A Novel Buffered Federated Learning Framework for Privacy-Driven Anomaly Detection in IIoT

Samira Kamali Poorazad*, Chafika Benzaïd*, and Tarik Taleb†
* Oulu University, Finland; Oulu University, Finland; † Ruhr University Bochum (RUB), Germany
Emails: samira.kamalipoorazad@oulu.fi, chafika.benzaid@oulu.fi, tarik.taleb@rub.de

GLOBECOM 2024 - 2024 IEEE Global Communications Conference | 979-8-3503-5125-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/GLOBECOM52923.2024.10901786

***Abstract*—Industrial Internet of Things (IIoT) is highly sensitive to data privacy and cybersecurity threats. Federated Learning (FL) has emerged as a solution for preserving privacy, enabling private data to remain on local IIoT clients while cooperatively training models to detect network anomalies. However, both synchronous and asynchronous FL architectures exhibit limitations, particularly when dealing with clients with varying speeds due to data heterogeneity and resource constraints. Synchronous architecture suffers from straggler effects, while asynchronous methods encounter communication bottlenecks. Additionally, FL models are prone to adversarial inference attacks aimed at disclosing private training data. To address these challenges, we propose a Buffered FL (BFL) framework empowered by homomorphic encryption for anomaly detection in heterogeneous IIoT environments. BFL utilizes a novel weighted average time approach to mitigate both straggler effects and communication bottlenecks, ensuring fairness between clients with varying processing speeds through collaboration with a buffer-based server. The performance results, derived from two datasets, show the superiority of BFL compared to state-of-the-art FL methods, demonstrating improved accuracy and convergence speed while enhancing privacy preservation.**

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

I. Introduction

Industrial Internet of Things (IIoT) is a form of Internet of Things (IoT) implemented in manufacturing and industry to automate processes and produce more effective and appropriate products [1]. IIoT offers a multitude of advantages over traditional Supervisory Control and Data Acquisition (SCASA) systems, including productivity, scalability, and data analysis [1]. Neverthless, the increase in connected devices and the lack of security design in older control systems make factories vulnerable to cyber-attacks such as denial of service (DoS) [1]. In addition to security challenges, data privacy is another critical concern in IIoT environments. This is attributed to the massive amount of data generated by IIoT smart devices, which may encompass sensitive information that service providers are reluctant to disclose to third parties [1]. As a result, an imperative arises for a solution to detect cyber-attacks while maintaining data privacy in IIoT domains.

Many IIoT contexts have employed centralized machine learning-based anomaly detection schemes as a solution [2]. In most cases, centralized schemes lack the flexibility to adapt to different scenarios, particularly with regard to privacy concerns [3]. This limitation arises from the inherent need of centralized methods to send all IIoT data to a single location for processing. This makes centralized approaches time-consuming (causing inefficiency in communication) and compromises the privacy of sensitive industrial processes [4]. To *preserve privacy* and improve communication efficiency, federated learning (FL) emerges as an appropriate distributed machine learning technique for IIoT environments [3]. FL allows clients to train a global model collaboratively by transmitting parameters and local models to a server instead of sharing raw data [4], [5]. This reduces training time and addresses privacy concerns. FL frameworks support synchronous and asynchronous modes of communication.

Synchronous FL (SFL) [6] requires all clients to submit their local models to the server for aggregation. However, problems may arise when clients train at different speeds due to resource differences, such as differences in memory capacity or heterogeneous data. In this situation, the aggregation server must wait for the slowest client to transmit its local model, resulting in slow convergence speeds and delays. This delay, known as the *straggler effect*, occurs due to resource differences, causing certain clients to lag behind. Therefore, SFL is not suitable for real-time applications. To avoid Straggler effects, asynchronous FL (AFL) [7] was introduced, where aggregation occurs immediately after a client trains its model, without waiting for the slowest client. Nevertheless, most asynchronous approaches experience *communication bottlenecks* because clients can communicate independently with the aggregation server.

To trade off SFL and AFL, buffered base solutions have been proposed. As an example, the buffered FL approach (Fed-Buff) [8] employs a K-size buffer at the server, processing client updates only after receiving K updates. A notable limitation of Fed-Buff is its inability to adapt to speed differences among heterogeneous clients. As a result, the buffer is overwhelmed by faster clients, introducing training bias. Moreover, without considering the differences in data and computational speed among clients, the accuracy of the global model could suffer as the local models of faster clients might be less accurate. Consequently, when leveraging different methods of FL within an IIoT heterogeneous environment, it is necessary to take into account the *tradeoffs between model convergence speed, model accuracy, and balance between clients with varying speeds*.

Furthermore, although FL method provides privacy preservation compared to traditional centralized methods, it is still vulnerable to privacy attacks involving model poisoning and model inference [9]. Attackers may eavesdrop on the communication channel or compromise the server to access model updates, resulting in data leakage [9], [10]. Thus, it is essential to *enhance the privacy of FL* in addition to focusing on privacy

concerns in IIoT through FL. Homomorphic encryption (HE) can be used to enhance FL privacy by aggregating encrypted model updates [9]. In contrast to some secure FL schemes, such as differential privacy (DP)-based FL, which introduce inevitable accuracy losses due to added noise, FL with HE preserves model accuracy while protecting data privacy [11].

The challenges previously highlighted motivate us to develop a Buffered FL (BFL)-based anomaly detection model for IIoT. This model effectively addresses three critical challenges from a new perspective: stragglers, communication bottlenecks, and privacy-preservation. BFL achieves this by combining synchronous and buffered FL concepts in a novel manner, despite heterogeneous client speeds. A global model in BFL is synchronously transmitted only to clients who have submitted their local parameters to the server's buffer within a specified time. Based on the training times of each client, the specific time is calculated using a novel weighted average time method. The proposed weighted average time method is designed to ensure fairness between fast and slow clients, minimizing the straggler effects, communication bottlenecks, and training bias. Moreover, HE-based secure communication is implemented to enhance model parameter privacy. The main contributions of this work are as follows:

1) Proposing a new anomaly detection framework that leverages a deep learning (DL)-based FL approach and HE to empower privacy-enhanced detection of cyber threats in industrial cyber-physical systems (CPS).
2) Developing BFL, a novel FL framework that combines the potential of synchronous and buffered FL approaches to address straggler and communication issues.
3) Designing a novel weighted average time method to balance a trade-off between fast and slow clients.
4) Utilizing two distinct datasets to demonstrate the generalization capabilities of BFL.

This article is organized as follows. Section II categorizes some related articles. Section III introduces the proposed method, system model, and attack model. Section IV presents implementation details and discusses the evaluation results. The article concludes in Section V.

II. Related Work

This section provides a categorized review of FL-based intrusion/anomaly detection schemes for IIoT environments.

A. FL-based Intrusion/Anomaly Detection

1) SFL approaches: The authors of [2] proposed a communication efficient-FL framework based on Convolutional Neural Network (CNN)-Long Short-Term Memory (LSTM) to reduce communication costs by using gradient compression and local computations. In order to identify the best gradients, the Top-k algorithm is used. In [3], a FL approach based on LSTM is proposed for detecting anomalies in energy consumption in smart buildings. Authors in [4] proposed an anomaly detection system based on federated self-learning to detect malicious devices in IoT. The system uses Gated Recurrent Units (GRUs) to classify data based on thresholds. Additionally, the self-learning mechanism improves the detection performance of the global model as the IoT environment changes.

The study in [12] aims at detecting anomalies in industrial control systems (ICS), leveraging a Variational Autoencoder-LSTM model to capture temporal dependencies and complex patterns. The work in [13] presents IIoT cyber-threat hunting model that uses a One-Class Support Vector Machine (OCSVM), Isolation Forests (IF), and Stacked Autoencoders (SAE). A combination of OCSVM and IF is used to detect potentially malicious data points. The SAE helps identify patterns and relationships in data by extracting features.

In addition to the lack of privacy-preserving methods, the discussed synchronous methods are particularly susceptible to straggler effects when more clients are involved. The straggler effect results in delays and slow convergence in real-time industrial domains.

2) AFL approaches: In [14], an AFL is proposed for Software Defined Networking systems with distributed control to improve efficiency. Models are trained by local controllers and are uploaded to root controllers asynchronously. Local models are aggregated by root controllers. In [15], a spectral clustering method based on the latency and direction of model updates is proposed to address the issue of model staleness caused by asynchronous updates. The scheme also improved test accuracy and convergence speed in non-independent and identically distributed (non-IID) datasets. In [16], the authors proposed an AFL-based digital twin architecture for IIoT applications to minimize straggler effects. Results showed faster convergence and higher learning rates with the suggested model. In [17], the authors propose a Semi-Asynchronous Federated Averaging (SAFA) to solve low round efficiency and poor convergence by using a novel aggregation algorithm with a cache structure.

Investigated asynchronous methods require more frequent communication between clients and server, resulting in higher communication costs. Convergence is also slowed by frequent aggregation on the server.

B. Privacy-enhancing FL

Although FL methodology offers privacy-preserving benefits, there are still privacy concerns associated with it. Various privacy-enhancing technologies can be used to enhance the effectiveness of FL.

In [18], FL-CNN is used to minimize sensitive information sharing, and HE is leveraged to enhance privacy. In [19], authors introduced a novel distributed FL scheme in an IIoT scenario based on K-means, distributed random forest, and AdaBoost algorithms with the incorporation of DP and HE techniques to enhance data privacy and provide safe data sharing among IIoT devices. The work in [20] proposes an approach to enhance privacy preservation of FL using HE. It addresses common issues such as privacy breaches, communication overheads, and lack of accountability. In [21], a FL based on CNN and GRU was employed to improve ICS intrusion detection accuracy by learning both spatial and

temporal features of network traffic data. HE is utilized in this method to enhance privacy.

Various privacy-enhancing techniques, such as DP and HE, have been successfully integrated into FL approaches to enhance data privacy in industrial scenarios. While DP is effective in protecting client privacy during aggregation, HE is preferred in certain cases to enhance the privacy of model updates and ensure data/parameter confidentiality. In this work, our objective is to improve the privacy of model updates using HE, as DP can introduce noise affecting model performance, such as utility.

III. Methodology

This section introduces the system and attack models as well as the proposed methodology for empowering privacy-driven FL-based anomaly detection in IIoT.

A. System Model

Fig. 1 illustrates a high-level system overview of the proposed privacy-preserving FL training method. It consists of industrial agents and an aggregation server. Industrial agents represent owners of CPSs. A third authority generates public and private keys first. Industrial agents construct a local DL model using their own industrial CPS data, encrypt the parameters of the model using the public key, and send them to the aggregation server. The aggregation server aggregates the encrypted model parameters from each industrial agent. The aggregated encrypted parameters are then sent back to the industrial agents. The industrial agents update their DL models by decrypting the aggregated parameters with the private key. This process is repeated until the model is converged.

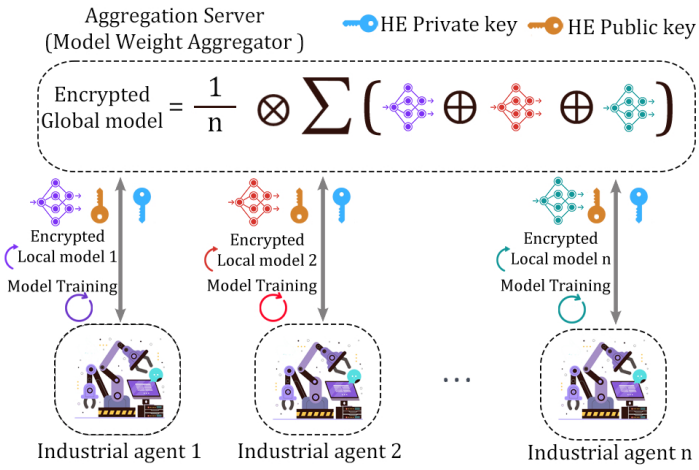

Fig. 1. High-level architecture for privacy-preserving BFL.

B. Attack model

The threat model includes cyber threats to industrial CPSs and the BFL framework. To provide a comprehensive understanding of the risks faced by industrial CPSs and the BFL framework, this work examines the following cyber threats:

1) Cyber Threats Against Industrial CPSs: We examine various cyber threats to industrial CPS remote processes, including response injection attacks, command injection attacks, reconnaissance attacks, and DoS attacks. Response injection attacks involve falsified response messages to querying entities, while command injection attacks introduce false commands into the industrial control systems. Reconnaissance attacks aim to gather information about industrial CPSs, and DoS attacks overload a target system, causing disruptions in industrial CPSs.
2) Cyber Threats Against BFL Framework: External attackers or malicious eavesdroppers may also intercept communication links. This study considers eavesdropping on the model parameters. Data resources can be analyzed through the parameters of the anomaly detection model in this attack.

C. Proposed Methodology

A novel FL framework and a HE-based secure communication protocol are used to network multiple industrial CPS owners to develop a DL anomaly detection model. We use a partial HE; a type of HE that permits both the addition and multiplication of ciphertexts. The rationale behind this choice is rooted in the superior speed of partial HE compared to fully HE. It is also motivated by the fact that partial HE is, in most cases, sufficient for homomorphic calculations, especially for FLs that only require addition and multiplication.

The BFL framework employs the SFL strategy to avoid communication bottlenecks. In this manner, the updated model will not be sent individually to each industrial agent, such as AFL. A buffer-based server is also used to prevent stragglers. Thus, BFL server does not wait for all agents to send their local models before aggregating. Rather, BFL server waits until a specific time for agents to send their local models. To calculate the specific time, avoid overloading the buffer-based server, and avoid training bias, the weighted average time method is used during the first iteration of BFL. The weighted average time method determines the specific time based on the training time spent by each agent. Algorithms 1 and 2 summarize the detailed training procedure.

The trust authority first generates a public key and a private key. The aggregator server selects initial parameters. In the first iteration of BFL, agents receive initial parameter values from the aggregator server, they train DL-based anomaly detection models locally with their own data (line 7 of Algorithm 1). As each agent trains a local model, the model parameters are encrypted using the public key (line 8 of Algorithm 1). Afterward, the encrypted parameters and training time for each agent are uploaded to the aggregator. After the aggregator server receives the training times and local encrypted parameters of the agents, encrypted global parameters and weighted average time are calculated (lines 13 and 15 of Algorithm 1). As a first step in calculating weighted average time, the aggregator server sorts the received training times in descending order (line 1 of Algorithm 2). Each agent's initial weight is derived by inverting training time (line 3 of Algorithm 2). However, to assign higher weights to slower

agents and lower weights to faster agents, the Initial weights must be reversed to achieve the final weights (line 4 of Algorithm 2).

Algorithm 1: Detection Algorithm

Input:
T: Number of Iterations; N: Number of Clients (Agents);
P_{key}: Public key, S_{key}: Private key;
$\oplus$: HE-based addition; $\otimes$: HE-based multiplication,
S_m: A list of selected clients

Output: Desired performance (e.g., accuracy)

1 $m = Initial\_parameters()$
2 $M_{agg} = 0$ # Encrypted global parameters
3 **for** $t = 1$ *to* T **do**
4 **if** $t = 1$ **then**
5 **for** *each client* $k = 1$ *to* N **do**
6 t_s = Start_Time()
7 $[Local\_m]^k = m.train()$
8 $[m_{enc}]^k = Encrypt([Local\_m]^k, P_{key})$
9 t_e = End_Time()
10 $[T_{\text{client}}]^k = t_e - t_s$
11 # Encrypted aggregation on Server
12 **for** $j \in [m_{enc}]$ **do**
13 $[M_{\text{agg}}] = (j \oplus M_{\text{agg}}) \otimes [N^{-1}]$
14 # Calculating weighted average time
15 $T_{\text{W\_Avg}}$ = Weighted_Average_Time($[T_{\text{client}}]$)
16 **for** $i \in [T_{client}]$ **do**
17 **if** $i \leq T_{W\_Avg}$ **then**
18 $[S_m] = i$
19 **else**
20 **for** *each client* $k \in [S_m]$ **do**
21 $[m_{dec}]^k = Decrypt([M_{agg}], S_{key})$
22 $[Local\_m]^k = m_{dec}.train()$
23 $[m_{enc}]^k = Encrypt([Local\_m]^k, P_{key})$
24 # Encrypted aggregation on Server
25 **for** $j \in [m_{enc}]$ **do**
26 $[M_{\text{agg}}] = (j \oplus M_{\text{agg}}) \otimes [N^{-1}]$

Algorithm 2: Weighted_Average_Time

Input: $Trainig\ Time\ of\ each\ client\ (Agent) = T^1_{client}, T^2_{client}, ..., T^N_{client}$
Output: Weighted Average Time

1 $Sort\ {T_{client}}^1, {T_{client}}^2, ..., {T_{client}}^N$ *in descending order* **for** *each* $i = 1$ *to* n **do**
2 # Initial Weight = $[W_1, W_2, ..., W_N]$
3 [Initial_Weight]$^i = \frac{1}{{T_{client}}^i}$
4 # Reverse Initial weight [Final_weight] $= [W_N, W_{N-1}, ..., W_1]$
5 $W_f = \sum_{j=1}^{N}$ Final_weight^j
6 $T_{fw} = \sum_{j=1}^{N}$ Time_training$^j \cdot$ Final_weight^j
7 $T_{\text{W\_Avg}} = \frac{T_{fw}}{W_f}$

Finally, the average weighted time can be obtained by multiplying each agent's training time by its corresponding final weight over the sum of all final weights (line 7 of Algorithm 2). The encrypted global parameters are then sent back to the agents. Agents obtain updated model parameters by decrypting the encrypted parameters model with a private key. After that, the DL model parameters are updated. For the second iteration of BFL to convergence, only agents with training times less than the calculated weighted average time will be selected (lines 19 to 26 of Algorithm 1).

IV. Experiment Setting and Evaluation

A. Experimental Settings

We evaluated BFL using two different datasets: (i) Gas pipeline dataset [22], which consists of 18 features related to Modbus protocol in 8 classes including normal, DoS, Naive/Complex Malicious Response Injection, Malicious State/Parameter Command Injection, Malicious Function Code Injection, and Reconnaissance; and (ii) the WUSTL-IIoT Dataset [23], which consists of 41 features of real cyber-attacks related to network flow in 5 classes: DoS, Reconnaissance, Backdoor, Command injection, and Normal.

BFL is compared to the three FL methods: FedBuff [8], SFL [6], and AFL [7]. We have implemented BFL in PyTorch and utilized the Paillier Framework for HE.

Both a CNN and a Multilayer Perceptron (MLP) were trained on two specified datasets. The CNN architecture consists of two 1D convolutional layers, two fully connected layers, one MaxPooling layer, and one AveragePooling layer. For both datasets, the first convolutional layer has 8 filters of size 3, and the second convolutional layer has 16 filters of size 3. The MaxPooling layer and AveragePooling layer, each has a kernel size of 2. The Rectified Linear Unit (ReLU) activation function is applied in the convolutional layers, and Softmax is used for multi-class classification. The MLP architecture applied to both datasets comprises three fully connected layers. For the Gas Pipeline dataset, the first layer maps the 18-dimensional input to a 54-dimensional space, the second layer reduces this to a 20-dimensional space, and the final layer maps it to 8 classes. For the WUSTL-IIoT dataset, the first layer maps the 41-dimensional input to a 9-dimensional space, the second layer also maps this to a 9-dimensional space, and the final layer maps it to 5 classes. Stochastic Gradient Descent (SGD) was selected as the optimization function. For the Gas pipeline dataset, a batch size of 64 was used, while a batch size of 1000 was used for the WUSTL-IIoT dataset. The learning rate and momentum were set at 0.01 and 0.8, respectively. These hyperparameters were determined through trial and error, with the combination of a 0.01 learning rate and 0.8 momentum yielding the best results. Due to space constraints, we present only the results of the model trained with SGD using these optimal hyperparameters. Datasets are divided into three parts: 80% for training, 10% for validation, and 10% for testing. The data is then normalized using the MinMaxScaler, which scales the values between 0 and 1.

After defining the model with the aforementioned values and executing it, the model is evaluated using the accuracy metric, overall training time (millisecond (ms)), and convergence speed (i.e., the number of iterations required to achieve a target accuracy).

B. Comparison of two DL algorithms on BFL

As a first step, we compared the efficiency of BFL based on CNN and MLP algorithms in terms of accuracy, and overall training time to determine the best algorithm to use for the next steps. In order to compare them in terms of overall training time, we take into account 5 clients (industrial agents), and the target accuracy of 94.7% for the Gas pipeline dataset and 99.8% for WUSTL-IIoT dataset. Both accuracy targets are achieved when all data is available. The accuracy of both models is also compared based on 5 clients, two iterations,

TABLE I
BFL OVERALL TRAINING TIME(MS) WITH 5 CLIENTS.

Modes	GAS Pipeline Target accuracy: 94.7%	WUSTL-IIoT Target accuracy: 99.8%
MLP	1655.78	5965.07
CNN	1995.05	8465.18

TABLE II
BFL ACCURACY EVALUATION WITH 5 CLIENTS AND 2 ITERATIONS.

Modes	GAS Pipeline	WUSTL-IIoT
MLP	94.59%	99.70%
CNN	93.60%	98.49%

TABLE III
MLP-BASED BFL - OVERALL TRAINING TIME(MS) BY INCREASING CLIENTS.

Clients	GAS Pipeline Target accuracy: 94.7%	WUSTL-IIoT Target accuracy: 99.8%
2	551.12	1562.53
5	1655.78	5965.07
10	2483.67	10737.15

10 local epochs, 0.01 learning rate, and stochastic gradient descent (SGD) optimization for both datasets. As shown in Table I and Table II, MLP outperforms CNN in both datasets. The simplicity of the MLP architecture, with fewer parameters to learn, compared to CNN with more layers, such as pooling layers, allows it to converge faster to the target accuracy. Therefore, MLP is a more appropriate approach for real-time environments with tabular data. Consequently, we select MLP as the baseline algorithm for further comparisons in the next sections (IV-C and IV-D).

C. Overall training time evaluation based on client numbers

It is evident from Table III that the overall training time for BFL increases as the number of clients increases. This is a result of two factors. Firstly, when datasets contain specific samples, dividing the data among a greater number of clients results in each client getting a smaller portion of the overall dataset. As a result, each client must train on a smaller dataset, resulting in slower convergence. Secondly, the increase in the number of clients requires the server to communicate with more clients to produce the aggregated model, which results in the algorithm taking longer to converge to the target accuracy.

D. Efficacy of communication and the Straggler effect of BFL

Accuracy and convergence speed allow to evaluate FL methods' robustness against stragglers and communication efficiency. We consider the same target accuracies that are mentioned earlier when evaluating the convergence speed of each FL method. Moreover, we compare model accuracy based on 5 clients, 10 iterations for the Gas pipeline dataset, and 4 iterations for the WUSTL-IIoT dataset. SFL and AFL have foundational implementations detailed in [6] and [7], respectively. Although FedBuff's accuracy and convergence speed increase with increasing buffer size, choosing the appropriate buffer size is its weakness. Therefore, we assume that the buffer size in FedBuff is set to half the number of clients participating in the heterogeneous environment. Accordingly, for a scenario with five clients, we consider a buffer size of two.

To simulate speed heterogeneity, we assigned 5 random integer values to 5 clients (1$\sim$ 3s to three of them and 5$\sim$10s to two of them). A higher random value indicates a slower client, whereas a lower random value indicates a faster client. Additionally, to better demonstrate the effectiveness of the proposed model in scenarios involving heterogeneous datasets, we assume varying data sizes for each client. This means that clients with more data will have longer training times, while those with less data will train more quickly.

To simulate such a scenario, each of the three fast clients is allocated 5% random samples from the training dataset, while the two slow clients are randomly assigned 80% and 90% samples of the training data, respectively.

The results depicted in Fig. 2 demonstrate the superiority of BFL in achieving higher attack detection accuracy, compared to baseline FL methods. We observe that BFL outperforms SFL, AFL and FedBuff in terms of accuracy by, respectively, 0.18%, 3.53%, and 1.63% on Gas pipeline dataset and 0.13%, 6.9%, and 1.1% on WUSTL-IIoT dataset. This is attributed to BFL's novel weighted average time, which can more effectively engage straggling clients, resulting in better prediction results (3 clients are selected by BFL). As SFL and BFL follow the same synchronous updating strategy, they have the closest prediction performance. AFL achieved the lowest accuracy, which can be explained by the fact that AFL aggregates weights from one client at a time and does not provide an effective way to deal with stragglers. In addition, FedBuff is unable to converge faster because its buffer is rapidly filled with clients that have less data and less accuracy.

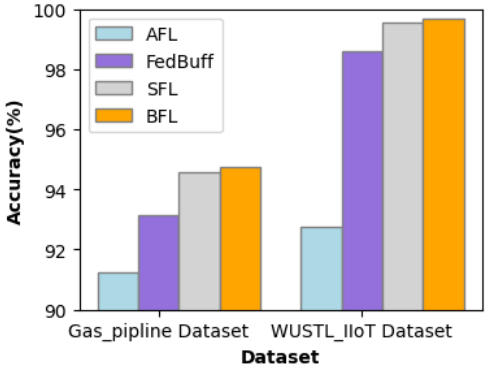

Fig. 2. Accuracy comparison of MLP-based algorithm: 5 clients, 10 iterations for Gas pipeline and 4 iterations for WUSTL-IIoT.

A comparison of the convergence speed results shown in Fig. 3 also reveals a clear difference in performance. On both datasets, BFL converges faster towards the target accuracy than all three other methods. On Gas pipeline dataset, BFL convergences 1.33, 16.66, and 11.11 times faster than SFL, AFL and FedBuff, respectively. Similarly, BFL surpasses SFL, AFL and FedBuff by, respectively, 2, 10, and 8.4 times on WUSTL-IIoT dataset. The faster convergence of BFL is the result of the balance that BFL creates between fast and slow clients, avoiding to wait for the slowest client like in

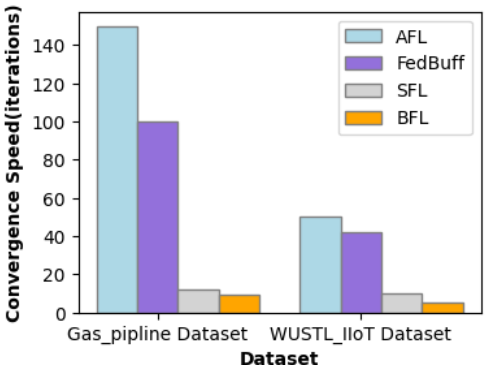

Fig. 3. Convergence speed comparison of MLP-based algorithms.

SFL or favor faster clients as in FedBuff. Additionally, the convergence speed result for AFL indicates that AFL methods, where the server simply communicates with all clients, suffer from a severe communication bottleneck.

Hence, considering both attack detection performances and convergence speed, BFL provides the best performance-cost balance.

V. Conclusion

In this paper, we proposed a novel FL framework, named BFL, to detect cyber threats against industrial CPSs. BFL employs a HE-based secure communication protocol that preserves the privacy of model parameters during training. In BFL, the following modules are synthesized cohesively: (1) a buffering strategy to handle stragglers; (2) synchronous updating of the global model to prevent communication bottlenecks; (3) a novel, weighted average time method that the FL server uses to balance between fast and slow clients in a heterogeneous environment. In extensive experiments using two real industrial CPS datasets, BFL achieved the highest prediction accuracy, converged the fastest, and had superiority over state-of-the-art FL schemes. In future work, we intend to improve the weighted average time method to dynamically adapt the selection of clients at each training iteration based on the changing computation capability of clients and the fluctuating network conditions. The methodology will also be incorporated into an Encryption as a Service (EaaS) platform, which could provide scalability benefits [24].

Acknowledgment

This research work is partially supported by the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022, the Research Council of Finland 6G Flagship program (Grant No. 346208), and the European Union's Horizon Europe research and innovation programme HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933). The paper reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information this paper contains.

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*, Dec. 2023, pp. 2760–2765.

[2] Y. Liu *et al.*, "Communication-efficient federated learning for anomaly detection in industrial internet of things," in *Proc. GLOBECOM 2020-2020 IEEE Global Communications Conference*, Taipei, Taiwan, Feb. 2021.

[3] 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, Aug. 2021.

[4] T. D. Nguyen *et al.*, "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.

[5] M. Farhoudi *et al.*, "Discovery of 6g services and resources in edge-cloud-continuum," *IEEE Network*, pp. 1–1, 2024.

[6] B. McMahan *et al.*, "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., Apr. 2017.

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

[8] J. Nguyen *et al.*, "Federated learning with buffered asynchronous aggregation," in *Proc. 25th International Conference on Artificial Intelligence and Statistics*, Valencia, Spain, Feb. 2022.

[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, Sept. 2020.

[10] S. Kianpisheh and T. Taleb, "Collaborative federated learning for 6g with a deep reinforcement learning based controlling mechanism: A ddos attack detection scenario," *IEEE Transactions on Network and Service Management*, pp. 1–1, Apr. 2024.

[11] P. Boobalan *et al.*, "Fusion of federated learning and industrial internet of things: A survey," *Computer Networks*, vol. 212, p. 109048, Jul. 2022.

[12] T. Huong *et al.*, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Computers in Industry*, vol. 132, p. 103509, Jul. 2021.

[13] 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, Jan. 2023.

[14] X. Ma *et al.*, "Asynchronous federated learning for elephant flow detection in software defined networking systems," in *Journal of Physics: Conference Series*, vol. 2216, no. 1, Mar. 2022, p. 012085.

[15] Y. Zhang *et al.*, "Csafl: A clustered semi-asynchronous federated learning framework," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China, Sept. 2021.

[16] W. Sun *et al.*, "Adaptive federated learning and digital twin for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5605–5614, Aug. 2021.

[17] W. Wu *et al.*, "Safa: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, May 2021.

[18] F. Wibawa *et al.*, "Homomorphic encryption and federated learning based privacy-preserving cnn training: Covid-19 detection use-case," in *Proc. European Interdisciplinary Cybersecurity Conference*, Jun. 2022, pp. 85 – 90.

[19] B. Jia *et al.*, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4049–4058, Jun. 2021.

[20] 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*, Nov. 2019.

[21] B. Li *et al.*, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, Sept. 2020.

[22] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *Proc. of the 1st International Symposium on ICS & SCADA Cyber Security Research (ICS-CSR 2013)*, Sept. 2013, pp. 22 – 29.

[23] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Wustl-iiot-2021 dataset for iiot cybersecurity research," Oct. 2021. [Online]. Available: https://www.cse.wustl.edu/∼jain/iiot2/index.html

[24] A. Javadpour *et al.*, "Encryption as a service for iot: Opportunities, challenges, and solutions," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 7525–7558, Dec. 2024.