45]. FL is proven to conserve bandwidth [23], improve performance [57], and optimize system resource usage [26]. Additionally, it accelerates training processes by distributing computations among clients [45], [50]. Further improvements include enhanced communication efficiency via gradient compression [37] and refined system architecture for quicker attack responses in industrial IoT environments [56].

While numerous studies have explored the application of FL to anomaly detection, comparatively little work has focused specifically on log-based anomaly detection in distributed systems. Guo et al. [18] propose a streamlined FL method for large-scale software ecosystems, whereas Li et al. [35] introduce a CNN–LSTM framework with gradient compression to improve communication efficiency while preserving privacy. Shin et al. [52] evaluate multiple models for federated log anomaly detection, demonstrating that performance comparable to centralized approaches can be achieved, and Himler et al. [21] reported similar findings by adapting the DeepLog model to an FL setting. A key challenge in FL, however, is the prevalence of non-IID data across clients, which can hinder model accuracy and increase communication overhead [27], [68]. For example, Himler et al. [21] examined how non-IID distributions affect the efficiency of federated log anomaly detection, while Li et al. [35] leveraged the lottery ticket hypothesis to better tailor global models to heterogeneous datasets, improving personalization and reducing communication costs. Overall, existing studies differ substantially in terms of client population, data heterogeneity, and task formulations, complicating direct comparison and leaving open questions about the robustness and reliability of their findings across diverse deployment scenarios [18], [21], [35], [52]. In this work, we address these gaps by systematically evaluating lightweight and deep learning-based approaches within a federated framework, with a particular emphasis on effectiveness and efficiency under realistic data and system conditions.

Privacy preservation is another critical challenge in FL, as locally trained models aggregated into a global model remain vulnerable to attacks such as membership inference [53] and model inversion [13]. To mitigate these risks, several privacy-enhancing techniques have been proposed, including Homomorphic Encryption (HE) [48], Differential Privacy (DP) [43], and Secure Multi-party Computation (SMPC) [6], with DP in particular recognized for providing strong theoretical guarantees. In the context of log anomaly detection, Guo et al. [18] proposed an FL-based anomaly detection framework augmented with HE. To address bandwidth constraints—particularly relevant given HE's sensitivity to model size—they introduce lightweight models that improve communication efficiency in this setting. In this work, we advance prior efforts by investigating the residual privacy risks of federated learning with lightweight anomaly detection methods and proposing mitigation strategies designed to enhance both confidentiality and system reliability (see Section VII).

III. Federated Log Anomaly Detection Pipeline

This section outlines a generic use case for the proposed pipeline and describes its key components, with detailed analysis provided in subsequent sections.

A. Use Case

Our proposed use case describes a federated learning architecture with multiple client servers and a central server (Fig. 1). Clients keep their own system logs and run local anomaly detection, but collaborate via federated learning because individual datasets are too small to train effective anomaly detection models. Raw logs never leave the clients; only model updates are sent to the central server, which aggregates them into a global model and redistributes them to clients for further training, evaluation, or direct use in anomaly detection, improving performance, and preserving confidentiality.

According to [40], anomaly detection systems must work quickly and with minimal labeled data, as such data are often scarce. For this reason, we focus on smaller deep learning models such as DeepLog and LogAnomaly rather than larger, transformer-based architectures. In addition, we include simpler and more computationally efficient techniques that avoid heavy optimization; we refer to these methods as **lightweight**. It is important to note that even when powerful computing resources are available, they are rarely dedicated to monitoring tasks, and many real-world environments, such as IoT networks, industrial systems, and edge deployments, as exemplified in [61], are inherently resource constrained. These considerations motivate

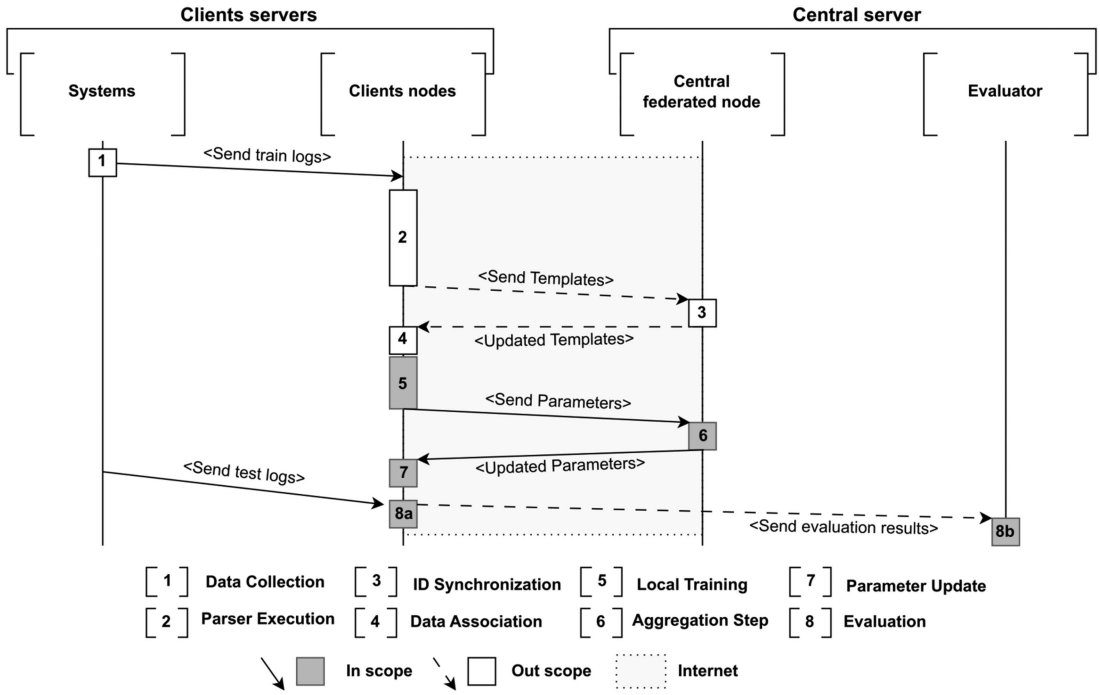


Fig. 1. Federated log anomaly detection training pipeline.

our emphasis on lightweight approaches that balance detection performance with computational efficiency, a principle that guides the design of the present work.

B. Pipeline

The federated log anomaly detector training pipeline presented in Fig. 1 can be broken down into the following steps.

1) *Data Collection:* Each client $c \in C$ continuously collects system log data L_c.
2) *Parser Execution:* A log parsing function $P(\cdot)$ is executed locally to transform unstructured logs into structured templates (see Fig. 2)

$$T_c = P(L_c)$$

where T_c denotes the set of log templates extracted from client c's logs. Parsers such as Drain [20] automatically abstract variable tokens (timestamps, identifiers, values), yielding template representations [4].
3) *ID (Template) Synchronization:* Each client transmits its local template set T_c to the server, which constructs a global dictionary

$$\mathcal{D} = \{(t_i, e_i)\}$$

mapping each unique template t_i to a global event ID e_i. Although templates abstract sensitive values, the presence of certain template types may still reveal information; appropriate mitigations may be required (see Section VII).
4) *Data Association:* Clients receive a global template dictionary $\mathcal{D}$ from the server and map each parsed log message to the corresponding global event ID. Each client thus obtains a dataset of sequences

$$S_c = \{s_1, s_2, \ldots\}$$

where $s_j = (e_1, e_2, \ldots)$ represents a sequence of events. The overall process can be seen in Fig. 2.
5) *Local Training:* Each client trains a local anomaly detection model on its event sequences S_c.
6) *Aggregation Step:* Clients send their trained models to the central server. Raw logs L_c always remain local.
7) *Parameter Update:* The server aggregates the local models using an appropriate aggregation technique (see

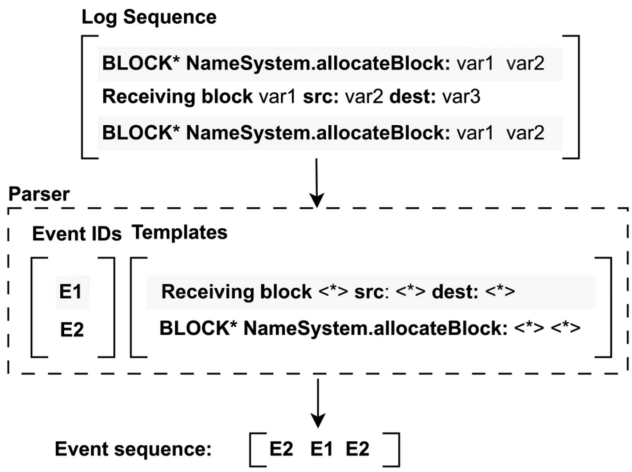


Fig. 2. Log sequence conversion to event sequence with the parser.

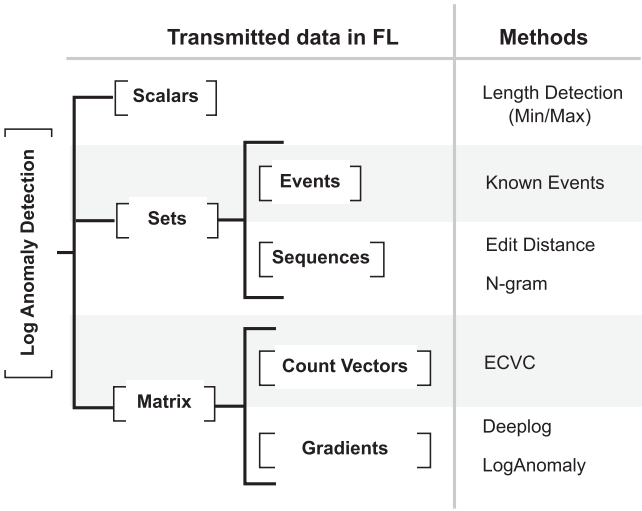


Fig. 3. Anomaly detection methods considered in this work, categorized by the type of data transferred during federated learning.

Section IV) and redistributes the updated global model to the clients.

8) *Evaluation:* Model performance is evaluated either locally using client-specific test sets, or centrally if a global test set is available.

As this study focuses on establishing strong baselines, we limit our analysis to the core federated training part (steps 5–8), and leave federated parsing (steps 2–4) outside the scope of this work. Accordingly, we simulate a federated setting by assuming that clients already possess log data parsed into global templates and prepared for training.

IV. Federated Lightweight Log Anomaly Detection Methods

In this study, we evaluate multiple log anomaly detection methods—both deep learning models and lightweight statistical techniques within a federated learning setting. Building on evidence that lightweight methods can outperform deep learning approaches [30], [65], we adapt five such techniques to the federated setting by designing custom aggregation strategies. Using a unified preprocessing pipeline (log parsing into templates, mapping messages to template sequences, and learning sequence patterns), we compare these lightweight methods against standard deep learning baselines with respect to detection accuracy and computational efficiency. Fig. 3 provides an overview of the evaluated methods and the types of data exchanged during FL.

A. Lightweight Anomaly Detection Definition

Building on the assumption that these methods must operate on IoT networks or other resource-constrained edge devices (see Section III-A) and must provide high detection speed [40], we define **lightweight log anomaly detection methods** as techniques that, *in a semisupervised setting, detect abnormal logs while maintaining low computational complexity during both training and inference*. We exclude supervised methods because they introduce several challenges in anomaly detection, including the unrealistic assumption of balanced normal and abnormal data distributions [7] and the reliance on optimization procedures that offer no guarantees for identifying previously unseen anomalies [60].

B. Lightweight Anomaly Detection Techniques in Federated Learning

We adapt the lightweight techniques introduced in [30] to a federated setting. Each method employs a different strategy for the aggregation phase, which we implemented using the Flower framework [5].

*1) Length Detection **(Length)**. Centralized version:* We determine the shortest and longest sequence lengths in the training set by counting how many log entries appear in each sequence. During inference, any sequence that falls outside this range is deemed abnormal.

Federated version: Each client computes the minimum and maximum sequence lengths in its local data and transmits these values to the server. The server then determines the global length range by taking the minimum of all local minima and the maximum of all local maxima across clients.

*2) Known Events **(Events)**. Centralized version:* A set is formed that comprises all unique events present in the training dataset. During inference, if an event not included in this set appears in an event sequence, the entire sequence is classified as abnormal.

Federated version: Each client generates a set of unique events locally. During the aggregation phase, these sets are combined through a set union operation, forming a global anomaly detector.

*3) Edit Distance **(Edit)**. Centralized version:* We compile a set S that contains all distinct sequences of the training dataset. During inference, the score for a new sequence $\hat{s}$ is calculated as

$$score = \min(LV(\hat{s}, s) \forall s \in S) \tag{1}$$

where $LV(\cdot, \cdot)$ is the Levenshtein distance, i.e., the minimum number of edits (insertions, deletions, or substitutions) required to change one sequence into the other [34]. A sequence is considered abnormal if the score exceeds a given threshold.

Federated version: Analogously to computing *New Event Types* (c.f. Section IV-B-2), each client compiles a set with distinct sequences based on their local data, and the central server aggregates local sets using the union operation. Computing the score is then the same as for the centralized version.

4) N-Gram. Centralized version: The method involves preprocessing all sequences by segmenting them into subsequences of length n. We then collect all the distinct subsequences into a set S. During inference, a subsequence not found in the set is deemed anomalous [12].

Federated version: The server computes the union of all the locally pre-processed sequences from the clients.

*5) Event Count Vector Clustering **(ECVC)**:* Every sequence is preprocessed and converted into a count vector $y = CV(\cdot, n)$, with n denoting the number of IDs of events in the sequence, which will be the length of the resulting vector y. This new vector captures the frequency of event occurrences within the

sequence. Normalization is applied by dividing each element by the sum of all elements, which results in $\hat{y} = norm(CV(\cdot, n))$. To illustrate the behavior of CV, consider the following example that counts the occurrences of the events [0..6] in the given sequence $[1, 3, 4, 3, 1]$

$$CV([1, 3, 4, 3, 1], 7) = [0, 2, 0, 2, 1, 0, 0].$$

Centralized version: All distinct normalized count vectors from the training data are combined to form a matrix $W \in \mathbb{R}^{m \times n}$, where m denotes the total number of unique vectors found in the training set. For inference, the score for a new normalized count vector $\hat{y}$ is calculated as specified in (2). A sequence is considered abnormal if this score exceeds a set threshold

$$score = \min\left(\left\{\frac{\sum_{i=0}^{n-1}|\hat{y}_i - W_{i,j}|}{\sum_{i=0}^{n-1}\max(\hat{y}_i, W_{i,j})} | j \in \mathbb{N}, j < m\right\}\right). \tag{2}$$

Observe that the vector $\hat{y}$ might include events that were not present during the aggregation phase. In such scenarios, it is necessary to expand W by adding zeros.

Federated version: We combine all local W matrices and eliminate duplicate columns to construct a global W. Keep in mind that if the local W matrices vary in column size, we need to standardize their dimensions by appending zeros.

6) Combining Lightweight Techniques: In addition to the five separately evaluated lightweight techniques, we evaluate four combinations of such techniques to improve anomaly detection effectiveness. These combinations were also considered in [30] and include: a) **2-Gram + Length**; b) **Events + Length**; c) **Events + Length + ECVC**; d) **Events + Length + Edit**. In each of these combination cases, we train the methods separately locally and then independently aggregate these local models using the corresponding aggregation methods. Thus, the final global model is represented by an ensemble of 2–3 global models (e.g., 2-Gram and Length). A sequence is classified as an anomaly if at least one of the models in the ensemble predicts an anomaly.

C. Deep Learning Models for Log Anomaly Detection in Federated Learning

To evaluate lightweight log anomaly detection techniques, we compare them against two widely used LSTM-based deep learning baselines: DeepLog [10] and LogAnomaly [44]. We specifically select LSTM-based baselines because they are smaller and computationally less demanding than transformer-based architectures such as LogBERT [17] and therefore represent comparatively lightweight deep learning techniques. Moreover, these models are among the few deep learning approaches commonly applied in semi-supervised log analysis and have demonstrated strong performance [33]. Most other log anomaly detection models in the literature rely on labeled data as they are trained in a fully supervised manner [33], [18], making them unsuitable for our setting. Like the lightweight methods, both baselines operate in a semisupervised manner: they are trained exclusively on normal log sequences to learn typical event patterns. Given a window of past log events, the model predicts a probability distribution over the next possible event; if the true next event does not appear among the top g predicted candidates, the sequence is flagged as anomalous.

In the federated learning setup, training begins with a randomly initialized global model on the central server. This global model is distributed to all clients for training on their local data. Clients then return their updated parameters, which the server aggregates using FedAvg [42]. This server–client update cycle continues until the global model converges.

TABLE I
EXPERIMENTAL SETUP COMPARISON

Work	Task	HDFS	BGL	IID	No-IID	N. clients
Guo [18]	SUP	✓	*No*	✓	*No*	50
Li [35]	SSL	✓	✓	✓	✓	10
Shin [52]	SUP	✓	✓	✓	*No*	5
Himler [21]	SSL	✓	*No*	✓	✓	5
Ours	SSL	✓	✓	✓	✓	5, 10, 50

(SUP—Supervised, SSL—Semisupervised Learning).

V. EXPERIMENTAL SETUP

In this section, we outline the general setup of our experiments, which were conducted using the Flower framework.[1] To ensure reproducibility, we provide the full implementation of our methods and experiments.[2]

A. Dataset

To enable comparison with earlier research [16], [18], [30], [35], we use two benchmark datasets frequently used in log anomaly detection. These datasets were preprocessed using the templates outlined in [30]. As is the previous log anomaly detection works [10], [17], [28], we randomly sample 1% of the normal sequences for training an anomaly detection model; the remainder of the dataset is used for evaluation.

The Hadoop Distributed File System **(HDFS)** *dataset* [62] contains 33 log event types, 558 223 normal and 16 838 abnormal event sequences.

The BlueGene/L **(BGL)** *dataset* [46] consists of system logs collected during the operations of the BlueGene/L supercomputer. The dataset consists of 37 823 normal and 31 429 abnormal sequences and has 394 event types.

B. Comparison With Previous Work

As shown in Table I, prior studies exhibit inconsistencies in their experimental setups. While there is general agreement on using the HDFS and BGL datasets, the number of clients considered in federated learning varies across works: [21] used five clients due to resource constraints, [35] consider ten clients, and [18] evaluate scenarios with fifty clients. To enable a broader and more robust comparison, we evaluate settings with 5, 10, and 50 clients, covering both small- and large-scale deployments. This design choice allows us to analyze how detection effectiveness and system efficiency scale under different operational conditions.

[1] https://flower.ai/
[2] https://github.com/ait-aecid/comparison-fed-centr-efficient-ad

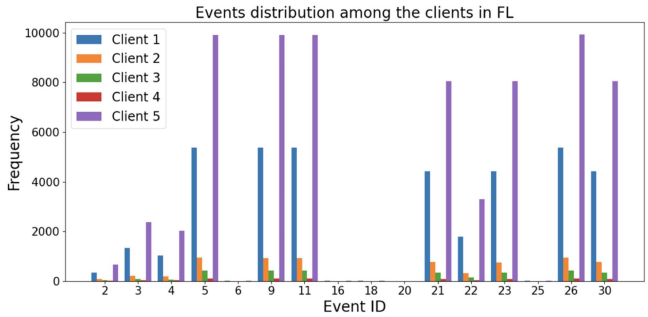


Fig. 4. Per-client distribution of log event frequencies in a non-IID FL setting with five clients on the HDFS dataset.

Non-IID split: Only a limited number of prior studies have investigated non-IID data, despite its critical impact on federated learning, where heterogeneous client distributions can significantly degrade model performance: Himler et al. [21] assign clients fixed dataset percentages; Li et al. [35] focused on feature distribution skew. In our work, we adopt a log-normal distribution ($\mu = 0, \sigma = 0.25$) [66] to allocate varying amounts of training data across clients, thereby generating diverse imbalanced distributions and enabling a fair evaluation of lightweight techniques under realistic non-IID conditions. To enhance the robustness and reliability of our findings, each experiment is repeated three times with different random seeds, and we report both mean and standard deviation of the results. Fig. 4 shows an example of a non-IID data split across five clients, highlighting the heterogeneity in log event distributions at each client.

C. Model Training and Parameter Optimization

We perform the experiments utilizing the methods described in Section IV. DeepLog and LogAnomaly are implemented based on the open source library LogDeep.[3] For DL models, we perform hyperparameter tuning on the validation set (10% of the training data) using grid search. In our setting, no abnormal sequences are assumed to be available for validation or hyperparameter tuning; therefore, model selection is performed solely based on the validation loss. We try the following parameters for both BGL and HDFS datasets with LogAnomaly and DeepLog models: Window size [2,3,4,5,10,15], number of candidates for HDFS [5,9,10,11,15] for BGL [10,40,60,80], learning rate $[10^{-3}, 10^{-2}$, max epoch [1,5,10,30,50,100]. For the HDFS dataset, we find the optimal window size of 10 to transform the input data and select the top 9 candidates from the output. For the BGL dataset, we use a window size of 2 and the top 60 candidates. To train the DeepLog and LogAnomaly models, we use an architecture with two hidden layers and 64 neurons per layer. While DeepLog takes only the sequence of log events as input, LogAnomaly additionally takes the frequency of each event occurring in the sequence. For both models, we utilize the Adam optimiser with a learning rate of 10^{-3}, a Cross-Entropy Loss Function, a batch size of 2048, and 10 epochs for the HDFS dataset and 30 epochs for the BGL dataset.

[3]https://github.com/d0ng1ee/logdeep/tree/master?tab=readme-ov-file

For the lightweight methods *ECVC, N-Gram,* and *Edit*, we determine optimal thresholds by following a similar strategy as for hyperparameter tuning for DL.

D. Hardware

We ran our experiments on a computer equipped with an Intel Core Ultra 7 165Ux14 CPU and 16 GiB of RAM to assess the lightweight methods, while an NVIDIA GeForce RTX 3090 GPU and 1.0TiB of RAM was used for the DL models. This division ensured a fair comparison between the different approaches, as DL models perform better with a GPU, whereas the other methods benefit from a fast CPU.

E. Evaluation

For the evaluation, we consider the following metrics.

1) *Effectiveness:* We use F1 score, Precision, and Recall to assess how well the methods detect anomalies.
2) *Inference time:* We measure the inference speed (in seconds) required to process the entire test set.
3) *Local training time:* The duration (in seconds) required for training the local models is measured. This includes the time spent on training through the necessary number of FL iterations.
4) *FL iterations:* The total count of communication (aggregation) rounds needed throughout the training phase.
5) *Memory usage:* The total size, in bytes, of the lightweight methods and models after aggregation.

VI. Results and Discussion

This section presents the core findings of our study, where we rigorously compare lightweight and deep learning anomaly detection methods to evaluate their effectiveness and efficiency in federated environments.

A. Effectiveness

Table II presents the F1 scores for both the DL and lightweight methods (An equivalent table for recall and precision is presented in Table IX). **A key strength of the lightweight techniques is their ability to achieve identical performance in centralized, IID, and non-IID FL settings, as their global models are constructed via deterministic set-union or min–max operations that are independent of client data distributions.** DL methods also demonstrate robustness to varying numbers of clients and experience only a slight drop in effectiveness when dealing with non-IID data with the HDFS dataset, or when increasing to 50 clients with the BGL dataset (see Table II). The FedAvg algorithm aggregates DL models in FL by averaging local weights, making it inherently more susceptible to shifts in data distribution across participating clients. It is important to note, however, that DL techniques require more FL iterations to achieve high effectiveness, especially with a larger number of nodes in the setting (see Section VI-B).

A paired t-test was used to assess whether lightweight techniques differ significantly from the DL baselines. On the HDFS

TABLE II
F1-SCORE OF **GLOBAL** MODELS TRAINED IN FEDERATED LEARNING

	HDFS		BGL	
	IID	non-IID	IID	non-IID
LogAnomaly (centralized)	0.93±0.00		0.97±0.00	
DeepLog (centralized)	0.93±0.00		0.97±0.00	
LogAnomaly (5 clients)	0.92±0.01	0.93±0.00	0.97±0.00	0.97±0.00
DeepLog (5 clients)	0.93±0.00	0.91±0.00	0.97±0.00	0.97±0.00
LogAnomaly (10 clients)	0.93±0.00	0.92±0.00	0.97±0.00	0.97±0.00
DeepLog (10 clients)	0.93±0.00	0.92±0.00	0.97±0.00	0.97±0.00
LogAnomaly (50 clients)	0.93±0.00	0.93±0.00	0.96±0.00	0.97±0.00
DeepLog (50 clients)	0.93±0.00	0.93±0.00	0.96±0.01	0.96±0.00
2-Gram	0.85±0.10		0.63±0.00	
3-Gram	0.85±0.07		0.63±0.00	
ECVC	**0.95±0.01**		0.93±0.01	
Edit	0.72±0.00		0.93±0.00	
Events	0.62±0.15		**0.99±0.00**	
Length	0.57±0.03		0.03±0.03	
2-Gram + Length	0.87±0.10		0.63±0.00	
Events + Length	0.78±0.12		**0.99±0.00**	
Events + Length + ECVC	**0.95±0.01**		0.95±0.01	
Events + Length + Edit	0.87±0.09		0.95±0.00	

Bold values indicates the best-performing models within each setting.

dataset, *ECVC* and *Events + Length + ECVC* significantly outperformed both DL models ($p < 0.05$ across all settings), achieving F1-scores around 0.95 compared to 0.91–0.93 for DL (see Table II). In contrast, *Length* and *EditDistance* were significantly worse ($p < 0.01$), with mean F1-scores of 0.57 and 0.72. All remaining lightweight methods showed no significant differences from DL ($p > 0.05$), indicating comparable performance. Some of these methods (*Events*, *2-Gram*, *3-Gram*) display higher variance, suggesting sensitivity to missing rare normal event sequences, which increases false positives and recall variability (see Table IX).

On the BGL dataset, nearly in all settings lightweight methods differ significantly from the DL baselines across all FL configurations. Methods such as *Length* and the *n-gram* variants perform substantially worse than the DL models, while techniques like *ECVC* and *EditDistance* achieve moderately lower F1-scores than DL. In contrast, *Events* and *Events + Length + EditDistance* consistently outperform both DL methods with F1-score of 0.99, indicating that BGL anomalies are relatively easy to identify, as many abnormal sequences contain event types absent from the training data.

To assess the benefit of collaboration through FL, we also evaluated models trained solely on each client's local data (Table III). Comparing these results with the global FL models (Table II) shows that **all methods consistently benefit from aggregation, especially when data are sparsely distributed.** The only exception is the IID setting with five clients, where some methods perform nearly as well as the global model when trained only locally, indicating that each client has sufficient data for effective training (e.g. DL methods on HDFS dataset Table III). As the number of clients increases (10 and 50), local performance drops substantially due to limited data per client. Interestingly, on HDFS, the mean local F1-score is higher in the non-IID case for 10 and 50 clients than in the IID case. This arises because the non-IID setup allocates more training data to some clients, enabling them to train stronger models, which raises the mean but also increases variability across clients.

The local-only results for lightweight methods (Table III) show heterogeneous performance. Methods such as *ECVC*, *n-grams*, *EditDistance*, and *Length* follow the same trend as the DL baselines on HDFS: as the number of clients increases and per-client data decreases, local F1-scores decline. An exception is *Events*, which often performs better with smaller local datasets. On the BGL dataset, however, most lightweight methods struggle when trained locally, yielding lower F1-scores than global models in FL (Table II). Although some clients' local models achieve reasonable effectiveness, high standard deviation, especially under non-IID conditions, indicates that others perform poorly due to limited or skewed data.

Overall, these results show that **FL provides more robust and consistent performance among clients compared to models trained on local data alone, particularly in settings with limited or non-IID data distributions**

RQ1: How do lightweight anomaly detection models compare to centralized and local models in terms of detection effectiveness under IID and non-IID data in a federated setting? *RA1:* Lightweight anomaly detection models achieve identical effectiveness in centralized and federated settings under both IID and non-IID data, due to aggregation strategies that are robust to data distribution shifts across clients. In contrast, locally trained lightweight models exhibit reduced effectiveness and higher variability as data become sparse or skewed with an increasing number of clients, highlighting the clear benefit of federated learning for lightweight techniques.

B. *Efficiency*

In this subsection, we evaluate the results by analyzing the communication cost, focusing on the amount of transferred data, the number of aggregation rounds in FL, as well as the training and inference times of the various anomaly detection methods.

a) Communication cost: Collaboration and communication between clients in federated systems are essential for constructing a global model. However, communication overhead often poses a significant challenge and can become a bottleneck in federated learning (FL) [27]. Notably, **lightweight anomaly detection techniques require only a single aggregation round** in the federated setting, in contrast to DL methods that typically require multiple FL iterations to train an effective model. Table IV reports the number of FL iterations required to train a well-performing anomaly detector, illustrating how often local and global models must be exchanged during the federated learning process. DL approaches require more aggregation rounds to converge, particularly as the number of participating clients increases. When data are distributed among 50 clients, each client holds only a limited amount of training data, which prolongs the optimization process and leads to 4–5 aggregation rounds before convergence (see Table IV). Interestingly, across both datasets, non-IID data require fewer FL iterations to converge than IID data. This can be attributed to the fact that, in the non-IID setting, some clients possess sufficient data to train effective local models, thereby improving the efficiency of the overall FL training process.

TABLE III
F1-SCORE OF **LOCAL** MODELS (MEAN AND STANDARD DEVIATION BETWEEN LOCAL MODELS FROM ALL CLIENTS)

HDFS Dataset

	5 clients		10 clients		50 clients	
	IID	non-IID	IID	non-IID	IID	non-IID
LogAnomaly	**0.92±0.00**	0.42±0.41	0.28±0.0	0.38±0.36	0.09±0.03	0.11±0.03
DeepLog	**0.92±0.00**	0.42±0.41	0.29±0.0	0.37±0.36	0.06±0.00	0.08±0.03
2-Gram	0.86±0.04	0.67±0.27	**0.83±0.04**	0.64±0.28	0.57±0.09	0.45±0.24
3-Gram	0.73±0.04	0.54±0.29	0.62±0.03	0.48±0.25	0.32±0.04	0.28±0.15
ECVC	0.89±0.02	0.70±0.27	0.82±0.03	0.65±0.29	0.56±0.08	0.47±0.22
Edit	0.65±0.03	0.54±0.18	0.64±0.02	0.51±0.20	0.45±0.03	0.36±0.16
Events	0.74±0.12	**0.76±0.13**	0.77±0.09	**0.80±0.05**	**0.89±0.05**	**0.79±0.16**
Length	0.59±0.00	0.57±0.03	0.58±0.01	0.51±0.15	0.54±0.05	0.48±0.13

BGL Dataset

	5 clients		10 clients		50 clients	
	IID	non-IID	IID	non-IID	IID	non-IID
LogAnomaly	0.94±0.01	**0.92±0.04**	0.91±0.01	0.88±0.07	**0.88±0.04**	**0.84±0.08**
DeepLog	0.93±0.06	0.91±0.04	0.90±0.01	0.86±0.06	**0.88±0.03**	0.83±0.09
2-Gram	0.63±0.00	0.63±0.00	0.63±0.00	0.63±0.00	0.63±0.00	0.63±0.00
3-Gram	0.63±0.00	0.63±0.00	0.63±0.00	0.63±0.00	0.63±0.00	0.63±0.00
ECVC	0.89±0.01	0.82±0.11	0.87±0.02	0.81±0.11	0.71±0.15	0.64±0.25
Edit	0.89±0.01	0.81±0.11	0.87±0.01	0.81±0.10	0.73±0.16	0.66±0.25
Events	**0.96±0.0**1	0.89±0.13	**0.94±0.02**	**0.91±0.07**	0.87±0.06	0.81±0.11
Length	0.21±0.16	0.30±0.25	0.27±0.15	0.35±0.22	0.39±0.18	0.40±0.23

Bold values indicates the best-performing models within each setting.

TABLE IV
NUMBER OF FEDERATED LEARNING ITERATIONS REQUIRED TO OBTAIN AN EFFECTIVE GLOBAL MODEL

	HDFS		BGL	
	IID	non-IID	IID	non-IID
LogAnomaly (5 clients)	1	1	2	1
DeepLog (5 clients)	1	1	3	2
LogAnomaly (10 clients)	2	1	4	2
DeepLog (10 clients)	2	2	5	3
LogAnomaly (50 clients)	5	3	5	5
DeepLog (50 clients)	5	4	5	5
Lightweight techniques	1			

All evaluated lightweight techniques require only a single aggregation iteration.

TABLE V
SIZE OF THE RESULTING GLOBAL MODELS IN FEDERATED LEARNING AND INFERENCE TIME ON TEST DATA

	Model size (bytes)		Inference time (sec.)	
	HDFS	BGL	HDFS	BGL
LogAnomaly	609,856	609,856	282.0±3.0	930±11.0
Deeplog	305,728	305,728	193.2±0.9	455±12.0
2-Gram	6,011	10,976	4.20±0.40	1.30±0.30
3-Gram	24,304	27,076	4.00±0.30	1.14±0.03
ECVC	61,003	1,767,145	93.00±6.00	33.0±1.00
Edit	836,211	346,313	30.00±1.00	7.80±0.50
Events	457	1,801	0.80±0.20	0.10±0.00
Length	**56**	**56**	**0.43±0.07**	**0.05±0.01**
2-Gram + Length	6,067	11,032	4.60±0.20	1.20±0.03
Events + Length	513	1,857	1.16±0.08	0.10±0.00
Events + Length + ECVC	61,516	1,769,002	110.0±19.0	38.00±2.00
Events + Length + Edit	836,724	348,170	32.00±2.00	7.90±0.50

Bold values indicates the best-performing models within each setting.

b) Models size: The size of the shared models directly determines the bandwidth needed for communication between clients. While DL models maintain a constant size throughout the FL process, the size of lightweight models depends on the training data and may vary between local and global instances. For instance, in the case of the *Edit* method, the model size is proportional to the number of unique sequences (which can constitute a large portion of the original dataset, particularly when many sequences are rare but distinct), whereas for *Events*, it is governed by the number of unique log events present in the training data, which is typically quite limited. Table V demonstrates that the *Edit* model for the HDFS dataset is larger than for BGL, mirroring the greater number of unique sequences in HDFS. Conversely, the *Events* model for HDFS is smaller than for BGL, due to a lower number of distinct log events. All reported model sizes are given in bytes using Python's default memory representation: lightweight methods store integers (each occupying 28 bytes), while deep learning methods store 32-bit floating-point values (float32). Table V indicates that **most lightweight methods are more memory-efficient than DL models**, the latter being among the most memory-demanding approaches. However, **methods such as *Edit*and *ECVC*may, in some scenarios, reach or even surpass DL models in size**, depending on the size and variability of the training data. Nevertheless, the memory footprint of *Edit* and *ECVC* models remains bounded in practice, as the finite event vocabulary limits the number of distinct patterns that can be stored over time; once the vocabulary is saturated, memory usage stabilizes, with new events mapped to existing representations rather than introducing unbounded growth.

c) Inference Time: Inference time is a critical factor in anomaly detection tasks, as faster detection enables quicker responses to potential threats, thereby enhancing system protection. As shown in Table V, **the lightweight techniques significantly outperform DL methods in terms of inference time**. Among the lightweight methods, *ECVC* has the highest inference time: 93 s on the HDFS test set and 33 s on BGL test set, yet it still outperforms the DL approaches in terms of detection speed,

TABLE VI
TRAINING TIME (IN SECONDS) FOR LOCAL MODELS IN FEDERATED LEARNING FOR HDFS AND BGL DATASETS

HDFS Dataset

	Centralized	5 clients		10 clients		50 clients	
		IID	non-IID	IID	non-IID	IID	non-IID
LogAnomaly	10±0.5	1.90±0.08	6±1.0	1.9±0.2	3.06±0.04	1.71±0.04	1.88±0.23
DeepLog	10±1.0	1.78±0.08	6±0.5	1.8±0.2	5.23±0.79	1.80±0.20	2.52±0.05
2-Gram	0.05±0.01	0.01±0.00	0.02±0.00	0.01±0.00	0.01±0.01	0.0±0.0	0.0±0.0
3-Gram	0.05±0.01	0.01±0.00	0.03±0.01	0.01±0.00	0.02±0.01	0.0±0.0	0.0±0.0
ECVC	0.77±0.08	0.04±0.03	0.21±0.08	0.03±0.02	0.08±0.02	0.02±0.01	0.02±0.01
Edit	0.01±0.00	0.0±0.0	0.01±0.00	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Events	0.01±0.01	0.0±0.0	0.01±0.00	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Length	0.01±0.00	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0

BGL Dataset

	Centralized	5 clients		10 clients		50 clients	
		IID	non-IID	IID	non-IID	IID	non-IID
LogAnomaly	29.36±1.76	25.89±1.66	8.27±7.97	23.92±3.14	10.61±8.11	10.67±5.08	12.14±9.69
DeepLog	21.62±0.70	16.85±1.42	6.50±6.32	14.98±3.05	7.88±6.87	7.43±4.12	5.66±4.61
2-Gram	0.01±0.00	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
3-Gram	0.01±0.00	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
ECVC	0.13±0.06	0.02±0.00	0.03±0.01	0.05±0.03	0.08±0.03	0.02±0.01	0.03±0.01
Edit	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Events	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Length	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0

which require between 3 and 15 minutes to process the test set. In contrast, methods such as *Event* and *Length* achieve the fastest inference times, typically completing within less than a second.

Inference time varies notably across datasets and methods. DL approaches are comparatively faster on the HDFS dataset, whereas most lightweight methods achieve faster inference on the BGL dataset. This difference is driven by dataset characteristics: the average sequence length in HDFS is 29±6, while BGL exhibits a much higher variability at 69±747, indicating the presence of a few extremely long sequences. DeepLog and LogAnomaly rely on LSTM-based architectures that process log entries sequentially, making inference time highly sensitive to sequence length. As a result, despite BGL having a substantially smaller test set than HDFS, DL methods require more inference time due to these long sequences. In contrast, lightweight methods are far less sensitive to sequence length and scale primarily with the size of the test set, leading to higher inference time on the larger HDFS test set. An additional practical advantage is that lightweight techniques do not require GPUs, and their fast inference makes them particularly suitable for deployment in resource-constrained environments, such as IoT systems where computational efficiency is critical.

d) Training time: Client-side computational constraints in FL can pose significant limitations, making it crucial to assess the computational effort required to build an anomaly detection model. While DL methods involve local training, lightweight techniques typically perform lighter operations, such as data transformations, preprocessing, or simple aggregations (e.g., computing the min or max length of a sequence). To provide a fair evaluation for estimating training time at each client, we run the training sequentially in our simulation, ensuring that each client has the same amount of allocated resources (see Section VIII). Table VI shows the training times for the slowest client in each setting, as this client constitutes the bottleneck of the overall runtime.

When data are distributed across multiple clients, the computational load is shared, enabling faster parallel training. DL methods, however, sometimes require multiple FL iterations to train an effective model (e.g., 3–5 iterations with 50 clients), so the reported local training time reflects the total time across these iterations. In a centralized setting, DL methods train on the full dataset in about 10 s on the HDFS dataset and 20–30 s on the BGL dataset (see Table VI). The differences between IID and non-IID settings can be explained by the fact that the slowest client determines the FL bottleneck. In the non-IID setting, this slowest client typically has more training data than the others, which impacts the overall training dynamics. While DL methods training time varies depending on the data distribution among the nodes, from 1.7 to almost 30 seconds, lightweight methods require minimal computation time, less than a second in all considered settings. Therefore, we conclude that **lightweight methods are more efficient in terms of training time than DL and require less computational load in FL environments.**

e) Scalability Analysis: Deep learning models have a fixed parameter size that is independent of the dataset; however, their training and inference costs scale with increasing sequence length, the number of clients, and the repeated communication of model weights between clients and the server. In contrast, lightweight methods construct models whose size may grow with the diversity of observed patterns, such as the number of unique events or subsequences. This growth is bounded by the finite event vocabulary and the observed pattern space of a dataset and, on larger datasets, can lead to increased bandwidth and memory requirements due to the larger amount of information exchanged and aggregated in the federated learning process. Despite this potential variability in model size, our

experimental results show that lightweight methods consistently incur substantially lower computational overhead and faster inference compared to deep learning–based approaches across all evaluated settings.

RQ2: How do lightweight anomaly detection methods compare to DL-based approaches within federated learning in terms of both effectiveness and computational efficiency? *RA2:* In federated learning, lightweight anomaly detection methods can match, and in some cases outperform DL in detection effectiveness. While no single lightweight model consistently outperforms deep learning baselines in all setups and datasets, the Events+Length+EVEC variant comes closest, delivering performance on par with deep learning models on both datasets. Lightweight approaches are more robust to variations in the number of clients and data distribution but are highly dataset-dependent (e.g., the events-based method excels on BGL yet is only moderately effective on HDFS). In terms of the size of the shared models, some of the lightweight approaches are extremely compact, while others (ECVC, Edit) can be more memory intensive as ML models. Nonetheless, all lightweight methods are faster than DL for both inference and training, therefore requiring fewer client-side computational resources.

VII. Confidentiality Analysis in Federated Log Anomaly Detection

In the following section, we identify potential confidentiality risks in our federated log anomaly detection pipeline (Fig. 1). We begin by describing the threat model and outlining potential attack scenarios, followed by a risk classification for each method. Finally, we discuss defense strategies to mitigate residual privacy risks in FL-based anomaly detection, highlighting their role in enhancing system reliability and trustworthiness and discussing their limitations.

A. Threat Model

FL adheres to the principle of data minimization to mitigate privacy risks, as it does not require the sharing of raw sensitive data. In the context of machine learning, models are trained locally on the client and subsequently shared with a central server for aggregation. The lightweight anomaly detection techniques examined in this study differ fundamentally from machine learning models. Unlike DL models, these techniques do not require training; instead, they operate by extracting specific features or parameters directly from the data. In an FL setup, clients are required to share these locally derived features or parameters, which are based on their potentially sensitive data. In the context of log anomaly detection, the primary asset is the log data, whose sensitivity depends on its representation (see Table VII).

In our federated anomaly detection threat model, we therefore consider the following assets.

1) **Raw logs** associate with *the highest risk*, as they may contain operational details (e.g., system activities, user actions, IP addresses, system paths). Exposure of raw logs could compromise system security and lead to violations of data protection regulations. In the federated anomaly detection pipeline, raw logs remain exclusively on the client servers, ensuring that their confidentiality is preserved.
2) **Log templates** (dictionaries mapping log events to specific IDs) are used to transform raw log sequences into sequences of event IDs. In the FL pipeline (see Section III), clients transmit their locally generated templates to the central server, which compiles a global template dictionary. This process ensures that all clients consistently map raw log data into event sequences using a unified scheme, so the preprocessed data can be used for FL. While log templates contain less sensitive information than raw logs, they may still reveal architectural or version-specific details that adversaries could exploit to infer system vulnerabilities. Moreover, parsing errors may accidentally expose confidential parameters in the templates. We therefore classify log template exposure as a *medium-to-low risk*.
3) **Log event sequences**, derived from raw logs using templates, are considered *lower risk* in isolation since they consist only of event IDs. However, they may still fall under privacy regulations (e.g., GDPR, HIPAA) because of their potential to reveal user behavior. The risk increases to *medium* if event sequences are combined with log templates that map event IDs to specific actions. In the FL setting, methods such as Edit Distance and n-gram require clients to share complete event sequences or subsequences with the server.
4) **Log sequence statistical features** are generally considered *low risk*, but they can still leak information about client system behavior. In the FL pipeline, for instance, the *Length Detection* method requires clients to share the minimum and maximum lengths of their log sequences, while the *ECVC* method involves sharing event frequency patterns, and the *Known Events* method requires communicating unique event IDs from clients to the central server. Although these features may appear harmless in isolation, over time, they can reveal usage patterns, workload characteristics, or operational profiles. Moreover, when combined with auxiliary information (e.g., template mappings or external knowledge of system behavior), even simple statistics may allow adversaries to infer sensitive attributes or identify system vulnerabilities.

TABLE VII
Privacy (Priv.) Risks and Mitigations (Mit.) in Federated Log Anomaly Detection

Method	Information shared in FL	Priv.risk	Mit.
Events	IDs of Unique log events	Low	MPSU, DP
ECVS	Events frequencies	Medium	
Edit	Unique log events sequences	Medium	
n-grams	Unique log events sub-sequences	Medium	
Length	Sequences length	Very low	SMPC, HE, DP
DeepLog	Models updates/weights	Low	
LogAnomaly	Models updates/weights	Low	

5) **ML model updates or weights** are exchanged during FL and are considered *low risk*, however, model parameters still encode characteristics of the training data and can leak potentially sensitive information through attacks such as membership inference [53], model inversion [13], or property inference [14].

In our federated anomaly detection pipeline, we consider four attackers scenarios targeting data confidentiality.

a) Scenario 1—Honest-but-curious server: The central server follows the prescribed federated learning protocol but seeks to gain additional insights into clients' private data. It has access to the model updates submitted by each client and may attempt inference attacks on these local ML models to recover sensitive information about the underlying log data. In the case of lightweight anomaly detection, the server receives abstracted representations from all clients, including e.g., log templates, event sequences, or statistical features—which, although less detailed than raw logs, may still disclose sensitive information about client systems.

b) Scenario 2—Honest-but-curious client: A participating client correctly follows the protocol but seeks to extract information about other clients' data. It can observe the global model distributed by the server after aggregation and may conduct inference attacks on the global model to learn sensitive properties of other participants' training data. In the case of lightweight anomaly detection, the client may also exploit access to the global template dictionary, aggregated event sequences, or shared statistical features to deduce confidential information about other systems.

c) Scenario 3—Malicious client: A client intentionally deviates from the protocol to actively attack the confidentiality of other clients' data. This may involve crafting and sending arbitrary or manipulated local models to the server in order to influence the global model in a way that increases the success rate of inference attacks on the training data of other participants.

d) Scenario 4—Colluding parties: In this scenario, multiple clients, or a client colluding with the central server, collaborate to infer sensitive information about other participants' data. Such collusion increases the adversaries' attack surface: they may combine insights from local model updates, the aggregated global model, and shared data representations (e.g., templates, event sequences, or statistical features) to build stronger attack models.

Confidentiality risks have significant implications for the reliability and trustworthiness of FL-based anomaly detection systems. Privacy leakage can erode stakeholder confidence, undermining the willingness to deploy such solutions in critical infrastructures. Collusion among clients or between clients and the server further reduces resilience, as coordinated adversaries can leverage combined knowledge to extract sensitive information or destabilize training. Beyond confidentiality, security risks also threaten system availability: poisoning attacks [2] by malicious clients or a compromised server may degrade detection accuracy, increasing the risk of undetected anomalies and subsequent system failures. Furthermore, the reliance on a central server introduces a single point of failure, where its disruption could simultaneously impact availability and amplify privacy breaches. Addressing these risks is thus essential not only for safeguarding confidentiality, but also for ensuring the dependable, resilient, and trustworthy operation of federated anomaly detection in real-world environments.

TABLE VIII
MITIGATIONS (MIT.) AGAINST PRIVACY RISKS THEIR LIMITATIONS IN FL

Mit.	Protection level	Limitations
HE, SMPC, MPSU	Honest-but-curious server	Computational overhead, communication overhead
DP	Honest-but-curious server/client, malicious client, colluding parties	Computational overhead, privacy-utility trade-off

B. Mitigation Strategies

While FL reduces exposure by keeping raw logs local, the exchange of model updates, templates, or statistical features may still pose residual privacy risks. To address these vulnerabilities, we now turn to mitigation strategies and discuss countermeasures aimed at preserving confidentiality, ensuring availability, and strengthening the overall dependability of federated anomaly detection in real-world deployments. Table VIII presents an overview of mitigation techniques and their limitations.

Secure Multi-Party Computation (SMPC) [6] and **Homomorphic Encryption** (HE) [48] can be applied in FL to protect local ML models from inference attacks, thereby mitigating risks in scenarios involving an adversarial server. SMPC enables clients to jointly compute a global model without revealing their individual local models to any other party. HE allows computations to be performed directly on encrypted data: Clients encrypt their local models, the server aggregates them in encrypted form, and the result is sent back for decryption by the clients. An SMPC protocol can also be used for lightweight techniques, e.g., to compute the minimum or maximum of values distributed among parties without revealing individual inputs in the context of the *Length Detection* Method. This can be achieved using techniques such as secret sharing [51]. HE can also be applied to compute the minimum and maximum of sequence lengths while preserving client privacy in federated learning settings [9].

Anomaly detection methods such as Known Events, Event Count Vector Clustering, Edit Distance, and n-grams typically require transmitting sets of events or sequences to a central server. Similarly, preprocessing log data involves sharing log templates so that the server can construct a global dictionary through a union operation. To safeguard these data exchanges while still enabling set union, **Multi-Party Private Set Union** (MPSU) protocols can be employed. MPSU allows multiple parties to compute the union of their sets without revealing any information about the individual inputs. Some MPSU protocols rely on secret sharing [36], while others are based on HE [22]. More recent approaches, such as Membership Oblivious Transfer [15] and the protocol proposed by Bao at al. [3], avoid computationally intensive operations like HE and SMPC, achieving significant improvements in runtime and bandwidth usage compared to earlier protocols.

TABLE IX
PRECISION AND RECALL OF GLOBAL MODELS IN FEDERATED LEARNING

	HDFS				BGL			
	Precision		Recall		Precision		Recall	
	IID	non-IID	IID	non-IID	IID	non-IID	IID	non-IID
LogAnomaly (centralized)	0.87±0.0		**0.99±0.0**		0.97±0.0		0.97±0.0	
DeepLog (centralized)	0.87±0.0		**0.99±0.0**		0.97±0.0		0.97±0.0	
LogAnomaly (5 clients)	0.86±0.01	0.87±0.0	**0.99±0.0**	**0.99±0.0**	0.97±0.0	0.97±0.0	0.97±0.0	0.97±0.0
DeepLog (5 clients)	0.87±0.0	0.84±0.01	**0.99±0.0**	**0.99±0.01**	0.97±0.0	0.97±0.0	0.97±0.0	0.97±0.0
LogAnomaly (10 clients)	0.87±0.0	0.86±0.0	**0.99±0.0**	**0.99±0.0**	0.97±0.0	0.97±0.0	0.97±0.0	0.97±0.0
DeepLog (10 clients)	0.87±0.0	0.86±0.0	**0.99±0.0**	**0.99±0.0**	0.97±0.01	0.97±0.0	0.97±0.0	0.97±0.0
LogAnomaly (50 clients)	0.87±0.0	0.87±0.0	**0.99±0.0**	**0.99±0.0**	0.95±0.0	0.96±0.0	0.97±0.0	0.97±0.0
DeepLog (50 clients)	0.87±0.0	0.87±0.0	**0.99±0.0**	**0.99±0.0**	0.94±0.01	0.94±0.0	0.97±0.0	0.97±0.0
2-Gram	0.93±0.02		0.80±0.17		0.46±0.00		**1.00±0.00**	
3-Gram	0.81±0.03		0.89±0.12		0.46±0.00		**1.00±0.00**	
ECVC	0.93±0.01		0.97±0.01		0.90±0.01		0.96±0.01	
Edit	0.93±0.00		0.59±0.00		0.91±0.00		0.94±0.00	
Events	**0.99±0.01**		0.46±0.17		0.98±0.01		**1.00±0.00**	
Length	**0.99±0.01**		0.40±0.03		0.80±0.07		0.01±0.01	
2-Gram Length	0.93±0.01		0.83±0.17		0.46±0.00		**1.00±0.00**	
Events Length	0.98±0.01		0.66±0.18		**0.98±0.01**		**1.00±0.00**	
Events Length ECVC	0.92±0.0		0.98±0.02		0.90±0.01		**1.00±0.00**	
Events Length Edit	0.94±0.01		0.82±0.15		0.91±0.01		**1.00±0.00**	

Bold values indicates the best-performing models within each setting.

Limitations: Most MPSU protocols operate under the semi-honest model [3], [15], [36], where participants are assumed to follow the protocol but may attempt to infer additional information. SMPC, HE, and MPSU offer strong confidentiality guarantees but introduce significant computational and communication overhead, which can hinder scalability and reduce availability in large-scale deployments. This overhead usually increases with the size of the data being processed and the number of participants (n). Total communication can be $O(n^2)$, since many protocols require all-to-all communication. In a large-scale FL setting, this can become a bottleneck, slowing down the system and making it less scalable. While these techniques effectively protect local models in FL, the aggregated global model may still remain vulnerable to privacy leakage.

Differential Privacy (DP) [11] is a widely adopted technique for mitigating privacy leakage in FL by introducing carefully calibrated noise during federated training [43]. DP can protect both local and global models, thereby reducing the risk of inference attacks from adversarial clients or servers. DP in FL can be implemented either through central DP, where noise is added to the global model at the server, or local DP, where each client perturbs its model updates before transmission [47]. In ML-based approaches, DP can be applied through methods such as DP-SGD [1], DP-FedAvg algorithm [43], noising before model aggregation FL (NbAFL) [59]. For lightweight anomaly detection methods, DP can be applied by perturbing derived statistics (e.g., sequence length or events frequencies) or by injecting noise directly into sequential log data before aggregation.

Limitations: The main challenge lies in balancing privacy guarantees against model utility, as excessive noise can significantly reduce anomaly detection performance [47], [49], potentially rendering the model ineffective. Selecting an appropriate privacy-loss parameter (often denoted as ϵ) requires empirical tuning, since the privacy–utility tradeoff is highly dependent on the specific data and application context. DP also introduces computational overhead, though typically less severe than HE or SMPC, making it more practical for resource-constrained environments such as IoT.

RQ3: What disclosure risks arise when performing anomaly detection in federated learning, and how can these risks be mitigated? *RA3:* Some lightweight methods considered in this work may expose sensitive information by transmitting portions of the input data during aggregation. Deep learning models can leak sensitive information about training data through the models shared in federated settings. To mitigate these risks in confidentiality-sensitive applications, additional privacy-preserving mechanisms are required: MPSU, SMPC, and HE can secure the aggregation of lightweight methods, while SMPC, HE, and DP can protect deep learning models. Although SMPC, HE, and MPSU offer strong confidentiality, their computational overhead can limit scalability and availability; DP introduces a privacy–utility tradeoff that may reduce detection accuracy. Therefore, selecting an appropriate mitigation requires balancing confidentiality with system reliability, efficiency, and scalability for real-world deployment.

VIII. THREATS TO VALIDITY

In this section, we will examine different risks that could compromise the validity of this study.

A. Reproducibility Between Studies

Different works that use an identical dataset may yield varying results. For example, the DeepLog method applied to the HDFS dataset exhibits distinct F1 scores: LogBert work [17] reports

a score of 0.77, Landauer et al. [30] a score of 0.88, while our findings show a performance of 0.93. We recognize the challenge of comparing results across different studies within this domain. Therefore, in this study, we exclusively compare outcomes generated by our codebase, employing averages from multiple runs, and maintaining strict control over the dataset's partitioning. As a result, direct comparison of reported metrics across papers is often unreliable. To ensure fairness and reproducibility, we evaluate all methods within a unified implementation, using identical preprocessing steps, controlled data partitioning, and averaged results over multiple runs.

B. Training Multiple Clients on Single Machine

This study involved training multiple clients within a federated process on a single computer. Although this arrangement does not have a significant impact on most metrics, it does affect the training time, as all clients were executed concurrently. We recognize that this parallelization affects training times, and the results may differ if the clients were distributed across multiple computers in a real-world FL setup. This is the reason why, for estimating local models' training time, clients were executed sequentially and independently.

IX. Conclusions and Future Work

This research develops a federated pipeline for log anomaly detection that integrates both deep learning and lightweight models. We systematically evaluate these methods across varying numbers of clients, datasets, and distributions (IID and non-IID), while analyzing communication and computational overhead, including model size, aggregation rounds, and training and inference times. We further assess privacy risks by defining a threat model and proposing mitigation strategies tailored to confidentiality-sensitive deployments. Our results show that lightweight methods can match or exceed the detection performance of DL models while offering substantially faster training and inference and greater robustness across heterogeneous FL settings. Although LogAnomaly and DeepLog achieve competitive F1-scores and moderate memory usage, their training and inference times make them unsuitable for time-critical applications [40], where lightweight methods offer a viable and efficient alternative. Finally, we demonstrate that federated learning consistently outperforms isolated local training, particularly when client data are limited or non-IID.

Future work includes incorporating more recent log datasets that better reflect modern cloud and distributed environments and are suitable for federated evaluation. We also plan to integrate privacy-preserving mechanisms such as MPSU and DP and empirically assess their impact on effectiveness and efficiency of federated anomaly detection process. In addition, we aim to develop and evaluate federated parsing and template synchronization modules to build a practical, end-to-end, industrial-grade pipeline that ensures reliability, resilience, and trustworthiness in real-world deployments such as supervising vital infrastructure or IoT systems that span multiple allied countries or organizations [61].

Acknowledgment

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Computer Commun. Secur.*, New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 308–318.
[2] E. Bagdasaryan et al., "How to backdoor federated learning," in *Proc. Twenty Third Inter. Conf. Artif. Intell. Statist.*, Jun. 2020, pp. 2938–2948.
[3] Y. Bao, L. Wei, and D. Han, "Efficient and secure multi-party private set operation protocol suitable for IoT devices," in *Proc. 2023 Int. Conf. Artif. Intell., Syst. Netw. Secur.*, New York, NY, USA: Association for Computing Machinery, Jun. 2024, pp. 283–290.
[4] V. Beck et al., "SoK: LLM-based log parsing," 2025, *arXiv:2504.04877*.
[5] D. J. Beutel et al., "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.
[6] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA: Association for Comp. Machinery, Oct. 2017, pp. 1175–1191.
[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.
[8] L. Cui et al., "Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3492–3500, May 2022.
[9] W. Ding, Z. Yan, X. Qian, and R. H. Deng, "Computing maximum and minimum with privacy preservation and flexible access control," in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2019, pp. 1–7.
[10] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Computer Commun. Secur.*, New York, NY, USA: Association for Computing Machinery, 2017, pp. 1285–1298.
[11] C. Dwork, *Differential Privacy. In Automata, Languages and Programming*, vol. 4052 Lecture Notes in Computer Science. Berlin, Heidelberg, Germany: Springer, 2006, pp. 1–12.
[12] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proc. IEEE Symp. Secur. Privacy*. IEEE, 1996, pp. 120–128.
[13] M. Fredrikson et al., "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Computer Commun. Secur.*, New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1322–1333.
[14] K. Ganju et al., "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM SIGSAC Conf. Computer Commun. Secur.*, New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 619–633.
[15] J. Gao et al., "Toward a practical multi-party private set union," in *Proc. PETS*, 2024, pp. 622–635.
[16] A. G. Gómez et al., "Collaborative anomaly detection in log data: Comparative analysis and evaluation framework," *Future Gener. Comp. Syst.*, vol. 175, 2025, Art. no. 108090.
[17] H. Guo et al., "LogBERT: Log ANOMALY DETECTIon via BERT," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2021, pp. 1–8.
[18] Y. Guo et al., "Anomaly detection using distributed log data: A lightweight federated learning approach," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2021, pp. 1–8.
[19] X. Han, S. Yuan, and M. Trabelsi, "LogGPT: Log anomaly detection via GPT," in *Proc. IEEE Inter. Conf. Big Data*, Dec. 2023, pp. 1117–1122.
[20] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proc. Int. Conf. Web Serv.*. IEEE, 2017, pp. 33–40.
[21] P. Himler et al., "Anomaly detection in log-event sequences: A federated deep learning approach and open challenges," *Mach. Learn. with Appl.*, vol. 16, Jun. 2024, Art. no. 100554.
[22] J. Hong et al., "Constant-Round Privacy Preserving Multiset Union," 2011. Publication info: Published elsewhere. Unknown where it was published.

[23] T. T. Huong et al., "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Comput. Ind.*, vol. 132, Nov. 2021, Art. no. 103509.
[24] T. T. Huong et al., "LockEdge: Low-complexity cyberattack detection in IoT edge computing," *IEEE Access*, vol. 9, pp. 29696–29710, 2021.
[25] Y. Ji et al., "Adapting large language models to log analysis with interpretable domain knowledge," in *Proc. 34th ACM Int. Conf. Inf. Knowl. Manag.*, 2025, pp. 1135–1144.
[26] J. Jithish et al., "Distributed anomaly detection in smart grids: A federated learning-based approach," *IEEE Access*, vol. 11, pp. 7157–7179, 2023.
[27] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1-2, pp. 1–210, Jun. 2021.
[28] M. Landauer et al., "AMiner A modular log data analysis pipeline for anomaly-based intrusion detection," *Digit. Threats*, vol. 4, no. 1, pp. 12:1–12:16, Mar. 2023.
[29] M. Landauer et al., "Deep learning for anomaly detection in log data: A survey," *Mach. Learn. with Appl.*, vol. 12, Art. no. 100470, 2023.
[30] M. Landauer, F. Skopik, and M. Wurzenberger, "A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques," in *Proc. ACM Softw. Eng.*, 2024, vol. 1(FSE), pp. 1354–1375.
[31] V.-H. Le et al., "PreLog A pre-trained model for log analytics," *Proc. ACM Manage. Data Data*, vol. 2, no. 3, pp. 1–28, 2024.
[32] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *Proc. 36th IEEE/ACM Int. Conf. Automated Soft. Eng.*. IEEE, 2021, pp. 492–504.
[33] V.-H. Le and H. Zhang, "Log-based anomaly detection with deep learning: How far are we?," in *Proc. 44th Int. Conf. Soft. Eng.*, New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 1356–1367.
[34] V. I. Levenshtein et al., "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, pp. 707–710. Soviet Union, 1966.
[35] B. Li, S. Ma, R. Deng, K. -K. R. Choo, and J. Yang, "Federated Anomaly detection on system logs for the Internet of Things: A customizable and communication-efficient approach," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 1705–1716, Jun. 2022.
[36] X. Liu and Y. Gao, "Scalable multi-party private set union from multi-query secret-shared private membership test," in *Adv. Cryptology–ASIACRYPT 29th Inter. Conf. Theory Appl. Cryptology Inf. Secur.*, Guangzhou, China, Dec. 2023, pp. 237–271.
[37] Y. Liu et al., "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.
[38] Y. Liu et al., "Interpretable online log analysis using large language models with prompt strategies," in *Proc. 32nd IEEE/ACM Inter. Conf. Prog. Comprehension*, pp. 35–46, 2024.
[39] S. Lu, X. Wei, Y. Li, and L. Wang, "Detecting anomaly in Big Data system logs using convolutional neural network," in *Proc. IEEE 16th Int. Conf Dependable, Autonomic Secure Comput.*. IEEE, 2018, pp. 151–158.
[40] X. Ma et al., "Practitioners' expectations on log anomaly detection," *IEEE Trans. Softw. Eng.*, vol. 51, no. 9, 2025, pp. 2455–2471.
[41] W. Marfo et al., "Federated learning for efficient condition monitoring and anomaly detection in industrial cyber-physical systems," in *Proc. Int. Conf. Comput., Netw. Commun.*, Honolulu, HI, USA, 2025, pp. 740–746.
[42] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[43] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. 6th Int. Conf. Learn. Representations, ICLR 2018*, Vancouver, BC, Canada, Apr. 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018, pp. 1926–1939.
[44] W. Meng et al., "Loganomaly Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, vol. 19, pp. 4739–4745.
[45] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-Learning-Based Anomaly Detection for IoT Security Attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022 .
[46] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *Proc. 37th Annu. IFIP Int. Conf. Dependable Syst. Netw.*. IEEE, 2007, pp. 575–584.
[47] A. El Ouadrhiri et al., "Differential privacy for deep and federated learning: A survey," *IEEE Access*, 10, pp. 22359–22380, 2022.
[48] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[49] N. Ponomareva et al., "How to DP-fy ML: A practical guide to machine learning with differential privacy," *J. Artif. Intell. Res.*, vol. 77, pp. 1113–1201, Jul. 2023.
[50] R. Abdel Sater and A. Ben Hamza, "A federated learning approach to anomaly detection in smart buildings," *ACM Trans. Internet Things*, vol. 2, no. 4, pp. 28:1–28:23, Aug. 2021.
[51] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[52] T.-H. Shin and S.-H. Kim, "Utility Analysis about Log Data Anomaly Detection Based on Federated Learning," *Appl. Sci.*, vol. 13, no. 7, Jan. 2023, Art. no. 4495.
[53] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, Los Alamitos, CA, USA, May 2017, pp. 3–18.
[54] B. Sudqi Khater et al., "A lightweight perceptron-based intrusion detection system for fog computing," *Appl. Sci.*, vol. 9, no. 1, Jan. 2019, Art. no. 178.
[55] S. Tao et al., "BigLog: Unsupervised large-scale pre-training for a unified log representation," in *Proc. IEEE/ACM 31st Inter. Symp. Qual. Service*. IEEE, 2023, pp. 1–11.
[56] H. T. Truong et al., "Light-weight federated learning-based anomaly detection for time-series data in industrial control systems," *Comput. Ind.*, vol. 140, Sep. 2022, Art. no. 103692.
[57] X. Wang et al., "Toward: Accurate anomaly detection in industrial Internet of Things using hierarchical federated learning," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7110–7119, May 2022.
[58] A. A. Wardana and P. Sukarno, "Taxonomy and survey of collaborative intrusion detection system using federated learning," *ACM Comput. Surv.*, vol. 57, no. 4, pp. 1–36, 2024.
[59] K. Wei et al., "Federated: Learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.
[60] M. Weiss and P. Tonella, "Simple techniques work surprisingly well for neural network test prioritization and active learning (Replicability study)," in *Proc. 31st ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2022, pp. 139–150.
[61] M. Wurzenberger et al., "Newsroom Towards automating cyber situational awareness processes and tools for cyber defence," in *Proc. 19th Int. Conf. Availability, Rel. Secur.*, 2024, pp. 1–11.
[62] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, "Mining console logs for large-scale system problem detection," *SysML*, vol. 8, pp. 4–4, 2008.
[63] L. Yang et al., "Semi-supervised log-based anomaly detection via probabilistic label estimation," in *Proc. IEEE/ACM 43rd Inter. Conf. Softw. Eng.*, 2021, pp. 1448–1460.
[64] Z. Yang et al., "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Comput. Secur.*, vol. 116,, 2022, Art. no. 102675.
[65] B. Yu et al., "Deep learning or classical machine learning? An empirical study on log-based anomaly detection," in *Proc. 46th IEEE/ACM Inter. Conf. Softw. Eng.*, 2024, pp. 1–13.
[66] D. Zeng, S. Liang, X. Hu, H. Wang, and Z. Xu, "FedLab: A flexible federated learning framework," *J. Mach. Learn. Res.*, vol. 24, no. 100, pp. 1–7, 2023.
[67] X. Zhang et al., "Robust log-based anomaly detection on unstable log data," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Foundations Softw. Eng.*, 2019, pp. 807–817.
[68] H. Zhu et al., "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, Nov. 2021.