Towards Securing IIoT: An Innovative Privacy-Preserving Anomaly Detector Based on Federated Learning

Samira Kamali Poorazad, Chafika Benzaïd, and Tarik Taleb

***Abstract*—In the light of the growing connectivity and sensitivity of industrial data, cyberattacks and data breaches are becoming more common in the Industrial Internet of Things (IIoT). To cope with such threats, this study presents an anomaly detection system based on a novel Federated Learning (FL) framework. This system detects anomalies such as cyberattacks and protects industrial data privacy by processing data locally and training anomaly detection models on industrial agents without sharing raw data. The proposed FL framework incorporates two key components to enhance both privacy and efficiency. The first component is Homomorphic Encryption (HE), which is integrated into the framework to further protect sensitive data transmissions such as model parameters. HE enhances privacy in FL by preventing adversaries from inferring private industrial data through attacks, such as model inversion attacks. The second component is an innovative dynamic agent selection scheme, wherein a selection threshold is calculated based on agent delays and data size. The purpose of this new scheme is to mitigate the straggler effect and the communication bottleneck that occur in traditional FL architectures, such as synchronous and asynchronous architectures. It ensures that agents are not unfairly selected by the different delays resulting from heterogeneous data in IIoT environments, while simultaneously improving model performance and convergence speed. The proposed framework exhibits superior performance over baseline approaches in terms of accuracy, precision, F1-scores, communication costs, convergence speeds, and fairness rate.**

***Index Terms*—Federated Learning, Privacy-preserving, Industrial Internet of Things, and Anomaly Detection.**

I. Introduction

MANUFACTURING and industrial sectors use Internet of Things (IoT) technologies to automate processes and improve product quality through Industrial Internet of Things (IIoT) [1]. IIoT enhances productivity and scalability through intelligent interconnection and remote management [1]. However, due to the inherent broadcast nature of wireless communications, IIoT presents cybersecurity risks, such as command injection attacks [2]. These attacks take the form of anomalies – unexpected deviations from the system behavior – which may indicate malicious activities or system malfunctions. The Stuxnet on Iran's nuclear power plant in 2010 is a high-profile example of these attacks [3]. The risk of such attacks is especially pronounced in older industrial systems, which were not originally designed with security in mind. [4]. A robust anomaly detection system that continuously monitors and identifies potential attacks based on data flows in IIoT is crucial for mitigating potential security vulnerabilities. In this vein, many IIoT environments have recently adopted centralized machine learning-based anomaly detection methods [5]. The use of these centralized approaches provides significant benefits in terms of improved model accuracy and ease of deployment [6]. However, they also introduce communication inefficiencies and raise potential privacy concerns, as large volumes of IIoT data must be transferred and processed at a central server [6]. In fact, requiring a central server to manage all data from IIoT agents increases the risks of data breaches and establishes a single point of failure [7]. Furthermore, IIoT data holders tend to avoid sharing sensitive information with third parties. Therefore, a practical, distributed anomaly detection system that protects data privacy in IIoT environments is imperative.

Federated Learning (FL), a form of distributed machine learning, offers a promising solution to meet the aforementioned need [8]. Through FL, industrial agents are able to work collaboratively to train global models by transmitting parameters and local models to a central server rather than sharing raw training data [7]. Compared to centralized machine learning approaches, FL does not only safeguard privacy but also significantly reduces communication overhead, since sending only model parameters is much more efficient than sending the original training data. While FL significantly reduces privacy risk, adversaries can still compromise sensitive data by intercepting communication channels or compromising the aggregator server [9]. Thus, FL remains vulnerable to critical threats such as model poisoning and model inference attacks [9], [10]. It is crucial that FL's privacy be further enhanced, particularly in the context of IIoT. Multiple secure FL schemes have been developed to address this need, including differential privacy (DP), Multiparty Computation (MPC), and homomorphic encryption (HE) [11].

The choice of communication mode –synchronous or asynchronous– is as critical as addressing privacy concerns in FL frameworks. In heterogeneous IIoT environments, where devices operate at different speeds, due to varying computational resources and diverse data, the type of communication mode becomes even more significant. As an example, in FL with synchronous communication mode, all industrial agents must upload their local models to the server at the same time for aggregation [12]. This requirement forces the server to wait

Samira Kamali Poorazad and Chafika Benzaïd are with the Centre for Wireless Communications, University of Oulu, Oulu, Finland (emails: samira.kamalipoorazad@oulu.fi, chafika.benzaid@oulu.fi).

Tarik Taleb is with the Department of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany (email: tarik.taleb@rub.de).

for the slowest device, commonly referred to as the *straggler*, which delays the entire training process. This delay, known as the *straggler effect*, makes Syncrounou Federated learning (SyncFL) unsuitable for real-time or heterogeneous IIoT applications, as it significantly slows down the convergence speed of the model and reduces the training efficiency of the anomaly detection model. The asynchronous FL (AsyncFL) [13] was introduced to avoid straggler effects by performing global aggregation right after a local model has been received. Despite the frequent model transfer aggregations, AsyncFL approaches may be problematic for IIoT due to *communication bottlenecks* caused by agents communicating with the aggregation server at different times, rather than in a coordinated manner. One solution for balancing SyncFL and AsyncFL is to use buffered-based solutions. Fed-Buff [14] is an example of buffered FL, wherein local model updates are processed after a K-size buffer is filled at the server. A disadvantage of Fed-Buff is that it could favor agents with fast training speeds and results in low model accuracy. Recent research [15] proposes a Buffered FL (BFL) and an agent selection method based on the training time of agents in order to fill the gap in Fed-Buff. However, the authors in [15] considered only training time and used the agent selection method exclusively in the first training round, leading to two significant challenges. First, considering that the computing capabilities of IIoT agents can fluctuate over time due to resource heterogeneity, selecting agents only during the first round of FL is neither dynamic nor efficient. Agents not selected in the initial round may have sufficient resources in later rounds and should be reconsidered for selection in subsequent rounds. Second, other factors, such as the communication delay, which is a variable factor between the agents and the central server should be considered. Therefore, focusing solely on training time is not sufficient. Consequently, there is a need for a comprehensive FL framework that addresses the highly heterogeneous IIoT environment by balancing the trade-offs between model convergence speed, accuracy, and varying agent speeds.

As a remedy to the above-mentioned challenges, a novel Dynamic HE-based FL (DyHFL) framework for the detection of IIoT anomalies is developed. HE and an innovative Dynamic Agent Selection method are used to address three critical challenges, namely: privacy preservation, stragglers, and communication bottlenecks. HE is chosen for its ability to preserve both model accuracy and data privacy, whereas DP, despite its privacy benefits, tends to degrade model accuracy due to the introduced noise. MPC was excluded due to its high computational and communication overhead from constant data exchanges, causing delays that are unsuitable for real-time IIoT applications that require low-latency responses. In contrast, HE performs computations on encrypted data without continuous interaction, reducing overhead and making it more efficient for real-time IIoT environments. Furthermore, a novel **Dynamic Agent Selection** strategy is proposed to overcome the limitations of previous buffered FL methods, such as the static agent selection in BFL [15] and the fixed buffer-based aggregation in FedBuff [14].

Unlike BFL [15], which only considers training time and performs agent selection once at initialization, DyHFL introduces a *sliding window-based mechanism* that dynamically adjusts agent selection at every training round. This mechanism continuously evaluates agent performance based on three IIoT-relevant metrics: (1) *training time*, which reflects computational capacity, (2) *communication time*, which reflects the total round-trip latency, including model upload/download transmission with network latency, and (3) *local data size*, which represents workload imbalance. This tri-metric evaluation enables DyHFL to adapt to heterogeneous IIoT devices with varying resources and conditions. In contrast to FedBuff [14], which fills buffers solely based on the arrival order of updates—often favoring faster devices—DyHFL employs a threshold-driven buffer update policy that allows aggregation to proceed without waiting for all agents. To operationalize this policy, DyHFL introduces two key functions—Weighted Average Metrics (WAM) [15] and Exponentially Weighted Moving Average (EWA). These functions combine three IIoT-relevant metrics (training time, communication time, and local data size) to compute a dynamic time threshold, which is then used to categorize agents as fast or slow. In this way, DyHFL ensures a fair representation of both groups in the aggregation buffer, rather than exhibiting bias toward faster agents as in FedBuff. This work mainly contributes to:

1) Proposing a new method of anomaly detection that combines deep learning (DL) and HE to detect cyber threats in industrial cyber-physical systems (CPS).
2) Developing a novel FL framework that amalgamates the potential of synchronous and buffered FL approaches to effectively address straggler and communication issues while accounting for the heterogeneous nature of data and resources in IIoT environments.
3) Designing an adaptive and dynamic agent selection method that continuously monitors training time, communication delay, and data size to fairly balance participation between fast and slow agents across training rounds.
4) Evaluating the proposed DyHFL framework using three distinct industrial datasets to demonstrate its generalization ability, robustness, and superior performance compared to state-of-the-art FL baselines.

The remainder of this article is organized as follows. Section II discusses some related work in the literature. Section III introduces the system model, the attack model, and the proposed privacy-preserving FL method. Section IV discusses security analysis and communication complexity. Section V presents implementation details and discusses the evaluation results. Section VI highlights limitations and potential future directions. This article concludes in Section VII.

II. Related Work

This section briefly reviews some studies focusing on FL-based anomaly detection systems for IIoT environments.

A. FL-based Intrusion/Anomaly Detection

1) SyncFL approaches: In [7], the authors proposed a federated self-learning system for detecting malicious devices in IoT. Gated Recurrent Units (GRUs) are used in the system

to classify data based on thresholds. Additionally, the self-learning mechanism is leveraged to enhance the detection performance of the global model as the IoT environment changes. For detecting energy efficiency anomalies in smart buildings, the authors of [6] proposed a FL approach based on Long Short-Term Memory (LSTM). In [4], a Variational Autoencoder-LSTM model is used to detect anomalies in industrial control systems (ICS). The authors of [5] devised a FL framework based on Convolutional neural network (CNN)-LSTM to reduce communication costs through gradient compression and local computations, wherein the Top-k algorithm is used to identify and send the "k" largest gradients for communication. The work in [16] presents a combination of One-Class Support Vector Machine (OCSVM) and Isolation Forests (IF) to detect potentially malicious data points in IIoT. Stacked Autoencoders (SAE) are also used to extract features from data to help identify patterns and relationships.

The authors of [17] propose a FL-based anomaly detection framework for smart grids, enabling smart meters to train local models without sharing data, ensuring privacy while maintaining comparable performance to centralized models. It evaluates seven ML classifiers, demonstrating low resource consumption, but lacks advanced privacy techniques (e.g., HE). In [18] a FL framework for anomaly detection in IoT networks, integrating mutual information for feature selection and a deep neural network (DNN) for intrusion detection is proposed. It employs a mini-batch aggregation scheme to train models across distributed IoT devices, ensuring data privacy and high accuracy. However, the work in [18] lacks robust privacy mechanisms against model inversion attacks, and favoring fast and well-connected devices due to its mini-batch aggregation approach.

Aside from the lack of privacy-preserving methods, the discussed synchronous methods are particularly susceptible to straggler effects when there are more agents involved. In real-time industrial domains, stragglers cause delays and slow convergence, thereby undermining the timely detection and mitigation of anomalies.

2) AsyncFL approaches: There are often impractical assumptions behind synchronized schemes. First of all, they require all nodes to transmit weights during every training round, resulting in significant network congestion. Second, these schemes require waiting for the slowest agents (i.e., stragglers), resulting in substantial delays. To address these limitations, AsyncFL approaches have been explored as a more efficient alternative. For example, in [19], the authors proposed an AsyncFL-based digital twin architecture for IIoT applications to minimize straggler effects. Results showed faster convergence and higher learning rates with the suggested model. Authors in [20] introduced an AsyncFL scheme based on dynamically selected agents for heterogeneous IoT devices, which optimizes training by taking into account computing resources and network conditions. Compared to synchronous methods, the proposed AsyncFL is more efficient and accurate, and offers better scalability and robustness, especially in non-identical data scenarios. However, it introduces complexity and potential staleness. In [21], the authors proposed a spectral clustering method based on the latency and direction of model updates to prevent model staleness. In non-independent and identically distributed datasets, the scheme also improved accuracy and convergence speed. In [22], through the use of a novel aggregation algorithm combined with a cache structure, the authors developed a Semi-Asynchronous Federated Averaging (SAFA) method to solve low round efficiency and poor convergence. Authors in [23] suggested an AsyncFL model to detect low-rate Distributed Denial-of-Service (DDoS) attacks. This approach uses bidirectional LSTMs and an attention mechanism to improve detection accuracy by addressing missing data issues and ensuring the model learns from past and future data. The Proposed AsyncFL framework enables asynchronous updates across agents, reducing the impact of abnormal parameters and improving robustness. Study results indicate that the model outperforms state-of-the-art methods in terms of both accuracy and communication overhead, while potential implementation complexity was identified. Authors in [24] introduce DEAFL-ID, a delay and energy-efficient AFL framework designed for intrusion detection in heterogeneous IIoT. DEAFL-ID leverages deep Q-learning (DQN) for optimal device selection to minimize training costs while preserving detection accuracy. However, the DEAFL-ID framework tends to prioritize faster devices since the selection criteria favor lower delay, higher energy efficiency, and better detection accuracy, which may introduce bias in the global model.

According to the investigated studies, asynchronous approaches are more accurate, scalable, and efficient in complex and non-uniform data environments. Nevertheless, despite their advantages, these methods are not without challenges, including staleness, increased complexity, and heightened communication costs. The server may slow down model convergence if it has to aggregate frequently. As a result, it is necessary to balance the advantages and disadvantages of AsyncFL approaches.

3) Buffered FL approaches: Buffer-based approaches have been introduced to balance the limitations of SyncFL and AsyncFL frameworks [14], [15]. However, these methods come with their own drawbacks. In the approach proposed by [14], the authors overlooked the issue of fairness among agents, which can lead to the buffer being filled by faster agents, potentially sidelining slower, yet valuable contributors. On the other hand, while the work in [15] addressed fairness through an agent selection strategy, this solution has its limitations as it only applies during the first training round, lacking the flexibility to adapt dynamically throughout the learning process. Overall, there is a clear need for a more dynamic and efficient buffer-based solution to mitigate straggler by continuously adapting to the varying capabilities of agents, ensuring a fair and balanced contribution from all participants. In [25], the authors introduce a buffered FL approach for anomaly detection in drone networks named Agnostic Straggler Resilient (ASR_Fed), designed to handle straggling agents efficiently. The proposed method in [25] dynamically selects agents based on their accuracy and latency, ensuring that high-performing and responsive agents contribute early, while straggling agents are included later through a buffer-and-circumvent aggregation mechanism. Experimental results demonstrate that the proposed buffered methodology achieves

higher accuracy and lower communication overhead compared to traditional FL algorithms. However, by initially involving only fast agents and delaying the participation of stragglers, ASR-Fed limits exposure to the full data distribution, thereby reducing update diversity. This constraint negatively affects model generalization, resulting in poorer performance and slower convergence. While stragglers are eventually included, their late contribution diminishes the impact of rare or critical data during the early learning stages.

B. Privacy-enhancing FL

Though FL methodology offers significant privacy-preserving benefits, concerns about information leakage persist [10]. As outlined in [26], sensitive participant information in distributed FL can be compromised, even when only model update data is shared. Publicly available gradients can sometimes be used to reconstruct original training data. To address these vulnerabilities, Privacy Enhancing Technologies (PETs) such as DP, MPC and HE have been introduced to enhance FL's effectiveness by securing shared model updates.

In [27], an IoT anomaly detection FL model incorporating DP is proposed to enhance privacy. To this end, the method employs a Generative Adversarial Network (GAN) to generate synthetic local model parameters and adds controlled noise to the raw local models. The study in [28] examines the use of FL and HE to protect sensitive data shared during the training of CNN models. The study in [29] demonstrated that HE can provide comparable model accuracy to that of non-encrypted models while optimizing communication and computational efficiency. By employing an aggregated public key, the proposed method ensures the confidentiality of individual model updates without significantly affecting classification performance. Optimizing encrypted model updates is particularly useful for large-scale IoT deployments with limited resources. The Paillier Federated Multi-Layer Perceptron (PFMLP) proposed in [30] integrates HE with FL to ensure the security of gradient data during transmission while maintaining accuracy close to traditional methods. Authors in [31] provided a secure FL framework incorporating HE and Verifiable Computing (VC) in order to ensure confidentiality and integrity in model training, especially in cross-silo settings with few reliable agents. However, the increased complexity and resource demands of this approach make it less suitable for resource-constrained environments, despite maintaining model accuracy and providing strong privacy and integrity guarantees. The approach in [32] utilizes HE to enhance FL privacy, addressing common issues such as privacy breaches, communication overhead, and a lack of accountability. Meanwhile, the work in [33] applied FL based on CNN and GRU in order to improve the accuracy of ICS intrusion detection by learning both spatial and temporal features of network traffic data, while employing HE as a means of improving privacy. In [34], the authors presented a method called Partially Encrypted MPC for FL. The purpose of this approach is to reduce the high communication and computation costs associated with traditional MPC while preserving data privacy. To prevent sensitive information from being exposed during the aggregation of models, they selectively encrypt key parameter values or gradients. Despite improving efficiency, this approach may face complexities and challenges, particularly in dynamic environments with frequent model updates. Similarly, the work in [35] presents a method that strengthens FL security against indirect gradient leakage using MPC. In this method, a two-round model decomposition process ensures that the central server receives only a modified version of the model, preventing data reconstruction. Although MPC improves model accuracy and strengthens privacy safeguards, it also increases communication costs and computational requirements. In [36], the authors introduce a two-level privacy-preserving FL framework for attack detection in Consumer Internet of Things (CIoT), integrating Partially HE for secure model aggregation. While achieving high detection accuracy with reduced false positives, the framework lacks any straggler mitigation strategy, making it vulnerable to delays or failures when some CIoT devices are slow or intermittently connected.

Based on the investigated articles and our case which is IIoT environments, HE is preferred over DP and MPC due to its ability to protect model updates without compromising accuracy, which is vital for effective anomaly detection in IIoT environments. In fact, DP adds noise that can lower model accuracy. Meanwhile, MPC requires heavy communication and computation, making it unsuitable for quick-response scenarios in IIoT. In this work, the adopted HE scheme follows established approaches in the literature and is not intended to introduce a novel HE variant; rather, its role is to provide privacy preservation within DyHFL, enabling secure collaboration in IIoT without altering the core HE design.

Table I provides a comprehensive review of the studied articles, highlighting their gaps and advantages. It outlines the FL approaches, PETs, straggler issues, fairness considerations among agents with different speeds, and communication bottlenecks present in each study. This analysis offers a clear comparison of different methods and their associated challenges, emphasizing areas where further research is needed. Based on a thorough review of existing literature and the identified gaps in Table I, it has become apparent that studied methods lack a unified and efficient approach that provides a comprehensive solution to the straggler effect, communication bottlenecks, fairness, and privacy concerns. To fill this gap, this paper proposes a novel HE-based dynamic buffered FL framework for privacy-preserving anomaly detection in IIoT environments. The proposed framework aims to effectively balance the straggler effect and communication bottlenecks using a buffer and a novel dynamic agent selection strategy, while also enhancing privacy in environments with agents operating at different speeds. As a result, this approach, unlike existing solutions, optimally balances convergence speed, model performance, fairness, and communication costs.

III. DyHFL Framework

In this section, the system model, the attack model, and the proposed methodology are introduced. Table II summarizes the notations used in this study.

TABLE I: Summary of Research Gaps in Federated Learning Studies

Papers	Approach	Straggler mitigation strategy	PETs	Straggler	Fairness	Communication Bottleneck
[4], [5], [6], [7], [16], [17], [18]	SyncFL	×	×	×	N/A	N/A
[28], [29], [30], [31], [33], [36]	SyncFL	×	HE	×	N/A	N/A
[34], [35]	SyncFL	×	MPC	×	N/A	N/A
[19], [21]	AsyncFL	Clustering agents by computing power and Training data availability, communication latency	×	✓	N/A	×
[20]	AsyncFL	Greedy agent selection based on higher computing power and transmission delay	×	✓	×	✓
[22]	AsyncFL	lag-tolerant model distribution agent selection	×	✓	×	✓
[23]	AsyncFL	Agent selection based on dataset size and model accuracy	×	✓	×	✓
[24]	AsyncFL	Deep Q-Network based agent selection prioritizes high-accuracy, low-delay, and energy-efficient devices	×	✓	×	✓
[27]	AsyncFL	Basic asynchronous algorithm	DP	✓	N/A	×
[32]	AsyncFL	Basic asynchronous algorithm	HE	✓	N/A	×
[14]	Buffer-FL	Buffering the first K received updates per round	DP	✓	×	✓
[15]	Buffer-FL	Combination of buffering and agent selection based on training time in the first round	HE	✓	A/E	✓
[25]	Buffer-FL	Buffering based on prioritizing fast agents for early updates, delaying stragglers	×	✓	A/E	✓
DyHFL	Buffer-FL	Combination of buffering, agent selection, and sliding window based on training time, communication time, and data size	✓	✓	✓	✓

Note: ✓ – Addressed; × – Not addressed; N/A – Not an issue ; A/E – Addressed but not enough.

A. System Model

The system architecture in Fig. 1 is designed to ensure privacy while supporting FL across multiple industrial agents, each representing an owner of a CPS. It consists of the following steps:

1) **Key generation and Initialization:** At the beginning of the process, a trusted third party generates two cryptographic keys – one public and one private – for enabling secure communication between industrial agents and the FL aggregator server. The public key is used to encrypt data, while the private key is kept secret by each agent to decrypt data. The keys are then distributed to each industrial agent.
2) **Local Model Training:** Industrial agents use their local CPS data to train DL models.
3) **Encryption:** Once training is complete, the agent encrypts the model parameters (such as weights and biases) using the public key. In this way, sensitive model parameters are protected during transmission.
4) **Secure Aggregation of Models:** Each agent's encrypted model parameters are sent to the server for aggregation. The aggregation server computes the encrypted global model by securely aggregating the encrypted parameters received from each participating agent.
5) **Distribution of the Global Model:** The encrypted global model is sent back to all industrial agents once the aggregation process has been completed.
6) **Decryption:** The global model is then decrypted by each agent using its private key. The local model of the agent is subsequently updated using the decrypted model.

B. Attack Model

The threat model considered in this study addresses both cyber threats against industrial CPSs and adversarial attacks targeting the DyHFL framework. The study examines two main types of cyber threats facing the proposed model system:

TABLE II: List of Notations

Symbol	Description
T	Number of Rounds
F	Communication Frequency (local epochs)
N	Number of Agents
N_{sel}	Selected agents
M	Model Size
m	Initial parameters
M_{agg}	Encrypted global parameters
SW	Sliding window size
B	Size of Buffer
α and β	Floating-point numbers whose sum equals 1
t_st	Start training time
t_et	End training time
t_sc	Start communication time
t_ec	End communication time
Data_S	Local data size of the agent
$T_{\text{Train\_T}}$	Total Training time
$T_{\text{Com\_T}}$	Total communication time
Global_MT	Global Metric
ST_Thrsh	Short Term threshold
LT_Thrsh	Long Term threshold
c	A constant number
$ComCst$	Communication Cost
P_{key}	Public key
S_{key}	Private key
$\oplus$	HE-based addition
$\otimes$	HE-based multiplication

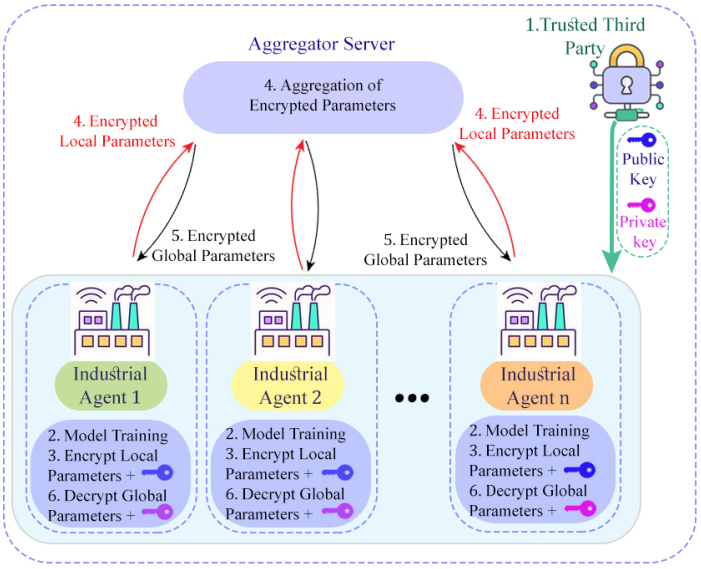

Fig. 1: High-level architecture of the DyHFL framework.

1) Cyber Threats Against Industrial CPSs: There are a number of threats to remote operations, including response injection attacks, command injection attacks, reconnaissance attacks, and denial of service (DoS) attacks. In response injection attacks, fake response messages are inserted into queries, whereas in command injection attacks, false commands are inserted into control systems. DoS attacks overload the target system, causing substantial disruptions to industrial processes, while reconnaissance attacks gather sensitive information about the CPSs.
2) Adversarial Threats Against the DyHFL Framework: This category focuses on two critical threats. In the first scenario, an "honest-but-curious" aggregation server may attempt to learn more than intended by inspecting the model parameters sent by agents. This could lead to the server inferring sensitive information about the agents' data or reverse-engineering the anomaly detection model, and compromising privacy in the FL process. The second concern involves external attacks or malicious eavesdroppers targeting communication links to intercept or alter the transmission of local parameters. This can expose sensitive data or manipulate updates, undermining the accuracy of the global model and threatening both the integrity and privacy of the DyHFL framework.

Algorithm 1: DyHFL-based Anomaly Detection

Input: T, N, c, P_{key}, S_{key}, α , β
Output: Desired performance (e.g., accuracy)

$m \leftarrow$ Initial_parameters()
$M_{agg} \leftarrow 0$ # Encrypted global parameters
$SW \leftarrow \frac{1}{c} \times T$
for $P \leftarrow 1$ **to** SW **do**
  for *each agent* $k \leftarrow 1$ **to** N **do**
    $t\_st \leftarrow$ Start_Time()
    $Local_{\text{m}}^{(k)} \leftarrow m$.train()
    $m_{\text{enc}}^{(k)} \leftarrow Encrypt(Local_{\text{m}}^{(k)}, P_{key})$
    $t\_et \leftarrow$ End_Time()
    $T_{\text{Train\_T}}^{(k)} \leftarrow t\_et - t\_st$
    $\text{Data\_S}^{(k)} \leftarrow Data\_Size()$
    $t\_sc \leftarrow$ Start_Time()
    send to server $(m_{\text{enc}}^{(k)})$
  # Encrypted aggregation on Server
  for *each* $j \in m_{enc}^{(k)}$ **do**
    $M_{\text{agg}} \leftarrow (j \oplus M_{\text{agg}}) \otimes [N^{-1}]$
  send to agents (M_{agg})
  for *each agent* $k \leftarrow 1$ **to** N **do**
    Receiving (M_{agg}) by agents
    $t\_ec \leftarrow$ End_Time()
    $T_{\text{Com\_T}}^{(k)} \leftarrow t\_ec - t\_sc$
    $m_{\text{dec}}^{(k)} \leftarrow Decrypt(M_{agg}, S_{key})$
    send to server $(T_{\text{Com\_T}}^{(k)}, T_{\text{Train\_T}}^{(k)}, \text{Data\_S}^{(k)})$
  # Calculating Final Threshold on Server
  Normalize $(T_{\text{Com\_T}}^{(k)}, T_{\text{Train\_T}}^{(k)}, \text{Data\_S}^{(k)})$
  $\text{Global\_MT}^{(k)} \leftarrow \alpha(T_{\text{Com\_T}}^{(k)} + T_{\text{Train\_T}}^{(k)}) + \beta(\text{Data\_S}^{(k)})$
  $ST\_Thrsh^{(P)} \leftarrow WAM\left(\text{Global}_{MT}^{(k)}\right)_{k=1}^{N}$
LT_Thrsh $\leftarrow \left(\text{EWA}\left((ST\_Thrsh^{(P)})_{P=1}^{SW}\right) + \text{MAX}\left((ST\_Thrsh^{(P)})_{P=1}^{SW}\right)\right)/2$
for *each* $i \in Global\_MT^{(k)}$ **do**
  if $i \leq LT\_Thrsh$ **then**
    $[S_m] \leftarrow i$
for $S \leftarrow SW + 1$ **to** T **do**
  for *each agent* $k \in [S_m]$ **do**
    $Local_{\text{m}}^{(k)} \leftarrow m$.train()
    $m_{\text{enc}}^{(k)} \leftarrow Encrypt(Local_{\text{m}}^{(k)}, P_{key})$

It is intended that by examining cyber threats, DyHFL will strengthen the security and resilience of industrial CPSs against a variety of malicious activities.

C. Proposed Methodology

The proposed HE-based FL framework employs two key strategies to enable efficient training of DL anomaly detection models across multiple industrial CPS owners: a **synchronous communication mode** and a **dynamic buffer-based agent selection mechanism**.

On one hand, the framework adopts the **synchronous communication mode**to mitigate the communication bottlenecks commonly associated with AsyncFL. In AsyncFL, updated models must be transmitted individually to each industrial agent, leading to increased network overhead and inconsistencies in model convergence. By contrast, the synchronous approach ensures that model updates are aggregated and distributed in a coordinated manner, significantly reducing network load and promoting more stable global model updates.

On the other hand, the framework introduces a **buffer-based agent selection mechanism** to reduce delays caused by slow agents, commonly known as the straggler effect in SyncFL. In synchronous FL, the server must wait for all selected agents to return their local model updates before aggregation. This can significantly hinder training progress. Proposed mechanism resolves this by introducing a buffer and a time threshold: the aggregation server collects model updates into a buffer and begins aggregation once the specified threshold time has elapsed, regardless of whether all agents have responded. This mechanism not only minimizes training delays but also mitigates bias introduced by uneven agent participation, leading to more balanced model performance. The dynamic buffer-based selection mechanism implemented by dividing the total number of training rounds *T* into two distinct phases: **Preliminary Rounds**, used to estimate the time threshold, and **Subsequent Rounds**, where agent selection is guided by this threshold.

1) **Phase One: Preliminary Rounds (Rounds 1 to SW)** The objective of the preliminary rounds is to estimate a time threshold that will guide agent selection in subsequent training rounds. This is achieved using a sliding window mechanism, designed to monitor agent-specific performance metrics over time and ensure fairness and adaptability. Each agent records the following performance metrics that will be used in computing the time threshold:
   - **Data size:** the volume of local data available for training.
   - **Training Time:** the duration taken to train the local model.
   - **Communication Time:** the time required to transmit the encrypted model to the server and receive the updated global model.

   Below is a detailed Step-by-Step Workflow:

   a) **sliding window size calculation:** In the first step, *the sliding window size (SW)* is set equal to the number of *preliminary rounds (P)*, which is computed as:

   $$SW = P = \frac{T}{c} \tag{1}$$

   where T is the total number of training rounds, and c is a constant that controls the proportion of rounds allocated to the threshold estimation phase (line 3 of Algorithm 1).

   b) **Model Training:** Once the public and private keys are generated, agents obtain parameter values from the aggregator server. The agents then train their DL models using their respective data in order to detect anomalies (line 7 of Algorithm 1).

   c) **Monitoring Metrics:** During the training process, the amount of time that each agent spends on training along with the size of data are measured (lines 10 and 11 of Algorithm 1).

   d) **Model Encryption and Transmission:** Once the local model is trained, the parameters are encrypted using the public key (line 8 of Algorithm 1). Following training and encryption, the encrypted parameters are sent to the aggregator server (line 13 of Algorithm 1). This is the point at which the calculation of communication time begins until the agents receive the updated encrypted global model (line 12 of Algorithm 1).

   e) **Aggregation:** Aggregator server collects encrypted local parameters from agents and compute encrypted global parameters (line 16 of Algorithm 1). These encrypted global parameters are then sent back to the agents (line 17 of Algorithm 1).

   f) **Updating and Decryption:** Agents receive the encrypted global parameters, which marks the end of the calculation of communication times (lines 19 and 20 of Algorithm 1). Afterward, using private keys, the agents decrypt the encrypted global parameters and update the local model parameters accordingly (line 22 of Algorithm 1).

   g) **Short-Term Threshold (ST_Thrsh):** This step involves calculating the *Short-Term Threshold (ST_Thrsh)* using agent-specific performance metrics. The purpose of *ST_Thrsh* is to ensure fair participation by adjusting selection bias between fast and slow agents. To calculate the *ST_Thrsh*:
      - Each agent's performance metrics values (data size, training time, and communication time) are sent to the server (as described in line 23 of Algorithm 1).
      - The performance metrics values are normalized by the server using the *Min-Max scaler*, which rescales the values to the range of [0, 1] (line 25 of Algorithm 1). The Min-Max scaler is defined as follows:

      $$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2}$$

      where X represents the original value, $X_{\min}$ is the minimum value within its respective list (training time, communication time, or data size), and $X_{\max}$ is the maximum value within the same list.
      Normalization is necessary to balance the impact of different parameters, as they exist on different scales. Without normalization, larger numerical values (e.g., data size) could dominate the global metrics, leading to biased agent selection.
      - The normalized metrics are then summed to com-

pute the agent's global metric ***(Global_MT)*** (line 26 of Algorithm 1), as follows:

$$\text{Global\_MT} = \alpha \cdot (T_{\text{Train\_T}} + T_{\text{Com\_T}}) + \beta \cdot \text{Data\_S} \quad (3)$$

where α and β are floating-point numbers whose sum equals one $(\alpha + \beta = 1)$, serving as weighting factors and normalization coefficients to balance the contributions of different components in the computation of the *Global_MT*. Their primary purpose is to ensure a controlled trade-off between the communication and training time components $(T_{\text{Train\_T}+T_{\text{Com\_T}}})$ and the dataset size component (*Data_S*). This prevents any single component from disproportionately influencing the *Global_MT*. By adjusting α and β, the model can emphasize either computation time or dataset size depending on the specific requirements of the system, ensuring an adaptive and efficient evaluation of the agent's performance.

- The computed *Global_MT* values are fed into the *WAM* function to derive the *ST_Thrsh*. *WAM* assigns lower weights to faster agents and higher weights to slower ones, thereby promoting fair participation and preventing consistent exclusion from training. These weights are directly determined by each agent's normalized global metrics (line 27 of Algorithm 1 and Algorithm 2).

h) **Long-Term Threshold (LT_Thrsh):** In the final step of the preliminary phase, the *Long-Term Threshold (LT_Thrsh)*—used as the time threshold for buffer-based agent selection—is computed using an *EWA* applied to the sequence of *ST_Thrsh* values (line 28 of Algorithm 1 and Algorithm 3). *EWA* emphasizes recent *ST_Thrsh* values have a greater impact on the *LT_Thrsh*, enabling the threshold to adapt dynamically while smoothing out transient fluctuations. The resulting *LT_Thrsh* serves as the final decision boundary for agent selection in subsequent training rounds.

2) **Phase Two: Subsequent Rounds (Rounds SW+1 to T)** In this phase, the calculated time threshold (LT_Thrsh) from preliminary round is employed by the aggregation server to govern agent selection and aggregation behavior (lines 29 to 35 of Algorithm 1). This dynamic approach ensures that training progresses efficiently without being hindered by stragglers, communication bottlenecks, and still preserving fairness and representation across the agent population.
Algorithms 1, 2, 3 summarize the detailed training procedure.

Steps ***b*** through ***g*** of phase one are repeated until the number of preliminary rounds (P) reaches T/c, while the subsequent rounds of phase two continue until convergence. For instance, if T=100 and c=5, then the number of preliminary rounds will be P=20, which also defines the size of the sliding window (SW). The remaining 80 rounds constitute the subsequent phase of training. In this setup, the first 20 rounds

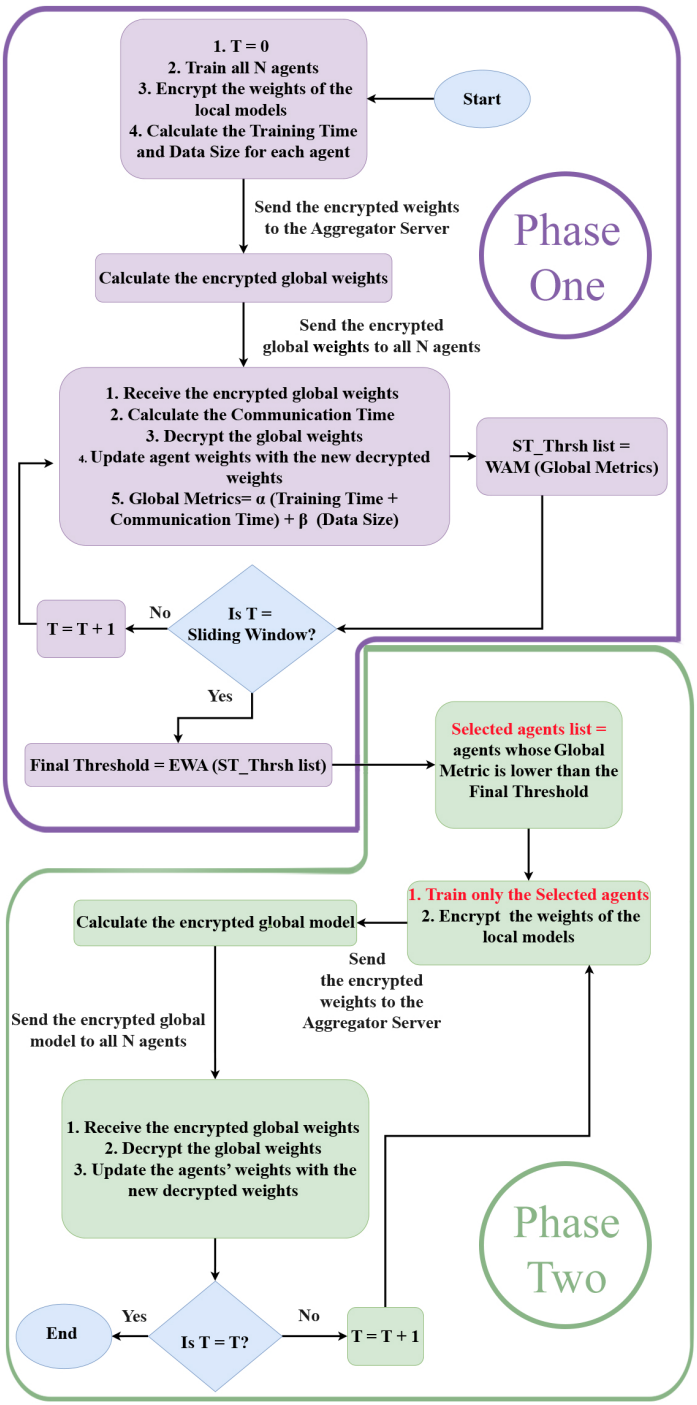

Fig. 2: DyHFL framework Flowchart.

Algorithm 2: Weighted_Average_Metrics (WAM)

Input: Global metrics for each Agent ← Global_MT$^1, \ldots,$ Global_MT^N
Output: Short-term threshold for each preliminary round
Sort $Global\_MT^1, Global\_MT^2, ..., Global\_MT^N$ in descending order
for *each* $i \leftarrow 1$ *to* n **do**
  $weight^{(i)} \leftarrow \frac{1}{Global\_MT^i}$
Reverse weight = $[W_1, W_2, \ldots, W_n]$
weight ← $[W_N, W_{N-1}, \ldots, W_1]$
W_Avg ← $\frac{\sum_{j=1}^{n} \text{Global\_MT}^{(j)} \cdot \text{weight}^{(j)}}{\sum_{j=1}^{n} \text{weight}^{(j)}}$

are allocated to the preliminary phase, during which the system monitors agent behavior and calculates the time threshold. The following 80 rounds proceed with selected agents based on the computed threshold, continuing until convergence.

Algorithm 3: Exponential_Weighted_Average (EWA)

Input: Short-term thresholds $\leftarrow ST\_Thrsh^1, ST\_Thrsh^2, \ldots, ST\_Thrsh^P$
Output: Final Threshold for short-term thresholds
for *each* $i \leftarrow 1$ *to* n **do**
  $\text{E\_Weight}^{(i)} \leftarrow \frac{i*(i+1)}{2}$
Thereshold $\leftarrow \frac{\sum_{j=1}^{n} \text{E\_weight}^{(j)} * ST\_Thrsh^{(j)}}{\sum_{j=1}^{n} \text{E\_weight}^{(j)}}$

IV. Security and Communication Complexity Analysis

This section provides a security and communication complexity analysis of the DyHFL framework. It demonstrates how HE safeguards model updates from both honest-but-curious servers and potential external eavesdroppers, while also providing a detailed comparison of the communication cost and computational complexity across different FL methods.

A. Protection Against an Honest-But-Curious Server

In DyHFL, the central server is assumed to be *honest-but-curious*—it correctly follows the protocol but tries to learn private information from the data it receives. To prevent this, each agent encrypts its local model updates using a public key before sending them to the server.

How It Works: Let Δw_i be the model update from agent i, and $P_{\text{key}}(\cdot)$ be the encryption function using a public key. Each agent sends:

$$P_{\text{key}}(\Delta w_i) \tag{4}$$

The server receives encrypted updates from all N agents:

$$P_{\text{key}}(\Delta w_1),\ P_{\text{key}}(\Delta w_2),\ \ldots,\ P_{\text{key}}(\Delta w_N) \tag{5}$$

It then performs **homomorphic aggregation**, which gives an encrypted sum of the model updates:

$$P_{\text{key}}(\Delta W) = P_{\text{key}}\left(\sum_{i=1}^{N} \Delta w_i\right) \tag{6}$$

The server cannot decrypt the encrypted model updates, as it does not hold the private key. The encrypted aggregated model $P_{\text{key}}(\Delta W)$ is then sent back to the agents, who can decrypt it using the private key S_{key}, which is exclusively held by agents:

$$\Delta W = S_{\text{key}}\big(P_{\text{key}}(\Delta W)\big) \tag{7}$$

Due to the **semantic security** of HE [37], which guarantees that without the private key the server cannot distinguish between any two encrypted values, the server cannot infer any information about the individual encrypted model updates, even though it performs the aggregation. This ensures the privacy of individual agents, even in the presence of an honest-but-curious server.

B. Security Against Eavesdropping During Transmission

HE also protects model updates during communication. This prevents external attackers from learning private information by intercepting data.

How It Works: Consider an external attacker $\mathcal{A}$ attempting to eavesdrop on the communication between agents and the server by intercepting model updates during transmission. Suppose $\mathcal{A}$ captures a ciphertext $P_{\text{key}}(\Delta w)$. Even if provided with two candidate plaintext model updates, Δw_0 and Δw_1, and their corresponding ciphertexts $P_{\text{key}}(\Delta w_0)$ and $P_{\text{key}}(\Delta w_1)$, the adversary cannot determine which ciphertext corresponds to which plaintext with non-negligible probability.

HE is **probabilistic**, meaning each encryption of the same plaintext gives a different ciphertext. Its security relies on the decisional composite residuosity assumption, a hard problem in number theory. This assumption guarantees that, without the private key, an adversary cannot distinguish between the ciphertexts of two different plaintexts.

C. Communication Cost Complexity

Communication cost refers to the total size of messages exchanged, specifically models transmitted between agents and the aggregator server, measured in megabytes (MB). The communication cost (**ComCst**) for various FL methods is calculated as follows:

$$\text{FedBuff ComCst} = M \times T \times B \tag{8}$$

$$\text{SyncFL ComCst} = M \times T \times N \tag{9}$$

$$\text{BFL ComCst} = (M \times 1 \times N) + (M \times (T-1) \times N_{\text{sel}}) \tag{10}$$

$$\text{ASR\_Fed ComCst} = M \times (T \times \text{Buffs} + (T - \text{CirT}) \times \text{Cirs}) \tag{11}$$

$$\text{DyHFL ComCst} = (M \times SW \times N) + (M \times (T - SW) \times N_{\text{sel}}) \tag{12}$$

$$\text{AsynFL ComCst} = M \times T \times N \times F \tag{13}$$

Where:

- M: Model size
- T: Total rounds
- N: Number of agents
- B: Buffer size
- N_{sel}: Number of selected agents
- SW: Sliding window size
- Buffs: buffer agent size in ASR_Fed
- Cirs: circumvent agent size
- CirT: circumvent threshold size
- F: Communication frequency in AsyncFL which refers to the average number of communication events—both uploads and downloads—between each agent and the server per training round.

Table III compares the time complexity $\mathcal{O}$ (Big O) and the best-case performance Ω (Big Omega) for various FL algorithms. Regarding **DyHFL** and **BFL**, the best-case complexity occurs when the number of selected agents k is minimal, resulting in a complexity of $\Omega(\text{T} \times k)$.

TABLE III: Computational Complexity of FL Baselines

Algorithm	Big O	Big Omega
DyHFL	$O(T \times n)$	$\Omega(T \times k)$
BFL	$O(T \times n)$	$\Omega(T \times k)$
FedBuff	$O(T \times n)$	$\Omega(T \times B)$
SyncFL	$O(T \times n)$	$\Omega(T \times n)$
AsyncFL	$O(T \times n \times F)$	$\Omega(T \times n)$
ASR_Fed	$O(T \times$ Cirs$)$	$\Omega(T \times$ Buffs$)$

In **Fedbuff**, the complexity is minimized when the buffer size is minimal, giving a complexity of $\Omega(\mathrm{T} \times B)$. For **ASR_Fed**, the complexity is minimized when the buffer agent size (Buffs) is maximized, resulting in a lower bound complexity of $\Omega(T \times Buffs)$. Conversely, the complexity is maximized when the circumvent agent size (Cirs) is maximized, leading to an upper bound complexity of $\mathcal{O}(T \times Cirs)$. The **SyncFL** method's complexity remains relatively stable across different scenarios, with a best-case of $\Omega(\mathrm{T} \times n)$. Lastly, in **AsyncFL**, the best case occurs when the number of communication frequencies is minimized, typically to one, leading to a complexity of $\Omega(\mathrm{T} \times n)$.

V. Experiment Setting and Evaluation

A. Experimental Settings

The proposed FL framework was evaluated using three datasets with different characteristics in terms of number of classes, the size of the data, and the distribution of the samples, namely Gas_Pipeline dataset [38], the WUSTL_IIoT dataset [39], and the Edge_IIoT dataset [40], as described in [15] to achieve a more comprehensive and reliable evaluation. The Gas_Pipeline dataset contains industrial data with specific features, while the WUSTL_IIoT and Edge_IIoT datasets covers IoT-related data. Using these three diverse datasets allows us to demonstrate that DyHFL is not dependent on a specific type of data and can maintain strong, and reliable performance across different data volumes, distributions, and structural complexities. This approach effectively demonstrates the generalization of the proposed method and prevents concerns about dataset-specific overfitting. PyTorch was used to implement the proposed FL framework, while Paillier was used to implement HE. For demonstrating the effectiveness of the proposed model in scenarios involving heterogeneous data, non-identical datasets are used. The non-identical data distribution allows us to identify which agents are fast and which ones are slow, allowing us to simulate performance differences more accurately. Accordingly, FedArtML is used for the generation of non-identical datasets (Dirichlet and no-label-skew). ***Dirichlet Method*** utilizes the Dirichlet distribution to create unequal distributions, resulting in agents receiving different proportions of data and varying percentages of labels. ***No-Label-Skew Method*** ensures that each agent receives data with the same labels, thereby the distribution of labels remains consistent. Due to this, agents with larger data may have longer global times, known as slow agents, whereas agents with smaller data may have shorter global times, known as fast agents.

A pre-processing operation was conducted in order to clean the input data and improve its accuracy. Gas_Pipeline dataset features with only one value are removed, reducing the number of features from 27 to 18. Datasets are split into three parts: 80% for training, 10% for validation, and 10% for testing. Min-Max scaling is used to normalize the dataset's features.

The multilayer perceptron (MLP)-based IDS is implemented after performing data pre-processing. MLP model with three fully connected layers was used to analyze the Gas_Pipeline, WUSTL_IIoT, and Edge_IIoT datasets. In the case of the Gas_Pipeline dataset, the model's first layer transforms the 18-dimensional input into a 54-dimensional space. The second layer reduces this to a 20-dimensional space, and the final layer maps it to eight classes. In the case of WUSTL_IIoT, the first layer transforms the 41-dimensional input into a 9-dimensional space, the second layer uses the same 9-dimensional space, and the final layer maps it to five classes. Regarding the Edge_IIoT dataset, the model's first layer transforms the 15-dimensional input into a 64-dimensional space. The second layer reduces this to a 32-dimensional space, and the final layer maps it to fifteen classes. The model structure was determined based on the nature of the datasets (dimensionality and class distribution), experiments with different layer sizes, and the structure presented in [15].

Stochastic Gradient Descent (SGD) was chosen as the optimization algorithm, with a batch size of 64 for the Gas_Pipeline dataset and 1000 for the WUSTL_IIoT and Edge_IIoT datasets. The learning rate was set to 0.01, and the momentum was set to 0.8. Based on best practices for the datasets, these settings were selected to ensure stable and efficient convergence. To optimize further, future work could explore different momentum settings or adaptive learning rates.

The proposed DyHFL framework uses a semi-synchronous buffer-based strategy. Therefore, to highlight its advantages, it is compared with five FL methods: SyncFL [12] and AsyncFL [13] as traditional baselines, and with FedBuff [14], ASR_Fed [25], and BFL [15] as state-of-the-art buffered or selective aggregation methods. This comparison illustrates improvements over both fully synchronous/asynchronous schemes and existing buffered approaches. Detailed descriptions of **SyncFL, AsyncFL, and BFL** implementations can be found in the referenced sources. A buffer size equal to 75% of total agents is recommended for **FedBuff** implementation, based on the information provided in Table V. The details of this choice are described in the Evaluation part of Section V.

The **ASR_Fed** implementation classifies agents into buffer agents and circumvent agents. Buffer agents include those whose accuracy and training time exceed a predefined threshold, while circumvent agents consist of those that do not meet this threshold. The accuracy and training time threshold is dynamically determined based on the average accuracy and training time of all participating agents. Agents categorized as buffer agents are considered fast agents, whereas those in the circumvent list are classified as slow agents. During the initial aggregation rounds, only updates from fast agents are incorporated. Slow agents participate in the aggregation process only when the total number of rounds reaches a predefined value, referred to as the circumvent threshold. This threshold is computed using the following formula based on reference [25]:

$$\text{Circumvent Threshold} = \text{int}\left(\left\lceil \frac{\sum(\text{circumvent agent delays})}{\text{buffer agent count}} \right\rceil\right) \quad (14)$$

Once the total number of rounds equals the circumvent threshold, slow agents also contribute to the aggregation process, ensuring a balanced and adaptive participation mechanism.

Four key metrics are evaluated in order to highlight the advantages of the proposed approach: **1. Agent Selection Fairness, 2. Convergence Speed, 3. Communication Costs, and 4. Model Performance**.

1) As mentioned earlier, one of the goals of the proposed method is to balance the participation between fast and slow (straggler) agents. **Agent selection fairness** is defined as assessing the fairness of the proposed method in terms of the participation of both fast and straggler agents across the entire agent population. To ensure that the proposed approach guarantees fair participation and performance for both fast and straggler agents, we consider two metrics: Straggler Rate Selection (SRS) and Fast Rate Selection (FRS). The formulas for these metrics are as follows:

$$\text{SRS} = \frac{\text{Number of Selected Straggler Agents}}{\text{All of Straggler Agents}} \quad (15)$$

$$\text{FRS} = \frac{\text{Number of Selected Fast Agents}}{\text{All of Fast Agents}} \quad (16)$$

SRS measures the proportion of selected straggler agents compared to the total number of straggler agents available. The FRS represents the proportion of selected fast agents relative to all fast agents in the system.

2) **Convergence speed** is a measure of the number of rounds required to achieve a specified level of accuracy (i.e., target accuracy). When an approach requires fewer rounds to achieve the target accuracy, this indicates a faster rate of convergence, and vice versa. The purpose of this metric is to assess the robustness of FL methods against stragglers as well as their effectiveness in communication efficiency. This study compares different methods using 20 agents as a baseline, and a **target accuracy of 94.6 % for the Gas_Pipeline dataset, 99.8 % for the WUSTL_IIoT dataset and 98.5 % for the Edge_IIoT dataset**. In both cases, the target accuracy is set slightly above the baseline accuracy that the non-federated method achieves with centralized data, ensuring that the FL method performs as well, if not better, with distributed data.
3) **Communication Cost**, as defined in Section IV.C, denotes the total size of data exchanged—specifically the model parameters transmitted between agents and the aggregator server—measured in megabytes (MB).
4) **Model Performance** is evaluated using three key classification metrics: Accuracy, Precision, and the F_1 Score. These metrics are defined as follows:
   - **Accuracy** reflects the overall correctness of the model, calculated as the proportion of correctly predicted instances (both positive and negative) out of all predictions:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}} \quad (17)$$

   - **Precision** quantifies the proportion of correctly predicted positive instances out of all predicted positives. It measures the reliability of the model's positive predictions:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (18)$$

   - **F_1 Score** provides a single performance metric by taking the harmonic mean of Precision and Recall. Recall measures the model's ability to identify all actual positive instances and is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (19)$$

   The F_1 Score is then calculated as:

$$F_1 \text{ Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (20)$$

   The F_1 Score is particularly useful in imbalanced classification problems, where it provides a more informative evaluation than Accuracy by considering both false positives and false negatives.

B. Evaluation

The results were averaged from four independent runs for the metrics described earlier (i.e., Agent Selection Fairness, Convergence Speed, Communication Costs, and Model Performance metrics).

The first step in the evaluation process involves determining the optimal sliding window size and the corresponding values of α and β, which are essential for computing the DyHFL method results. As shown in Table IV, sliding window sizes of $1/10$, $1/5$, and $1/2$ of the total training rounds were evaluated alongside various combinations of α and β values, specifically (0.3, 0.7), (0.7, 0.3), and (0.5, 0.5). The evaluation was performed using three datasets — Gas_Pipeline, WUSTL_IIoT and Edge_IIoT — under three distinct data distribution scenarios: Identical, Dirichlet (non-identical), and No-Label-Skew (non-identical), all based on convergence speed with 100 agents. Among the tested configurations, the sliding window size of $1/10$ with $\alpha = 0.7$ and $\beta = 0.3$ consistently yielded slightly better convergence performance compared to larger window sizes. Consequently, this configuration was selected as the most balanced and effective choice for use throughout this study.

The second step is to determine the appropriate buffer size for achieving appropriate FedBuff results. Buffer sizes of 4, 10, and 15 were tested, representing 25%, 50%, and 75% of the 20 participant agents. Using the previously defined target accuracy, the best buffer size was determined by examining the convergence speed. The results in Table V indicate that FedBuff faces challenges when dealing with buffer sizes of four and ten. These two buffer sizes (i.e., 4 and 10) perform poorly because they do not allow sufficient diversity in updates before aggregation. The situation is particularly problematic in non-identical data settings, where data distribution varies among agents. As an example, in the Dirichlet setting, the model failed to converge with these buffer sizes for both

TABLE IV: Sliding-Window Size Selection in DyHFL Based on Convergence Speed with 100 Agents

Dataset	SW Size	α / β	Dirichlet	No-Label-Skew	Identical
Gas_Pipeline	1/2	0.3 / 0.7	250	58	70
		0.7 / 0.3	296	58	70
		0.5 / 0.5	296	56	70
	1/5	0.3 / 0.7	250	55	62
		0.7 / 0.3	250	60	62
		0.5 / 0.5	250	78	62
	1/10	0.3 / 0.7	235	57	67
		0.7 / 0.3	232	55	67
		0.5 / 0.5	250	56	67
WUSTL_IIoT	1/2	0.3 / 0.7	10	9	9
		0.7 / 0.3	10	8	9
		0.5 / 0.5	8	7	9
	1/5	0.3 / 0.7	8	7	7
		0.7 / 0.3	9	9	7
		0.5 / 0.5	10	6	7
	1/10	0.3 / 0.7	9	9	8
		0.7 / 0.3	8	6	8
		0.5 / 0.5	8	9	8
Edge_IIoT	1/2	0.3 / 0.7	150	25	23
		0.7 / 0.3	150	25	23
		0.5 / 0.5	150	21	24
	1/5	0.3 / 0.7	125	23	24
		0.7 / 0.3	120	20	22
		0.5 / 0.5	122	21	22
	1/10	0.3 / 0.7	100	21	25
		0.7 / 0.3	96	20	20
		0.5 / 0.5	98	22	22

datasets. When there are not sufficient updates, the model may become overfit or biased toward certain agents, which negatively affects its ability to generalize. Alternatively, large buffer sizes, such as 15, have better results since there are enough updates to aggregate. Therefore, an ideal buffer size for FedBuff is 15 (75% of the participant agents) because this maintains sufficient diversity resulting in faster convergence and robust model performance.

TABLE V: Buffer Size Selection in FedBuff Based on Convergence Speed

Dataset	SW Size	Identical	Dirichlet	No-Label-Skew
Gas_Pipeline	4	80	not converged	284
	10	59	not converged	105
	15	63	400	61
WUSTL_IIoT	4	87	not converged	62
	10	95	not converged	125
	15	104	590	60
Edge_IIoT	4	20	not converged	90
	10	15	not converged	85
	15	10	500	60

The last step is to conduct the experiments and assess the results obtained using the defined evaluation metrics.

*1) **Agent Selection Fairness**:* To evaluate the performance of the different FL methods in terms of the defined SRS and FRS metrics, we consider various percentages of straggler agents. Specifically, we analyze scenarios where straggler agents constitute 10%, 20%, up to 90% of the total agent population, across different total agent counts (i.e., 10, 20, ..., 50 agents). Through 20 rounds, the delay times of fast agents were simulated using random integer values between 1 and 5 and those of stragglers were simulated using random integer values between 6 and 10.

According to the results presented in Tables VI and VII, DyHFL outperforms the BFL method introduced in [15] by using a sliding window mechanism, making the agent selection process more dynamic and responsive to changes in agent performance over time. This adaptive approach better handles variations in agent performance, especially in environments where agent availability and performance can fluctuate. In terms of SRS, the proposed method achieves an average of 56.44%, nearly double that achieved by the BFL method (i.e., 29.62%). This substantial improvement indicates that DyHFL is far more effective in managing stragglers across various agent groups. For example, in Table VII, a gradual increase in SRS as the straggler agent percentage rises can be observed. For lower percentages of straggler agents, SRS remains at 0%, but as the percentage of stragglers increases, SRS begins to rise, indicating a more adaptive selection process. In contrast, SRS for the BFL method remains at 0% for up to 50% straggler agents, suggesting that its initial round-based selection is less adaptive and might not effectively handle new stragglers appearing in later rounds. Both methods maintain high FRS, but DyHFL does so more effectively across a wider range of agent counts and straggler percentages. It achieves a 100% rate across all agent configurations, while the BFL method averages 97.91%. Although this difference might seem small, DyHFL guarantees FRS consistently, making it more reliable. This suggests that the new method's dynamic selection process helps consistently in identifying and leveraging fast agents for efficient training.

In both SRS and FRS, DyHFL outperforms the BFL method by incorporating a sliding window mechanism that adapts to changes in the agent performance. In environments with varying agent performance, this approach allows for better management of stragglers and more consistent utilization of fast agents.

*2) **Convergence Speed**:* Fig. 3 illustrates the convergence speed for different models on both identical and non-identical datasets for 100 agents.

According to Fig. 3, **DyHFL** consistently outperforms the other five methods —AsyncFL, FedBuff, SyncFL, ASR_Fed, and BFL— on both identically and non-identically distributed datasets, demonstrating its robustness to heterogeneous data. This robustness is due to its balanced approach of selecting both fast and slow agents, ensuring effective learning and balanced participation. For example, in the Gas_Pipeline dataset, DyHFL converges 11.9 times faster than AsyncFL, 5.6 times faster than FedBuff, 5.3 times faster than SyncFL, and 5.2 times faster than BFL on identical datasets. The performance advantage becomes more pronounced on non-identical datasets, with DyHFL converging 16.3 times faster than AsyncFL, 14.18 times faster than FedBuff,10.9 times faster than SyncFL, and 9 times faster than BFL on no-label-skew data, and 3.4, 3.4, 3, and 2.8 times faster on Dirichlet non-identical data, respectively. Similarly, with the WUSTL_IIoT dataset, DyHFL surpasses AsyncFL by 112 times, FedBuff by 62.5 times, 6.8 times faster than SyncFL, and BFL by 3.2 times on identical datasets. The advantages are even greater for non-identical data, with DyHFL converging 158 times faster than AsyncFL, 60 times faster than FedBuff, 53 times

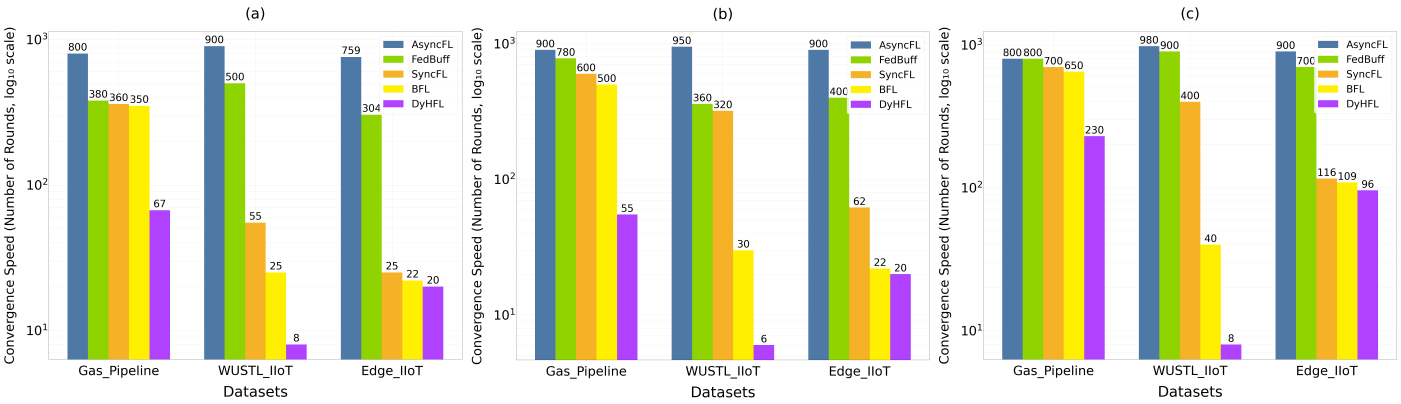

Fig. 3: Convergence Speed Comparison Across FL Methods based on 100 Agents
: (a) Identical Datasets, (b) non-identical datasets with No-Label-Skew distribution, (c) non-identical datasets with Dirichlet distribution. The y-axis indicates the number of FL rounds required to reach the target accuracy, plotted on a logarithmic scale.

TABLE VI: Straggler Agent Percentage in BFL

Straggler Agents Percentage	10 Agents		20 Agents		30 Agents		40 Agents		50 Agents	
	SRS	FRS	SRS	FRS	SRS	FRS	SRS	FRS	SRS	FRS
10%	0%	66.66%	0%	72.22%	0%	85%	0%	88.87%	0%	93.33%
20%	0%	100%	0%	100%	0%	100%	0%	100%	0%	100%
30%	0%	100%	16.66%	100%	22.22%	100%	25%	100%	26.66%	100%
40%	25%	100%	25%	100%	25%	100%	31%	100%	40%	100%
50%	20%	100%	30%	100%	33.33%	100%	35%	100%	44%	100%
60%	16.66%	100%	41.66%	100%	44.44%	100%	45.83%	100%	46.66%	100%
70%	28.57%	100%	42.85%	100%	47.61%	100%	57.14%	100%	62.85%	100%
80%	37.5%	100%	50%	100%	54.16%	100%	59.37%	100%	70%	100%
90%	44.44%	100%	50%	100%	66.66%	100%	69.44%	100%	71.11%	100%
Average	**19.13%**	**96.29%**	**26.6%**	**96.9%**	**30.13%**	**98.33%**	**33.08%**	**98.76%**	**37.18%**	**99.25%**

TABLE VII: Straggler Agent Percentage in DyHFL

Straggler Agents Percentage	10 Agents		20 Agents		30 Agents		40 Agents		50 Agents	
	SRS	FRS	SRS	FRS	SRS	FRS	SRS	FRS	SRS	FRS
10%	0%	100%	0%	100%	0%	100%	0%	100%	0%	100%
20%	0%	100%	25%	100%	33.33%	100%	25%	100%	50%	100%
30%	33%	100%	50%	100%	55.55%	100%	50%	100%	60%	100%
40%	50%	100%	62.5%	100%	66.66%	100%	68.75%	100%	65%	100%
50%	60%	100%	70%	100%	73.33%	100%	75%	100%	72%	100%
60%	66.66%	100%	75%	100%	77.77%	100%	79.16%	100%	76.66%	100%
70%	71.42%	100%	78.57%	100%	80.95%	100%	85.5%	100%	82.85%	100%
80%	75%	100%	81.25%	100%	83.33%	100%	85%	100%	74%	100%
90%	77.77%	100%	77.77%	100%	77.77%	100%	77.77%	100%	77.77%	100%
Average	**47.09%**	**100%**	**53.67%**	**100%**	**60.61%**	**100%**	**60.7%**	**100%**	**62.13%**	**100%**

faster than SyncFL, and 5 times faster than BFL on no-label-skew data, and 122, 112, 50, and 5 times faster on Dirichlet non-identical data, respectively. Regarding Edge_IIoT dataset, DyHFL surpasses AsyncFL by 37.95 times, FedBuff by 15.2 times, 1.25 times faster than SyncFL, and BFL by 1.1 times on identical datasets. The advantages are even greater for non-identical data, with DyHFL converging 45 times faster than AsyncFL, 20 times faster than FedBuff, 3.1 times faster than SyncFL, and 1.1 times faster than BFL on no-label-skew data, and 9.37, 7.2, 1.20, and 1.13 times faster on Dirichlet non-identical data, respectively.

This clearly demonstrates the DyHFL's efficiency and adaptability across diverse scenarios.

The differences in **DyHFL** convergence results between the three datasets can be attributed to their intrinsic properties. The Gas_Pipeline dataset is smaller in overall size. As a result, under No-label-skew conditions, due to the limited number of samples, agents receive very homogeneous data, making learning more difficult and convergence slower. Under the Dirichlet distribution, the diversity of the data mitigates some of these effects, leading to relatively better performance. On the other hand, the WUSTL_IIoT and Edge_IIoT datasets have a much larger volume of data. Consequently, even under challenging scenarios such as No-label-skew, agents still receive sufficiently diverse and abundant data, resulting in more stable model performance and less variation between different data distributions.

AsyncFL shows the slowest convergence overall, requiring the most rounds, particularly for non-identical datasets. Its asynchronous nature causes each agent to update independently, which leads to delays and communication bottlenecks. These findings suggest that methods relying on unoptimized communication between the server and agents are less efficient, especially when handling non-identical data. **FedBuff** requires significantly more rounds to converge on non-identical datasets than on identical ones, indicating a struggle to manage

TABLE VIII: Two-Tailed t-Test p-Values Comparing DyHFL with Other FL Baselines on Communication Cost

Dataset	Agents	BFL	Fedbuff	ASR_Fed	Async	Sync
Gas_Pipeline	20	1.05×10^{-5}	8.43×10^{-5}	1.38×10^{-5}	2.88×10^{-9}	4.97×10^{-6}
	40	2.81×10^{-6}	7.15×10^{-7}	8.81×10^{-10}	3.47×10^{-11}	5.24×10^{-8}
	80	9.49×10^{-5}	9.35×10^{-5}	3.30×10^{-7}	9.35×10^{-11}	2.90×10^{-7}
	100	1.25×10^{-2}	3.18×10^{-3}	2.17×10^{-3}	7.27×10^{-9}	2.58×10^{-5}
WUSTL_IIoT	20	7.11×10^{-4}	6.92×10^{-5}	1.17×10^{-5}	1.93×10^{-9}	3.59×10^{-6}
	40	1.81×10^{-3}	2.99×10^{-4}	1.25×10^{-4}	1.41×10^{-8}	2.16×10^{-5}
	80	3.09×10^{-2}	8.33×10^{-4}	4.23×10^{-5}	1.14×10^{-9}	4.55×10^{-6}
	100	4.55×10^{-2}	6.46×10^{-6}	2.17×10^{-7}	2.49×10^{-11}	8.10×10^{-8}
Edge_IIoT	20	3.11×10^{-2}	2.13×10^{-7}	3.47×10^{-9}	1.62×10^{-10}	1.72×10^{-7}
	40	1.04×10^{-4}	3.04×10^{-8}	4.74×10^{-7}	1.27×10^{-11}	1.41×10^{-8}
	80	8.40×10^{-5}	2.01×10^{-9}	1.44×10^{-15}	6.09×10^{-10}	1.25×10^{-9}
	100	3.02×10^{-2}	2.80×10^{-8}	2.54×10^{-8}	1.82×10^{-11}	2.27×10^{-8}

TABLE IX: Two-Tailed t-Test p-Values Comparing DyHFL with Other FL Baselines on Convergence Speed

Dataset	Data Type	Async	Sync	Fedbuff	BFL
Gas_Pipeline	Identical	3.49×10^{-9}	9.28×10^{-6}	2.36×10^{-10}	6.96×10^{-10}
	No-Label-Skew	2.49×10^{-10}	3.084×10^{-4}	2.28×10^{-7}	2.55×10^{-8}
	Dirichlet	3.19×10^{-9}	8.68×10^{-7}	4.64×10^{-7}	3.92×10^{-7}
WUSTL_IIoT	Identical	7.46×10^{-11}	1.28×10^{-4}	1.05×10^{-7}	9.00×10^{-9}
	No-Label-Skew	1.62×10^{-7}	5.27×10^{-6}	3.19×10^{-7}	9.51×10^{-6}
	Dirichlet	3.75×10^{-11}	2.50×10^{-6}	2.45×10^{-7}	1.31×10^{-6}
Edge_IIoT	Identical	7.103×10^{-7}	7.18×10^{-3}	7.224×10^{-7}	1.747×10^{-2}
	No-Label-Skew	7.798×10^{-11}	2.620×10^{-5}	9.706×10^{-10}	1.038×10^{-2}
	Dirichlet	5.824×10^{-8}	8.064×10^{-4}	1.374×10^{-7}	8.109×10^{-3}

diverse data distributions effectively. The buffer mechanism used in the aggregator server contributes to inefficiencies and delays, as faster agents with less training data tend to fill the buffer quickly, causing imbalances and biases in the training process. In comparison to FedBuff and AsyncFL, **BFL** performs better on both identical and non-identical datasets, as it employs an agent selection method to mitigate the straggler effect. However, since agent selection is only applied in the first round and lacks a dynamic approach within the FL framework, it is somewhat inefficient and less effective compared to DyHFL.

The **ASR_Fed** approaches failed to converge effectively in the experiments, particularly when the number of agents increased to 100. ASR_Fed, while designed to mitigate the straggler problem by prioritizing fast agents (buffer agents) and incorporating updates from slow agents (circumvent agents) only after a computed threshold, still suffers in high-agent scenarios. When the number of agents scales up, the threshold for including slower agents in ASR_Fed increases accordingly, causing many agents to be excluded from early aggregation rounds. This leads to insufficient global representation during training and consequently slows down or prevents convergence altogether.

Table IX presents the two-tailed t-test p-values from the comparison of DyHFL with other FL baselines (Async, Sync, Fedbuff, and BFL) in terms of convergence speed, evaluated across both identical and non-identical datasets.

The **two-tailed t-test p-value** is a statistical measure used to determine whether the observed differences between two methods are statistically significant or could have occurred by random chance. A p-value less than 0.05 indicates that the performance difference is statistically significant at the 95% confidence level.

The results in Table IX show that all p-values are far below the 0.05 threshold, often reaching the order of 10^{-7} to 10^{-20}. This confirms that DyHFL's improvements in convergence speed are not due to random variation but are statistically reliable.

Based on the consistent performance of DyHFL across all settings demonstrates that DyHFL significantly outperforms other baselines regardless of dataset type or data heterogeneity. This validates its robustness and generalizability in real-world IIoT scenarios, driven by its dynamic approach to balancing participation between fast and slow agents.

*3) **Communication Cost**:* Table X shows the settings that have been considered for this calculation.

TABLE X: Experimental Parameters

Parameter	**Value**
Rounds	10
Local epochs	10
Number of agents	20,40,80,100
Sliding window	1/10 of rounds
α and β	0.7 and 0.3
Buffer size for Feddbuff	75% of total agents
Gas_Pipeline dataset Model size	0.009 MB
WUSTL_IIoT dataset Model size	0.002 MB
Edge_IIoT dataset Model size	0.014 MB

According to Fig. 4, and compared to other algorithms, the **DyHFL** reduces communication costs by dividing the total communication cost into preliminary and subsequent rounds. The result of this optimization is lower communication costs, especially when there are a lot of agents.

The **BFL** method also reduces communication costs by dividing total communication costs between the first and subsequent rounds. However, the BFL approach is not as

effective as DyHFL, which results in higher communication costs. **Fedbuff** reduces communication costs by buffering and transmitting updates from a fraction of model updates (75% of total agents), which also minimizes model transmissions. While Fedbuff's buffering reduces communication overhead, it may slow convergence due to potential biases from agents with less data or lower accuracy. In contrast, **SyncFL** transmits updates from all agents in each round, resulting in higher communication costs. SyncFL's comprehensive aggregation captures diverse data patterns, leading to faster convergence despite higher costs.

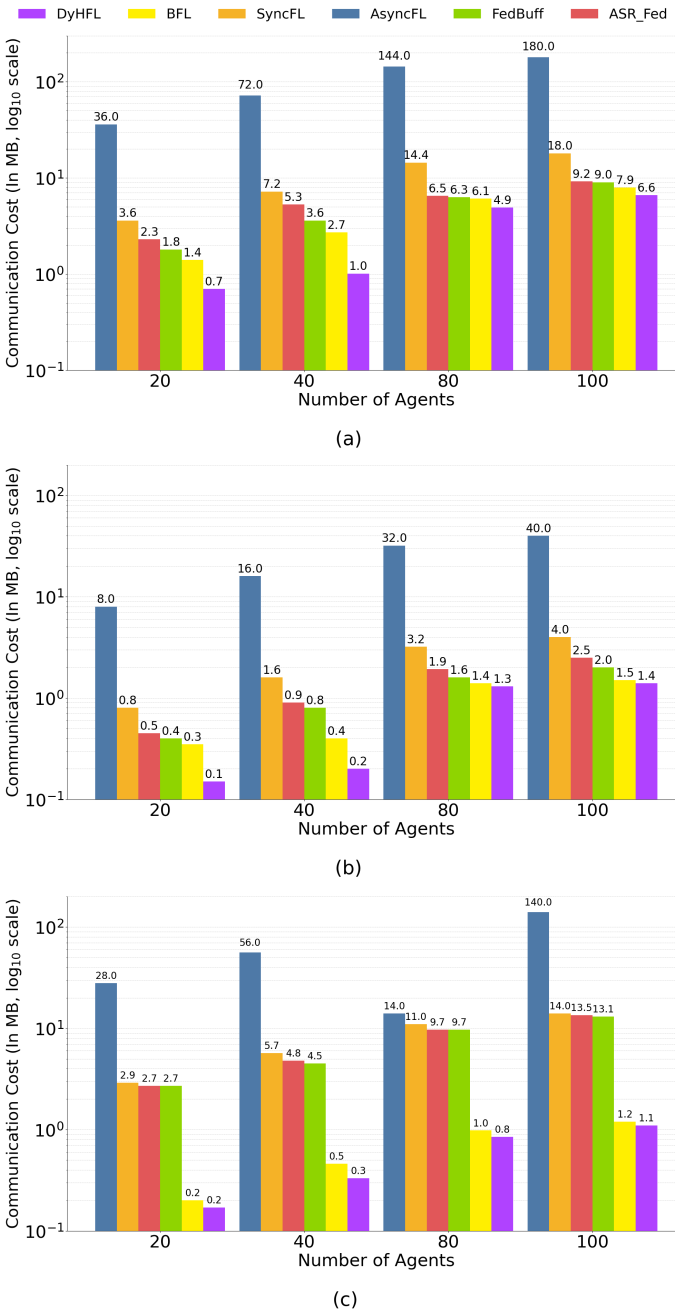

Fig. 4: Comparison of communication cost (exchanged message size in megabytes, presented on a logarithmic scale) across agents in FL methods: (a) Gas_Pipeline Dataset, (b) WUSTL_IIoT Dataset, and (c) Edge_IIoT Dataset.

AsyncFL has the worse results as a consequence of frequent communications with the server. AsyncFL's frequent communications significantly increase its communication costs, making it the least efficient of the other methods.

The communication cost of **ASR_Fed** is lower compared to **AsyncFL** and **SyncFL**, as it prioritizes fast agents and delays the participation of slow agents. This selective participation reduces the frequency and volume of communication, especially in the early rounds, leading to a better communication cost.

Table VIII presents the two-tailed t-test p-values comparing DyHFL against other FL baselines in terms of communication cost across three datasets and varying agent numbers. In all cases, the p-values are well below the standard significance threshold ($p < 0.05$), confirming that the observed differences are statistically significant. Notably, DyHFL consistently achieves lower communication costs, with extremely small p-values (as low as 10^{-11}) when compared to AsyncFL and ASR_Fed, highlighting a substantial performance gap. The consistent statistical significance across datasets and agent scales reinforces the robustness of DyHFL's communication efficiency advantage.

*4) **Model Performance**:* Fig. 5 compares six FL algorithms - SyncFL, AsyncFL, FedBuff, ASR_Fed, BFL, and DyHFL - on the mentioned datasets with 100 agents and 10 rounds, demonstrating their performance through key metrics: Accuracy, Precision, and F1 Score. In the image, each subfigure corresponds to one of three conditions (identical, no-label-skew non-identical, Dirichlet non-identical), and reveals key insights about FL baseline effectiveness in handling identical and non-identical data distributions.

DyHFL consistently shows the best performance across all datasets and conditions, scoring high across all metrics. Its success lies in its balanced selection of agents, which ensures a fair representation of both fast and slow agents. With this strategy, DyHFL is able to handle diverse and imbalanced data distributions more effectively than other algorithms.

BFL achieves similar results to DyHFL due to its approach in selecting agents that ensures fairness between them. However, BFL's results are not as optimal as DyHFL's because BFL's agent selection method lacks the dynamic adaptability of DyHFL, which makes it less effective.

SyncFL also performs well, with results similar to DyHFL's. This is largely due to its synchronous communication method, which ensures uniform model updates across all agents. SyncFL does not include the fairness adjustments that give DyHFL a slight advantage, especially when handling diverse data.

AsyncFL generally performs worse, especially in scenarios with imbalanced datasets, like the Gas_Pipeline dataset, or with highly non-identical data distributions. Its asynchronous approach struggles to handle the challenges posed by such data characteristics, resulting in lower F1 scores. While AsyncFL achieves high precision on the Gas_Pipeline dataset, this is due to the dataset's imbalance, which can lead to misleading results.

FedBuff maintains a competitive position. However, its buffering mechanism poses challenges, especially when deal-

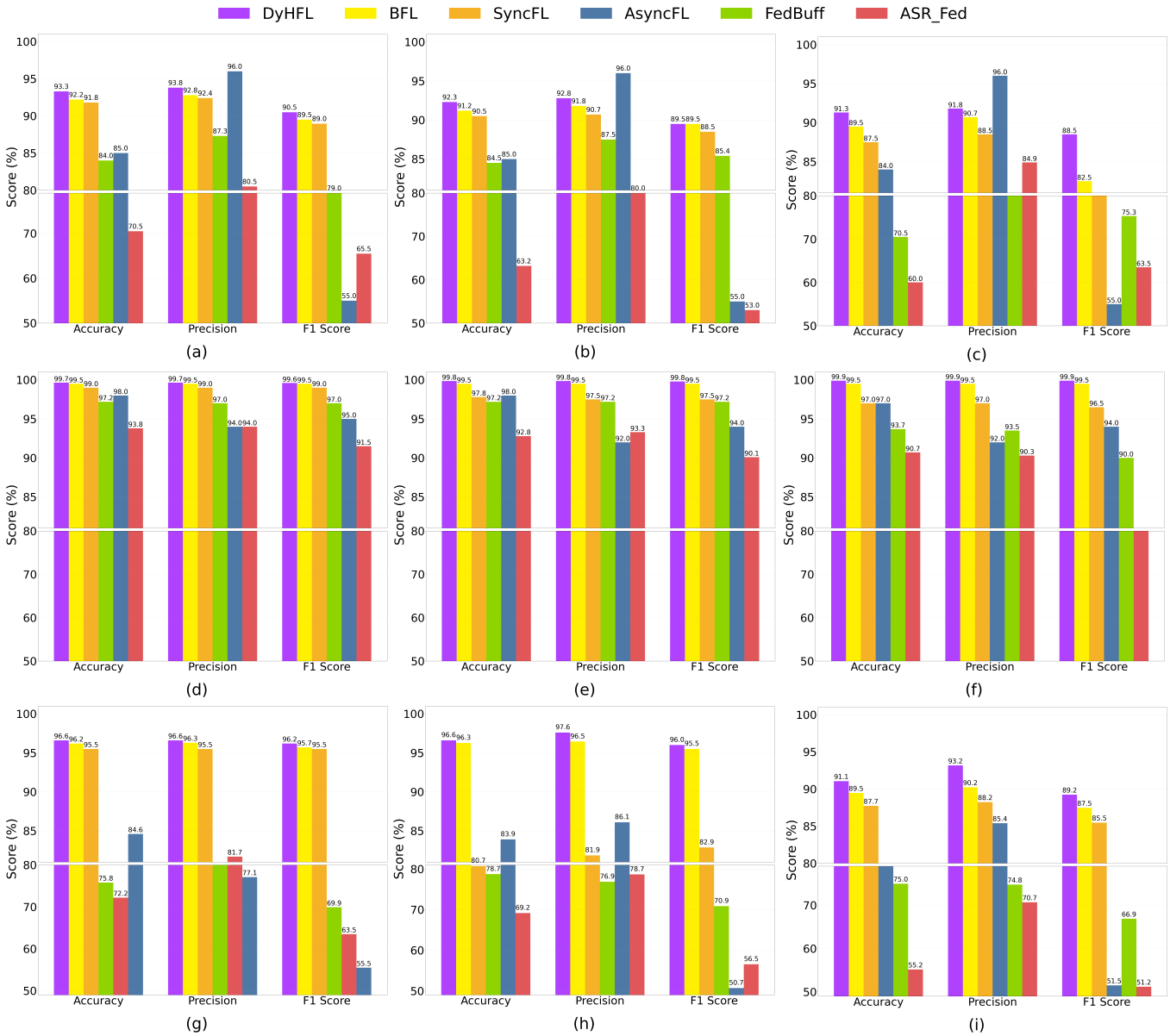

Fig. 5: Model performance comparison of six FL algorithms (SyncFL, AsyncFL, FedBuff, BFL, ASR_Fed, and DyHFL) across different datasets and conditions: (a)–(c) Gas_Pipeline dataset under identical, non-identical with No-Label-Skew distribution, and non-identical with Dirichlet distribution settings; (d)–(f) WUSTL_IIoT dataset under the same three settings; and (g)–(i) Edge_IIoT dataset under the same three settings.

ing with non-identical datasets. In FedBuff, the buffering strategy favors data from faster agents, which can lead to a poor generalization and training bias.

The model performance of **ASR_Fed** is, on average, consistently the lowest across all scenarios and datasets, as shown in Figures (a)–(i). This underperformance can be attributed to its selective aggregation strategy, where only fast agents participate in the early rounds. While this reduces communication cost and mitigates stragglers, it also delays the inclusion of slower agents (circumvent agents), leading to limited representation of the overall data distribution. Consequently, the model lacks diversity in the aggregated updates, which negatively impacts generalization and ultimately results in poorer model performance compared to the other approaches.

The results indicate that DyHFL is more effective for FL tasks with diverse and imbalanced data in a heterogeneous environment. Despite the similar convergence speed and model performance of DyHFL and BFL, DyHFL excels in other important aspects, such as agent selection fairness and communication cost, making it more suitable for real-world applications. This shows that DyHFL is more stable in the presence of straggler agents while preserving performance.

*5) **Component Ablation Analysis**:* To rigorously evaluate the individual contributions of HE and dynamic agent selection to the overall performance of the proposed DyHFL framework, an ablation study was conducted in which each component —either HE or dynamic agent selection—was selectively

removed.

In this analysis, the variants with and without encryption are denoted as *DyHFL Paillier* and *DyHFL Plain*, respectively, while the absence of dynamic selection reverts the framework to the baseline synchronous aggregation, referred to as *SyncFL Paillier* and *SyncFL Plain*. By disabling one component (HE and dynamic agent selection) at a time and monitoring changes in model performance, convergence speed, and communication cost, the study provides a clear quantitative assessment of the role played by each component. Tables XI, XII, and XIII present the ablation results across all settings.

Across all datasets and scenarios, DyHFL—both in encrypted and plaintext modes—achieves substantially faster convergence speeds compared to the synchronous FL (SyncFL) baseline. For example, in the no-label-skew WUSTL-IIoT dataset, DyHFL converges in only 6 rounds, whereas SyncFL requires over 320 rounds. This significant improvement demonstrates the effectiveness of the proposed dynamic agent selection with buffering in mitigating the straggler effect that typically slows down synchronous aggregation. By adaptively selecting a subset of agents within a sliding window and aggregating updates without waiting for all agents to complete their local training, DyHFL reduces delays caused by slower participants. This strategy not only accelerates convergence—often by up to an order of magnitude in heterogeneous data scenarios—but also directly lowers the overall communication cost, as the total cost is proportional to the total number of agents. The reduction in agents due to the DyHFL agent selection mechanism results in more efficient use of bandwidth. Moreover, DyHFL consistently maintains high accuracy, precision, and F1-score across all data distributions, outperforming SyncFL in both predictive quality and training efficiency. The dynamic selection mechanism further preserves participation diversity across rounds, which is crucial for sustaining model generalization under non-identical data distributions. Together, these results confirm that dynamic agent selection is the primary driver behind DyHFL's superior performance, enabling faster, more communication-efficient, and more accurate FL without compromising fairness or robustness.

Incorporating Paillier-based HE into DyHFL enables secure aggregation without exposing individual agent updates, providing strong privacy guarantees. The encryption process has a negligible impact on convergence speed or model performance because the Paillier scheme preserves exact numerical values during aggregation, avoiding quantization or approximation errors. Although HE theoretically increases per-round communication size due to ciphertext expansion, in this study the communication cost is computed using a formula based on model size (M), number of agents (N), total rounds (T), and buffer parameters, with M fixed for both encrypted and plaintext models. This approach isolates the algorithmic impact of HE and dynamic selection, resulting in identical reported communication costs for Paillier and Plain variants. In real deployments that account for the true ciphertext size, Paillier would introduce a modest increase in per-round cost; however, DyHFL's buffering mechanism would still offset this overhead by significantly reducing the total number of rounds, thereby preserving overall communication efficiency while ensuring data privacy.

Overall, the ablation results clearly demonstrate that dynamic agent selection is the dominant factor in enhancing efficiency, while HE strengthens privacy without sacrificing predictive performance or computational efficiency.

VI. Discussion and Future Directions

Recent advances in privacy-preserving FL motivate positioning DyHFL against emerging alternatives. One such method is Federated Transfer Learning (FTL), which allows collaboration among agents with heterogeneous feature or label spaces. While FTL is well-suited for cross-domain FL scenarios, it often lacks strong aggregation privacy guarantees and relies on pre-trained representations. In contrast, DyHFL assumes aligned input spaces but offers enhanced privacy through HE and ensures fairness through dynamic agent selection based on real-time metrics. This makes DyHFL particularly advantageous in IIoT environments where devices vary in capabilities, but privacy, latency, and coordination remain critical.

In the current design, DyHFL incorporates HE-based updates, threshold evaluation, and buffer management, which introduce additional processing time as well as memory and energy demands that may exceed the capabilities of embedded controllers and low-power sensors. The computational cost of encryption and decryption, combined with ciphertext expansion, also increases bandwidth usage and can place additional load on low-rate industrial links. Furthermore, as with most HE-based FL systems, DyHFL employs a trusted third party (TTP) for key generation and distribution, introducing a centralized element that may not align with the requirements of fully decentralized or adversarial IIoT environments. To mitigate this, several future research directions are proposed:

- Employ quantized or selectively applied encryption to reduce computational complexity while preserving confidentiality.
- Adaptive participation mechanisms in which severely resource-constrained devices participate at reduced frequencies, selectively transmit critical model parameters, or employ model update compression techniques to minimize communication and computation overhead.
- Combine the strong aggregation privacy guarantees of DyHFL with the cross-domain adaptability of FTL to address both statistical and feature-space heterogeneity in large-scale IIoT deployments.
- Integration of *Distributed Key Generation* (DKG) offers a promising solution to eliminating the single point of failure inherent in centralized key management. DKG allows agents to collaboratively generate cryptographic keys without reliance on a trusted third party. Incorporating DKG would enhance resilience, support dynamic key rotation, and improve security in decentralized or adversarial environments.

DyHFL achieves strong aggregation privacy and coordinated convergence in IIoT environments through the integration of

TABLE XI: Component Ablation Analysis on Identical Datasets

Dataset	Algorithms	Convergence Speed	Communication Cost	Model Performance		
				Accuracy	Precision	F1-score
Gas_Pipeline	DyHFL Pailliar	67	0.66	93.3	93.8	90.5
	DyHFL Plain	66	0.66	94.3	94.8	91.5
	Sync Pailliar	360	3.60	91.8	92.4	89.0
	Sync Plain	360	3.60	91.8	92.4	89.0
WUSTL_IIoT	DyHFL Pailliar	8	0.15	99.7	99.7	99.6
	DyHFL Plain	8	0.15	99.7	99.7	99.6
	Sync Pailliar	55	0.80	99.0	99.0	99.0
	Sync Plain	54	0.80	98.4	99.2	98.2
Edge_IIoT	DyHFL Pailliar	20	0.17	96.6	96.6	96.2
	DyHFL Plain	21	0.17	94.6	95.6	95.2
	Sync Pailliar	25	2.90	95.5	95.5	95.5
	Sync Plain	25	2.90	95.5	95.5	95.5

TABLE XII: Component Ablation Analysis on No-label-skew Non-Identical Datasets

Dataset	Algorithms	Convergence Speed	Communication Cost	Model Performance		
				Accuracy	Precision	F1-score
Gas_Pipeline	DyHFL Pailliar	55	0.66	92.3	92.8	89.5
	DyHFL Plain	53	0.66	92.7	93.2	90.5
	Sync Pailliar	650	3.60	90.5	90.7	88.5
	Sync Plain	648	3.60	91.3	92.2	89.7
WUSTL_IIoT	DyHFL Pailliar	6	0.15	99.8	99.8	99.8
	DyHFL Plain	6	0.15	99.8	99.8	99.8
	Sync Pailliar	320	0.80	97.8	97.5	97.5
	Sync Plain	324	0.80	96.3	93.2	95.4
Edge_IIoT	DyHFL Pailliar	20	0.17	96.6	97.6	96.0
	DyHFL Plain	20	0.17	96.6	97.6	96.0
	Sync Pailliar	62	0.29	80.7	81.9	82.9
	Sync Plain	61	0.29	80.7	81.9	82.9

TABLE XIII: Component Ablation Analysis on Dirichlet Non-Identical Datasets

Dataset	Algorithms	Convergence Speed	Communication Cost	Model Performance		
				Accuracy	Precision	F1-score
Gas_Pipeline	DyHFL Pailliar	230	0.66	91.3	91.8	88.5
	DyHFL Plain	229	0.66	91.7	92.1	88.7
	Sync Pailliar	700	3.60	87.5	88.5	80.8
	Sync Plain	703	3.60	85.2	85.3	76.4
WUSTL_IIoT	DyHFL Pailliar	8	0.15	99.9	99.9	99.9
	DyHFL Plain	8	0.15	99.9	99.9	99.9
	Sync Pailliar	400	0.80	99.5	99.5	99.5
	Sync Plain	400	0.80	99.4	99.2	99.6
Edge_IIoT	DyHFL Pailliar	96	0.17	91.2	93.2	89.2
	DyHFL Plain	93	0.17	92.7	94.8	91.3
	Sync Pailliar	116	0.29	87.7	88.2	85.5
	Sync Plain	114	0.29	88.4	88.9	86.3

HE with dynamic agent selection. The above mitigations form a practical roadmap to broaden DyHFL's applicability in real-world IIoT.

VII. Conclusion

In this paper, we introduced a new dynamic FL framework designed to detect anomalies, such as cyber threats, in industrial CPSs. The proposed DyHFL framework incorporates a secure communication protocol based on HE to protect model parameters from model inversion attacks. Additionally, an innovative agent selection strategy was developed. It effectively balances the performance of fast and slow agents in heterogeneous environments, minimizing straggler effects and reducing communication bottlenecks. Extensive experiments, conducted using two real-world industrial CPS datasets, demonstrated that DyHFL does not only achieve superior prediction accuracy but also converges more quickly compared to existing FL approaches, proving its effectiveness and efficiency in practical applications.

Acknowledgment

This work is partly conducted at ICTFICIAL Oy, Finland. It is supported in part by the European Union's Horizon Europe research and innovation program HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933), and the 6G-Path project under Grant No. 101139172. The paper reflects only the authors' views, and the European Commission bears no responsibility for any utilization of the information contained herein.

References

[1] S. K. Poorazad, C. Benzaïd, and T. Taleb, "Blockchain and deep learning-based ids for securing sdn-enabled industrial iot environments,"

in *Proc. of IEEE Globecom'23, Kuala Lumpur, Malaysia*, 2023, pp. 2760–2765.

[2] M. A. Khan, M. R. Karim, and Y. Kim, "A scalable and hybrid intrusion detection system based on the convolutional-lstm network," *Symmetry*, vol. 11, no. 4, p. 583, 2019.

[3] A. N. Jahromi, H. Karimipour, and A. Dehghantanha, "An ensemble deep federated learning cyber-threat hunting model for industrial internet of things," *Computer Communications*, vol. 198, pp. 108–116, 2023.

[4] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, B. D. Thang, K. P. Tran *et al.*, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Computers in Industry*, vol. 132, p. 103509, 2021.

[5] Y. Liu, N. Kumar, Z. Xiong, W. Y. B. Lim, J. Kang, and D. Niyato, "Communication-efficient federated learning for anomaly detection in industrial internet of things," in *Proc. GLOBECOM 2020-2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2021.

[6] R. A. Sater and A. B. Hamza, "A federated learning approach to anomaly detection in smart buildings," *ACM Transactions on Internet of Things*, vol. 2, no. 4, pp. 1–23, 2021.

[7] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏot: A federated self-learning anomaly detection system for iot," in *Proc. IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, Oct. 2019.

[8] Z. Ming, H. Yu, and T. Taleb, "Federated deep reinforcement learning for prediction-based network slice mobility in 6g mobile networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 937–11 953, 2024.

[9] C. Benzaïd and T. Taleb, "Ai for beyond 5g networks: a cyber-security defense or offense enabler?" *IEEE network*, vol. 34, no. 6, pp. 140–147, 2020.

[10] T. Taleb, C. Benzaïd, R. Addad, and K. Samdanis, "AI/ML for Beyond 5G Systems: Concepts, Technology Enablers & Solutions," *Elsevier Journal on Computer Networks*, p. 110044, 2023.

[11] P. Boobalan, S. P. Ramu, Q.-V. Pham, K. Dev, S. Pandya, P. K. R. Maddikunta, T. R. Gadekallu, and T. Huynh-The, "Fusion of federated learning and industrial internet of things: A survey," *Computer Networks*, vol. 212, p. 109048, 2022.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., Florida, USA, 2017.

[13] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2020.

[14] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *Proc. 25th International Conference on Artificial Intelligence and Statistics*, Valencia, Spain, 2022.

[15] S. K. Poorazad, C. Benzaïd, and T. Taleb, "A novel buffered federated learning framework for privacy-driven anomaly detection in iiot," in *Proc. of IEEE Globecom'24, Cape Town, South Africa*, 2024.

[16] A. N. Jahromi, H. Karimipour, and A. Dehghantanha, "An ensemble deep federated learning cyber-threat hunting model for industrial internet of things," *Computer Communications*, vol. 198, pp. 108–116, 2023.

[17] J. Jithish, B. Alangot, N. Mahalingam, and K. S. Yeo, "Distributed anomaly detection in smart grids: A federated learning-based approach," *IEEE Access*, vol. 11, pp. 7157–7179, 2023.

[18] X. Wang, Y. Wang, Z. Javaheri, L. Almutairi, N. Moghadamnejad, and O. S. Younes, "Federated deep learning for anomaly detection in the internet of things," *Computers and Electrical Engineering*, vol. 108, p. 108651, 2023.

[19] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, "Adaptive federated learning and digital twin for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5605–5614, 2021.

[20] Z. Chen, W. Liao, K. Hua, C. Lu, and W. Yu, "Towards asynchronous federated learning for heterogeneous edge-powered internet of things," *Digital Communications and Networks*, vol. 7, no. 3, pp. 317–326, 2021.

[21] Y. Zhang, M. Duan, D. Liu, L. Li, A. Ren, X. Chen, Y. Tan, and C. Wang, "Csafl: A clustered semi-asynchronous federated learning framework," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China, 2021.

[22] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2021.

[23] Z. Liu, C. Guo, D. Liu, and X. Yin, "An asynchronous federated learning arbitration model for low-rate ddos attack detection," *IEEE Access*, vol. 11, pp. 18 448–18 460, 2023.

[24] S. Liu, Y. Yu, Y. Zong, P. L. Yeoh, L. Guo, B. Vucetic, T. Q. Duong, and Y. Li, "Delay and energy-efficient asynchronous federated learning for intrusion detection in heterogeneous industrial internet of things," *IEEE Internet of Things Journal*, vol. 11, no. 8, pp. 14 739–14 754, 2024.

[25] V. U. Ihekoronye, C. I. Nwakanma, D.-S. Kim, and J. M. Lee, "Asr-fed: agnostic straggler-resilient semi-asynchronous federated learning technique for secured drone network," *International Journal of Machine Learning and Cybernetics*, vol. 15, no. 11, pp. 5303–5319, 2024.

[26] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706.

[27] L. Cui, Y. Qu, G. Xie, D. Zeng, R. Li, S. Shen, and S. Yu, "Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3492–3500, 2021.

[28] F. Wibawa, F. O. Catak, M. Kuzlu, S. Sarp, and U. Cali, "Homomorphic encryption and federated learning based privacy-preserving cnn training: Covid-19 detection use-case," in *Proc. European Interdisciplinary Cybersecurity Conference*, Barcelona, Spain, 2022.

[29] J. Ma, S.-A. Naas, S. Sigg, and X. Lyu, "Privacy-preserving federated learning based on multi-key homomorphic encryption," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 5880–5901, 2022.

[30] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, 2021.

[31] A. Madi, O. Stan, A. Mayoue, A. Grivet-Sébert, C. Gouy-Pailler, and R. Sirdey, "A secure federated learning framework using homomorphic encryption and verifiable computing," in *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, 2021, pp. 1–8.

[32] S. Awan, F. Li, B. Luo, and M. Liu, "Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain," in *Proc. ACM SIGSAC conference on computer and communications security*, London, United Kingdom, 2019.

[33] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2020.

[34] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, "Partially encrypted multi-party computation for federated learning," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 828–835.

[35] C. Zhang, S. Ekanut, L. Zhen, and Z. Li, "Augmented multi-party computation against gradient leakage in federated learning," *IEEE Transactions on Big Data*, pp. 1–10, 2022.

[36] E. Rabieinejad, A. Yazdinejad, A. Dehghantanha, and G. Srivastava, "Two-level privacy-preserving framework: Federated learning for attack detection in the consumer internet of things," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4258–4265, 2024.

[37] Y. Chen, B. Wang, H. Jiang, P. Duan, Y. Ping, and Z. Hong, "Pepfl: A framework for a practical and efficient privacy-preserving federated learning," *Digital Communications and Networks*, vol. 10, no. 2, pp. 355–368, 2024.

[38] T. Morris and W. Gao, "Industrial control system (ics) cyber attack datasets: Gas pipeline and water storage tank," ICS Data Sets, 2014.

[39] M. Zolanvari, "Wustl-iiot-2021 dataset for iiot cybersecurity research," IEEE DataPort, 2022.

[40] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning," IEEE DataPort, 2023.