Sensors and Materials, Vol. 38, No. 2 (2026) 871–885
MYU Tokyo

S & M 4353

Blockchain-integrated Federated Learning with Adaptive Fallback Mechanism for Resilient Automation Systems

Chuan-Kang Liu,[1] I-Hsien Liu,[2] Bo-Sian Liao,[2] and Jung-Shian Li[2*]

[1]Dept. of Artificial Intelligence and Computer Engineering, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 411030, Taiwan (R.O.C.)
[2]Dept. of Electrical Engineering, National Cheng Kung University, No. 1, Daxue Rd., East Dist., Tainan City 701401, Taiwan (R.O.C.)

(Received October 8, 2025; accepted December 12, 2025)

Keywords: automation systems, federated learning, blockchain, client dropouts

Automation systems play a critical role in modern industry, enabling efficient, precise, and large-scale operations across sectors such as manufacturing and smart infrastructure. These systems rely heavily on AI to process data, optimize decision-making, and enhance system reliability. As automation increasingly depends on distributed data from numerous devices, federated learning (FL) has emerged as an attractive solution. However, in real-world deployments of FL, client dropout attacks are commonly encountered, which significantly degrade the performance of automation systems employing FL. To mitigate the impact of dropouts on the training process, we propose a blockchain-integrated FL architecture with an adaptive fallback mechanism (BFL-AF). By leveraging the transparency and immutability of the blockchain, the system can effectively verify client participation and securely record model updates during each training round. Furthermore, an adaptive fallback mechanism is introduced, which utilizes clients' historical model weights to enhance training stability and recovery capability. Experimental results demonstrate that the proposed method significantly improves convergence speed and robustness under various abnormal dropout scenarios, offering strong resilience and a practical solution for a building privacy-preserving and security-enhanced FL-based automation system.

1. Introduction

Automation systems play a critical role in modern industry, enabling efficient, precise, and large-scale operations across domains such as manufacturing, healthcare, and smart infrastructure. These systems rely heavily on vast amounts of distributed sensor and device data to support real-time monitoring, decision-making, and control. However, traditional centralized learning approaches, which require transferring data from edge devices to a central server, introduce significant challenges. They not only create heavy communication overhead and

*Corresponding author: e-mail: jsli@cans.ee.ncku.edu.tw
https://doi.org/10.18494/SAM5967

ISSN 0914-4935

single points of failure but also raise serious privacy concerns and may violate regulations such as the General Data Protection Regulation (GDPR).

To address these issues, federated learning (FL) has gained increasing attention. FL is a decentralized machine learning framework that enables multiple clients to collaboratively train a global model without sharing raw data. This architecture preserves data privacy, reduces communication costs, and leverages distributed computing resources. As a result, FL has been successfully applied in smartphones,[1] IoT devices,[2] and the healthcare domain, and is increasingly considered a promising solution for automation systems that demand secure, efficient, and large-scale coordination across heterogeneous and geographically distributed environments. However, the decentralized nature of FL brings new challenges concerning, for example, client heterogeneity, communication efficiency, and system security. In real-world deployments, particularly in automation systems, clients often differ in hardware, network conditions, and data distribution, making model convergence difficult and unstable. Furthermore, there are also several attacks occurring in the FL framework, such as poisoning attack, backdoor attack and privacy attack.[3–5]. In this study, we focus on a new attack, termed client dropout attack,[6,7] in FL and propose an architecture that integrates blockchain technology with an adaptive fallback mechanism. The goal is to enhance model stability and robustness in environments with high participation variability, which is especially critical for automation systems that rely on consistent and reliable model performance. The proposed framework maintains model accuracy during abnormal dropouts and defends against targeted attacks on high-contribution clients. First, a blockchain-based participation verification mechanism is introduced to store the model updates of all clients. Blockchain's immutability and traceability ensure the secure logging of model updates and the verification of client participation,[8] which enhances trust and defends against malicious behavior.[9,10] Second, we design an adaptive fallback mechanism that uses clients' historical model updates as a surrogate, thus maintaining system stability while reducing the impact of dropouts. This approach outperforms traditional averaging aggregation algorithms by improving convergence and accuracy. Experimental results on real-world datasets show that the proposed method improves model accuracy and convergence in several dropout scenarios, demonstrating a higher resilience than baseline methods. The integration of the blockchain and adaptive fallback mechanism provides a practical and secure solution for future FL-based automation system design.

2. Related Works

FL is an emerging decentralized machine learning paradigm designed to address challenges in data privacy and model training in distributed automation systems. Unlike traditional centralized approaches, FL keeps data on end devices for local training and only exchanges model parameters after each training round. This design not only safeguards user privacy but also complies with privacy regulations such as GDPR, and it has been widely applied in scenarios including smartphones, IoT devices, vehicular networks, and edge computing. This section comprises a description of the background and work related to this study.

2.1 FL architecture

In an automation system, a typical FL training process involves the following steps: First, the central server initializes the global model parameters $w(0)$ and broadcasts them to all clients for the first local training round. Here, clients denotes the edge devices in a distributed automation system. Each client then trains this initial model on its local dataset, D_i, for several iterations to produce an updated model, using a standard loss function such as cross-entropy or mean squared error (MSE). Once training is complete, clients send their updated parameters or gradients back to the server. The server collects updates from all participating clients and performs weighted averaging to produce the new global model expressed by

$$w_{global}^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} w_i^{(t)}, \tag{1}$$

where N denotes the number of clients involved in the current round. Finally, the updated global model is broadcast again to all clients, initiating the next training round, and the above process repeats until the model converges or the predefined number of rounds is reached. Equation 1 represents an FL aggregation approach, namely, federated averaging (FedAvg), which is regarded as the most representative of FL aggregation algorithms.[11] Its main operation involves aggregating all updated local model parameters via weighted averaging to produce the global model. However, FedAvg is highly sensitive to data distribution homogeneity (IID assumption) and the number of local training epochs. In cases where data is non-IID or clients have inconsistent training conditions, the model may diverge or become unstable. To address these issues, the federated proximal (FedProx) algorithm was proposed as an extension of FedAvg.[12] FedProx incorporates a proximal term into the local training objective to penalize deviations between the local and global models, thereby improving convergence stability. This method is particularly well suited to heterogeneous environments where client computational capacities, data volumes, or communication frequencies vary.

2.2 Blockchain

Blockchain is a decentralized distributed ledger technology that leverages cryptography and consensus algorithms to ensure the traceability and immutability of information across nodes.[13] Each transaction or event is stored in the form of a block, which is linked to previous blocks through a hash function, as illustrated in Fig. 1, ensuring that historical records in the chain cannot be arbitrarily altered.[14,15] This technology effectively reduces the risk of model poisoning or data tampering, thereby enhancing the overall security and stability of the system.

For data integrity verification in blockchain, the Merkle Tree structure is often employed to validate model updates, ensuring that data remains unaltered while enabling fast consistency checks. Private blockchain refers to a blockchain network in which read, write, and verification rights are restricted to specific participants. It is commonly used for internal data sharing and trusted communication within enterprises, institutions, or closed-loop systems. Kim *et al.*[16]

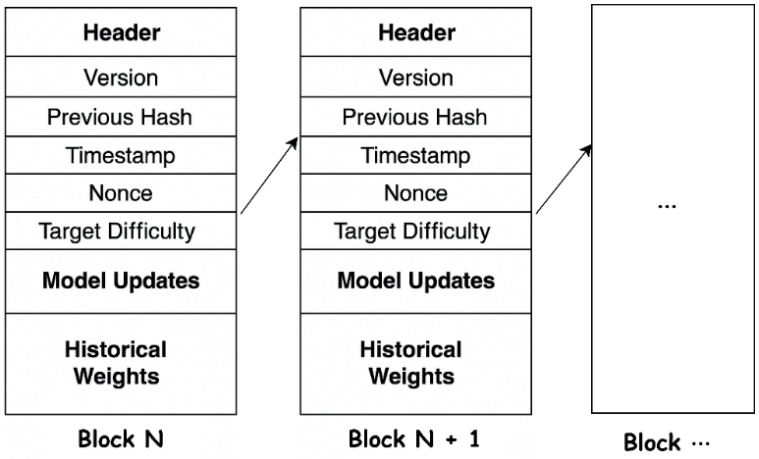

Fig. 1. Blockchain architecture.

have proposed blockchain FL to serve as the storage and query basis for recording client model updates in FL.

Compared with a public blockchain, a private blockchain exhibits several characteristics: (1) controlled node permissions—only authorized devices or user nodes are allowed to participate in block creation and validation, making it suitable for applications that require identity authentication; (2) high transaction processing efficiency—the limited number of nodes reduces resource consumption and excessive validation procedures, thereby shortening consensus time and increasing transaction throughput; and (3) enhanced data privacy—model parameters and records are exchanged only between authorized nodes, strengthening the credibility and security of the model update process. In the system design of this research, after completing local model training, each client uploads the updated weight parameters to a miner node, which encapsulates these updates into a block and appends it to the chain.

Proof of work (PoW), one of the most widely used consensus mechanisms in blockchain systems,[17] was originally introduced by Bitcoin to ensure the uniqueness of block generation and the immutability of network data. The core concept of PoW is that nodes (miners) must repeatedly perform computations to find a nonce such that, when combined with the block header and hashed, the result meets a predefined difficulty condition. In this mechanism, each miner packages the pending transaction data into a block data structure and computes its hash value. If the resulting hash is less than the target value specified by the system, the miner successfully mines the block and broadcasts it to the network, where other nodes verify its validity before adding it to the chain.

3. Proposed FL Framework

The architecture proposed in this study, as illustrated in Fig. 2, integrates blockchain technology into the FL process to enhance the FL system's defense capability and resilience against abnormal behaviors and potential attacks. The overall workflow is summarized as

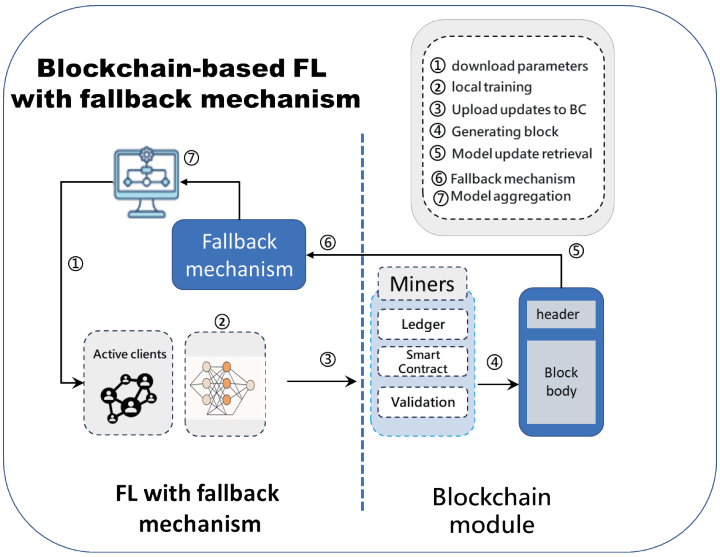

Fig. 2. (Color online) Integrated system architecture.

follows. At the beginning of FL, the server distributes the global model parameters to all participating clients for local training. After completing local model training, each client submits its updated model parameters to the blockchain module. The server subsequently retrieves the collected model updates from the latest block and aggregates them to generate the updated global model, which is then redistributed to the clients in the next FL round. Throughout this process, the blockchain module is responsible for verifying the legitimacy of submitting clients, recording the metadata of model updates into the blocks, and finalizing the block addition to the blockchain. In this blockchain module, the ledger in a miner stores all confirmed blocks containing clients' model update records, serving as a tamper-resistant historical audit log. Smart contracts are used to verify the legitimacy of clients, ensuring secure and policy-compliant participation in FL.[18] Validation refers to the PoW consensus mechanism, in which miners perform computational work to validate new blocks and maintain synchronization through the longest-chain rule, thereby ensuring the consistency and trustworthiness of the system. In cases of persistent or random client dropout attacks, the server can rely on the FL fallback mechanism and retrieve historical model update records from the blockchain, which serves as a tamper-resistant log to maintain the integrity and continuity of global model updates. Hence, the proposed FL framework is composed of two core components, a blockchain module and an FL algorithm enhanced with an adaptive model fallback mechanism. The details of the proposed FL framework for automation systems are presented as follows.

3.1 Blockchain module design

In this design, a blockchain architecture centered on blocks and miners is adopted, operating in a private-chain mode specifically for recording the local model update history in FL. As

shown in the overview of our proposed blockchain-based FL framework, after completing local training, each client sends update information—including the client ID, model weight parameters, and timestamp—to its associated miner. Once the miner has received updates from the clients, it encapsulates them into the block body and creates a new block according to predefined threshold conditions (e.g., when the total weight reaches a predefined threshold).

Each block consists of a block body and a block header. In addition to the metadata illustrated in Fig. 1, each block stores multiple model update records, the computed historical weights described in Eq. (5), the current hash digest, and a hash pointer in the block header that links to the previous block, forming a chain structure that prevents data tampering. To ensure tampering resistance and secure auditability, each block in the blockchain module also produces a fixed-length hash digest that uniquely represents the encapsulated header and body data, using a cryptographic hash function (e.g., SHA-256).

Following the PoW mechanism, the newly created block is broadcast to all miners and appended to the ledger. Every miner maintains its own local chain and uses the longest-chain rule to remain synchronized with other miners, ensuring the integrity, transparency, and trustworthiness of model update records. Figure 3 illustrates the logical architecture of the blockchain module in our proposed FL framework. If a client fails to submit its update within a communication round, it is considered disconnected. The system then retrieves the client's last historical update parameters from the blockchain as a fallback mechanism, thereby maintaining the integrity and continuity of global model update records even under client dropout conditions.

3.2 FL algorithm based on adaptive fallback mechanism

In the adaptive fallback mechanism, when a client drops out in a given round, the system retrieves its historical model weights from the blockchain to serve as substitutes. If the client successfully uploads updates in the current round, the system performs a weighted fusion of the current and historical weights to smooth the update curve and enhance training stability.

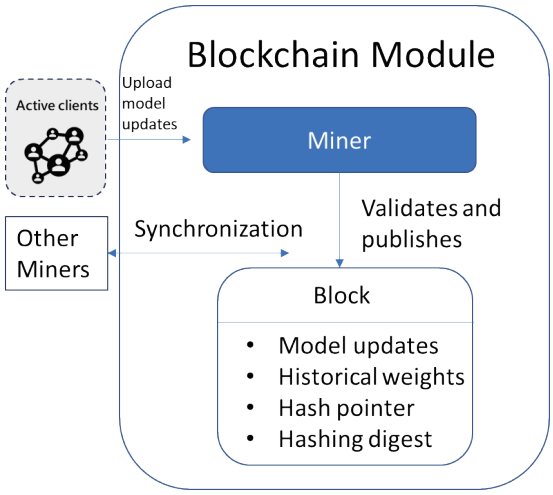

Fig. 3. (Color online) Logical architecture of the blockchain module in FL.

$$AW_i^{(t)} = \begin{cases} \beta \cdot W_i^{(t)} + (1-\beta) \cdot AW_i^{(t-1)}, & \text{for normal case} \\ AW_i^{(t-1)}, & \text{for dropout case} \end{cases} \quad (5)$$

- $AW_i^{(t)}$ represents the updated weight of client i after aggregation in round t.
- $W_i^{(t)}$ denotes the weight obtained from client local training in the current round t.
- $AW_i^{(t-1)}$ refers to the historical weight of client i retrieved from the blockchain for round $t-1$.
- β is the exponential decay factor, serving as the decay coefficient, and is set to the recommended value of 0.8 in this study.

Algorithm 1
Blockchain-integrated FL with adaptive fallback

Require: Total rounds T, Clients C = $\{C_1, C_2, ..., C_N\}$, Decay parameter β, Dropout round T_{drop}, Local epochs E, Proximal coefficient μ

Ensure: Final global model weights $w_{global}^{(T)}$, accuracy logs

Initialize global model weights $w_{global}^{(0)}$

Initialize blockchain

for $t = 1$ to T do

  for each client ∈ C do

    if C_i is active then

      $w_i^{(t)} \leftarrow \text{LocalTrain}\left(w_{global}^{(t-1)}, D_i, E\right)$

      Retrieve historical weights $aw_i^{(t-1)}$ from blockchain

      $aw_i \leftarrow \beta \cdot w_i^{(t)} + (1-\beta) \cdot aw_i^{(t-1)}$

    else

      Retrieve historical weights $aw_i^{(t-1)}$ from blockchain

      $aw_i^{(t)} \leftarrow aw_i^{(t-1)}$

    end if

    Record $aw_i^{(t)}$ to blockchain

  end for

  $w_{global}^{(t)} \leftarrow \text{FedProxAggregation}\left(\left\{aw_i^{(t)}\right\}, \mu\right)$

  Evaluate accuracy $Acc^{(t)}$

  Record $Acc^{(t)}$ to blockchain

end for

return Final global model weights $w_{global}^{(T)}$, accuracy logs

The blockchain-integrated FL algorithm is designed to address client disconnections and network instability. By leveraging blockchain technology, each client's model update history is securely recorded, allowing the system to activate the adaptive fallback mechanism based on historical weights. After completing local training, a client's model parameters are combined with its historical weights stored on the blockchain using the decay coefficient β. If a client leaves for some reason in the current round, its historical weights are directly used for a surrogate update, ensuring system stability.

During each training round, all client updates are recorded on the blockchain, and the central server collects the model updates from the blockchain and aggregates them using the FedProx aggregation algorithm to form the updated global model. This process maintains the stability and consistency of the global model even under client disconnections or delayed updates, while the immutability of the blockchain guarantees the traceability of historical updates, enhancing the transparency and security of the FL system.

4. Performance Evaluations

The experimental design of this study is aimed at evaluating the resilience and stability of a blockchain-integrated FL framework system under client dropout conditions, and to verify whether the proposed adaptive fallback mechanism can effectively mitigate the negative impact of dropouts on the global model's accuracy. The experiments utilize widely used image datasets, such as those from the Modified National Institute of Standards and Technology (MNIST) and Canadian Institute for Advanced Research 10 (CIFAR-10), and organize data into five data subsets to simulate the non-IID distributions commonly found in real-world environments. Each data subset was assigned four clients, resulting in a total of 20 clients. For CIFAR-10, image categories, such as vehicles and animals, were divided into five data subsets, while the MNIST digits were grouped sequentially on the basis of their numerical order. Table 1 presents data subset settings for CIFAR-10 and MNIST datasets. The model architecture employed in this experiment is a convolutional neural network (CNN) featuring convolutional and pooling layers. The total number of trainable parameters of the proposed CNN model is approximately 2.17 million (exact number: 2169770).

Table 1
Data subset settings and class names.

Dataset	Data subset	Class range	Class names
CIFAR-10	Data Subset 1	0, 1	Airplane, Automobile
	Data Subset 2	2, 3	Bird, Cat
	Data Subset 3	4, 5	Deer, Dog
	Data Subset 4	6, 7	Frog, Horse
	Data Subset 5	8, 9	Ship, Truck
MNIST	Data Subset 1	0, 1	Low digits
	Data Subset 2	2, 3	Small digits
	Data Subset 3	4, 5	Middle digits
	Data Subset 4	6, 7	Slightly Larger digits
	Data Subset 5	8, 9	Highest digits

To evaluate the system's stability under various dropout attack scenarios, we design three client dropout conditions: Shapley-value-based dropout, persistent dropout, and random dropout. During a given training round, certain clients will leave and not participate in FL. The aim of this section is to effectively reflect the performance of our proposed FL framework under real-world conditions, including unstable networks or targeted attacks against high-contribution nodes. Next, we introduce the following three dropout attacks in this study:

- Random dropout serves as the most intuitive scenario, simulating sudden disconnections caused by network instability, power interruptions, or device limitations. In each round, a subset of active clients is randomly selected to drop out, while they may rejoin in subsequent rounds, mimicking nonsystematic disconnections and their impact on global model accuracy and model convergence.
- Persistent dropout refers to long-term disconnections owing to network failures, hardware faults, or attacks, with two variants: the client cluster partial retention mode and client cluster full dropout mode. The former is to ensure at least one active client per client cluster to test the model performance under minimal participation, and in the latter, we attempt to disconnect all clients in a selected client cluster to observe the effect of losing entire data sources, particularly in non-IID distributions or sparse class scenarios.
- Shapley-value-based dropout simulates a disconnection attack from an adversarial perspective, where attackers are assumed to identify clients with high contributions to the global aggregation in FL. Effectively quantifying the contributions serves as an important basis for targeted attacks, such as contribution-based disconnection attacks.[19,20] Traditional FL mechanisms typically assume that all clients have equal influence on model updates. However, in real-world scenarios, some clients may hold more representative or critical data, thereby contributing more significantly to the model's convergence speed and final accuracy. Here, each client's marginal contribution is quantified using the Shapley value[21,22] and subsequently ranked. In designated rounds, high-contribution nodes are selected as attack targets. This strategy includes three experimental scenarios. The first, contribution-ranked dropout, selects clients solely on the basis of their Shapley values. The second, partial client cluster retention mode, ensures that at least one client per client cluster is preserved. The third, full client cluster dropout mode, disconnects all clients within the client cluster exhibiting the highest average contribution among all client clusters. These simulations facilitate the evaluation of the proposed fallback mechanism's effectiveness against high-risk targeted attacks, while also assessing system stability and resilience by analyzing both accuracy degradation and recovery speed when critical nodes are lost.

The total number of training rounds is set to 50, with dropout events fixed at middle rounds (e.g., round 25), allowing the model to train on complete data initially and then testing the system's resilience to missing data, as well as the immediate effect of the fallback mechanism under client dropout conditions. For further analysis, in this section, we compare our proposed blockchain-integrated FL with other FL variants, such as traditional FL and FL with a similarity recovery mechanism, to assess model performance under various client dropout scenarios. To evaluate the resilience of each FL variant, a baseline scenario is added, in which FL is performed without any client dropouts. Table 2 gives the full description of various FL frameworks.

Table 2
Description of various FL frameworks.

Method	Explanation
FL-B	This represents the ideal scenario where all clients fully participate in every training round without any dropout. This approach serves as a baseline to evaluate the impact of client dropouts on model performance.
FL	This is a classic FL. It performs the FL algorithm without any recovery mechanism. It reflects the system's basic resilience.
FL-S	This FL variant implements recovery based on the similarity between client models, attempting to identify the parameters of a model with high similarity to the disconnected client and use them as a surrogate.
BFL-AF	This refers to our proposed blockchain-integrated FL with an adaptive fallback mechanism.

4.1 Random dropout

The experimental results in Fig. 4(a) show that under random dropout conditions, blockchain-integrated FL architecture with an adaptive fallback mechanism (BFL-AF) achieves rapid convergence on the MNIST dataset starting from round 10, with accuracy steadily approaching that of FL-B. When the training rounds reach 25, random dropout is applied. As illustrated in Fig. 4(a), the classical FL method exhibits significant accuracy fluctuations, while FL-S yields inconsistent performance. In contrast, our proposed BFL-AF demonstrates strong stability and robustness, achieving a level of accuracy comparable to that of FL-B. On the more challenging CIFAR-10 task, BFL-AF still demonstrates strong resilience to random client dropout. In contrast, both classical FL and FL-S suffer from the absence of local models and achieve a substantially lower accuracy than BFL-AF. These results confirm the effectiveness of BFL-AF in mitigating the impacts of dropout and maintaining reliable convergence. Furthermore, they validate the soundness of our design choices for BFL-AF under such challenging conditions.

4.2 Persistent dropout

In this experimental setup, two scenarios with persistent dropout are analyzed. Here, a suffering client cluster is defined as a client cluster affected by dropout attack. The first scenario is that a suffering client cluster retains at least one active client when persistent dropout occurs, termed cluster-aware dropout attack, while the second scenario is that all clients in the suffering client cluster become completely offline, which is termed cluster-wide dropout attack. In the first scenario, although some clients in the suffering client cluster are disconnected, at least one data source remains active and can still contribute its features to the global model. In contrast, the suffering client cluster in the second scenario cannot contribute its data feature of the client cluster. Figure 5 shows the accuracy results of various FL methods in the first scenario. The results indicate that both classical FL and FL-S exhibit weak resilience to persistent client dropout. In contrast, our proposed BFL-AF leverages a fallback mechanism to maintain a stable training process and achieves high accuracy on both the MNIST and CIFAR-10 tasks, outperforming other methods. In the second scenario, as shown in Fig. 6, classical FL and FL-S suffer significant accuracy degradation and struggle to recover compared with their performance

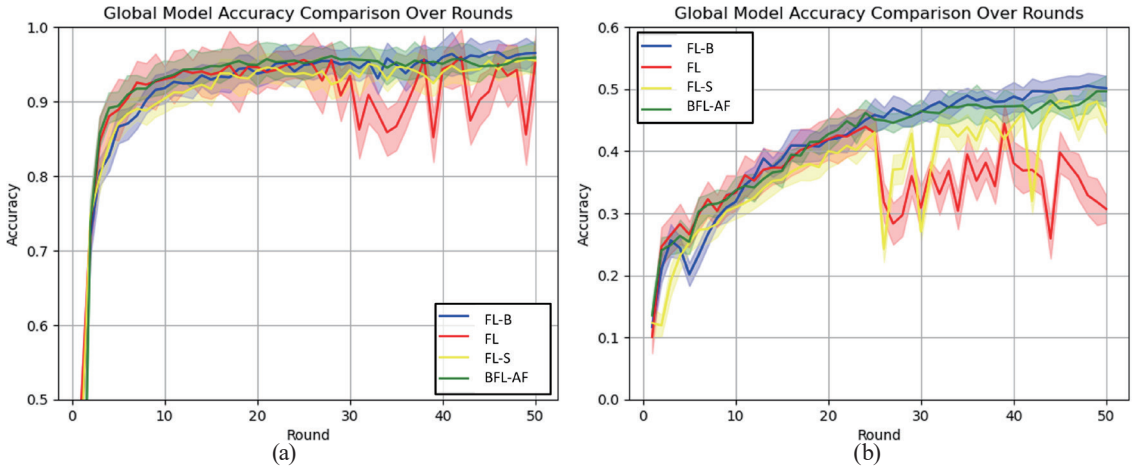

Fig. 4. (Color online) Accuracy rates of global models in FL: (a) accuracy rate for MNIST and (b) accuracy rate for CIFAR-10.

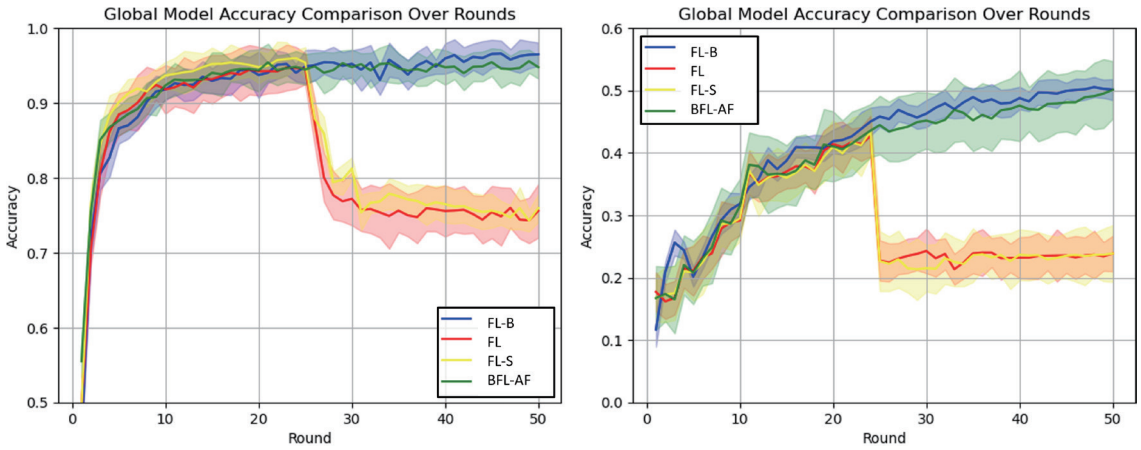

Fig. 5. (Color online) Accuracy under cluster-aware dropout attack.

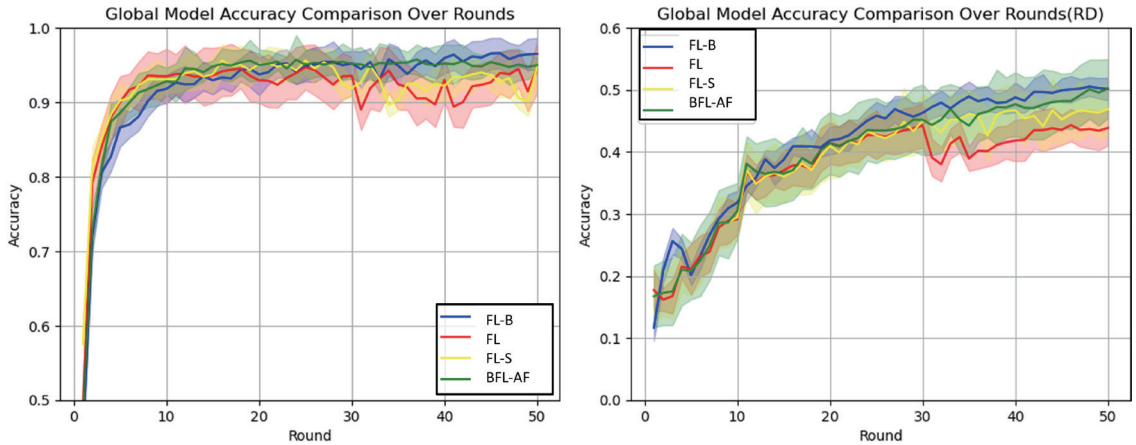

Fig. 6. (Color online) Accuracy under cluster-wide dropout attack.

in the first scenario. In contrast, BFL-AF effectively continues the learning process and gradually improves its model performance, demonstrating strong resilience and reliable compensation, particularly on the CIFAR-10 dataset.

In this study, two scenarios under persistent dropout simulation are analyzed: one where each client cluster retains at least one active client and another where an entire client cluster becomes completely offline. In the first scenario, although some clients are disconnected, the data source is not entirely lost. BFL-AF maintains stable training and high accuracy on both MNIST and CIFAR-10 tasks, outperforming other methods. In the second more extreme scenario where an entire client cluster drops out, FL and FL-S suffer significant accuracy degradation and struggle to recover. In contrast, BFL-AF effectively continues the learning process and gradually improves model performance, demonstrating strong resilience and reliable compensation, especially on the CIFAR-10 dataset.

4.3 Contribution-based dropout

We investigate three targeted dropout attack scenarios based on client contribution. The first is the dropout attack of the highest-contributing individual clients, which is termed contribution-based dropout. The second one is a client cluster dropout attack with at least one active client retained per client cluster, termed contribution-based cluster-aware dropout attack. The final one is a complete dropout attack of high-contribution client clusters, termed contribution-based cluster-wide dropout attack. Experiments on MNIST and CIFAR-10 demonstrate that BFL-AF consistently delivers strong resilience and stability across all cases.

As shown in Fig. 7, in the first scenario, FL and FL-S methods show significant performance degradation when key clients are disconnected, while BFL-AF quickly recovers and approaches the performance of the FL-B model. In the second scenario, where each client cluster retains at least one active client, BFL-AF achieves smoother accuracy curves and outperforms other methods in overall precision, as shown in Fig. 8. In the most extreme case, as shown in Fig. 9, where all high-contribution client clusters are disconnected, BFL-AF still maintains training continuity and mitigates severe accuracy drops.

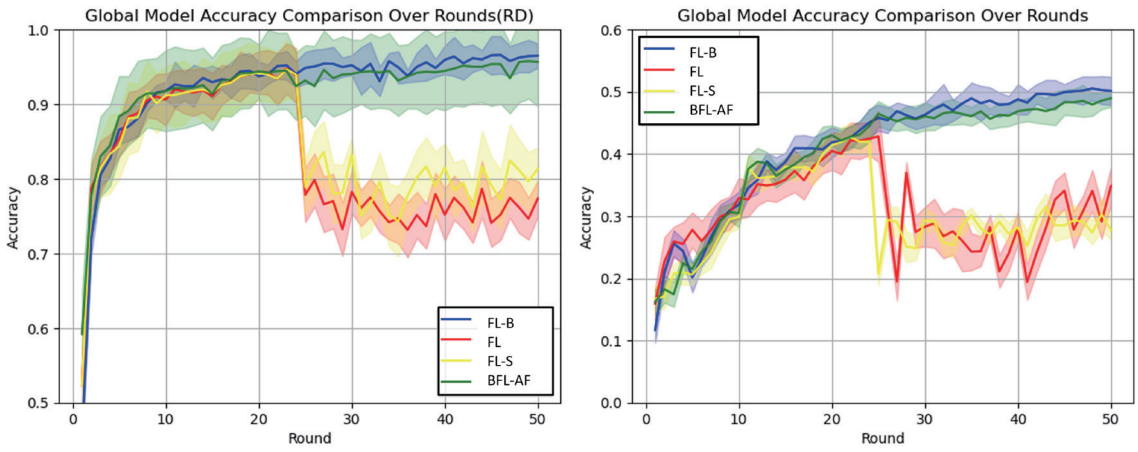

Fig. 7. (Color online) Accuracy under contribution-based dropout attack.

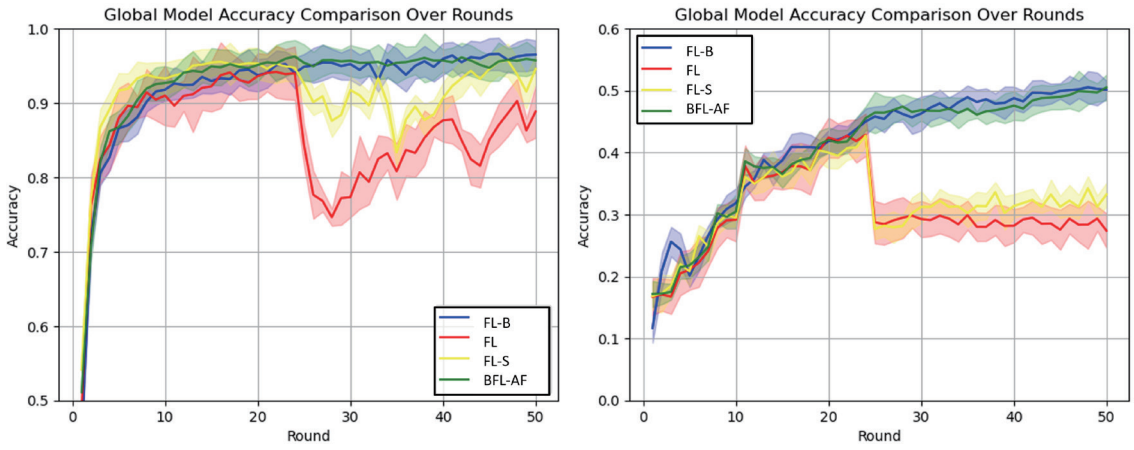

Fig. 8. (Color online) Accuracy under contribution-based cluster-aware dropout attack.

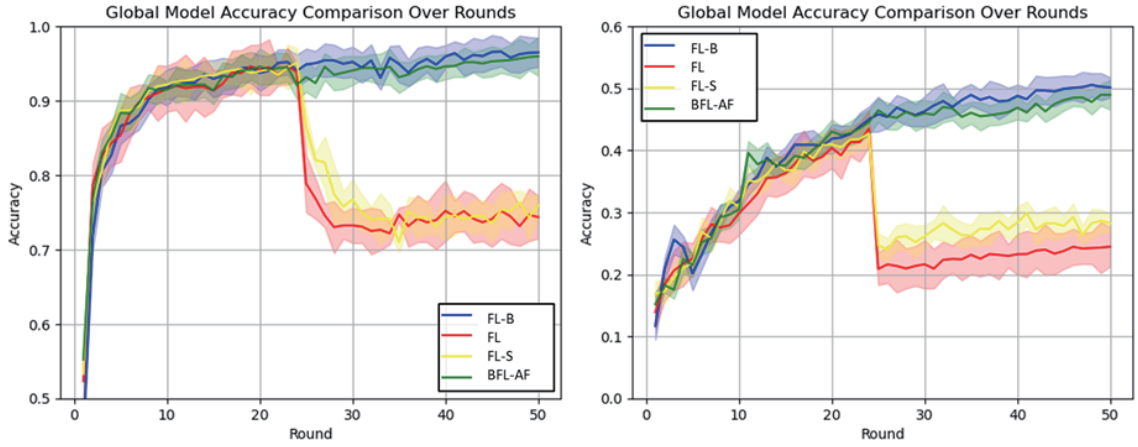

Fig. 9. (Color online) Accuracy under contribution-based cluster-wide dropout attack.

5. Conclusions

FL-based automation systems are gaining increasing attention owing to their benefits in privacy preservation and communication cost reduction. However, a new type of attack has emerged, arising from abnormal client connection dropouts. Hence, in this study, we conducted a systematic analysis and empirical evaluation of the stability and defense capabilities of FL systems facing client dropout and malicious behavior. By designing three dropout scenarios—random, persistent, and contribution-aware targeted dropouts—we compared multiple remediation strategies and their effects on the performance metrics of global models, such as convergence speed and prediction accuracy. From the results of our experiments, we observed that concentrated dropout in critical client clusters causes significantly more harm to model convergence than evenly distributed dropout. Our experiment results demonstrated that the

proposed BFL-AF mechanism, which leverages historical weight information, effectively mitigates performance degradation caused by client dropout. Even under severe conditions—such as the dropout of high-contribution clients or substantial data loss—BFL-AF exhibits strong recovery capability and stable performance. Additionally, the integration of blockchain technology enhanced system security and data integrity by ensuring the traceability and trustworthiness of model updates while also helping our framework filter out malicious clients. Hence, we proposed a FL solution that combines robustness and security, offering a dependable foundation for the deployment of distributed intelligent systems.

Acknowledgments

This work was supported by the National Science and Technology Council (NSTC) in Taiwan under contract number 113-2634-F-006-001-MBK

References

1 W. Ni, J. Zheng, and H. Tian: IEEE Internet Things J. **10** (2023) 11942. https://doi.org/10.1109/JIOT.2023.3253853
2 J. Michalek, V. Skorpil, and V. Oujezsky: Proc. 14th Int. Congr. Ultra Mod. Telecommun. Control Syst. Workshops (ICUMT), Valencia, Spain, 2022. https://doi.org/10.1109/ICUMT57764.2022.9943382
3 Z. Liu, Z. Liu, and X. Yang: Proc. 13th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence), Noida, India, Jan. 19–20, 2023. https://doi.org/10.1109/Confluence56041.2023.10048854
4 D. Wu, L. Hao, B. Wei, K. Hao, T. Han, and L. He: Proc. 7th Int. Symp. Autonomous Syst. (ISAS), Chongqing, China, May 7–9, 2024. https://doi.org/10.1109/ISAS61044.2024.10552567
5 E. Sharma, R. C. Deo, C. P. Davey, B. D. Carter, and S. Salcedo-Sanz: Proc. IEEE Conf. Artif. Intell. (CAI), Singapore, Jun. 25–27, 2024. https://doi.org/10.1109/CAI59869.2024.00259
6 W. Qian, Q. Shen, H. Xu, X. Huang, and Z. Wu: Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Seoul, South Korea, 2024, 4870–4874. https://doi.org/10.1109/ICASSP48485.2024.10446609
7 H. Wang and J. Xu: Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Seoul, South Korea, 2024, 8896–8900. https://doi.org/10.1109/ICASSP48485.2024.10447268
8 D. Huang, Y. Yang, and Y. Huang: Proc. 3rd Int. Conf. Electron. Inf. Eng. Comput. Commun. (EIECC), Wuhan, China, 2023. https://doi.org/10.1109/EIECC60864.2023.10456734
9 H. Kim, J. Park, M. Bennis, and S.-L. Kim: IEEE Commun. Lett. **24** (2020) 1279. https://doi.org/10.1109/LCOMM.2019.2921755
10 Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood: ACM Comput. Surv. **55** (2022) 1. https://doi.org/10.1145/3524104
11 H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas: Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS), Florida, USA, Apr. 20–22, 2017. https://doi.org/10.48550/arXiv.1602.05629
12 T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith: Proc. 3rd MLSys Conf., Austin, TX, USA, Mar. 2–4, 2020. https://doi.org/10.48550/arXiv.1812.06127
13 S. Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf (accessed May 2025).
14 S. Pahlajani, A. Kshirsagar, and V. Pachghare: Proc. 1st Int. Conf. Innov. Inf. Commun. Technol. (ICIICT), Chennai, India, Apr. 25–26, 2019. https://doi.org/10.1109/ICIICT1.2019.8741353
15 P. K. Paul, P. S. Aithal, R. Saavedra, and S. Ghosh: Int. J. Appl. Sci. Eng. **9** (2021) 189. https://doi.org/10.30954/2322-0465.2.2021.7
16 H. Kim, J. Park, M. Bennis, and S.-L. Kim: IEEE Commun. Lett. **24** (2020) 1279. https://doi.org/10.1109/LCOMM.2019.2921755
17 B. M. Yakubu, N. S. M. Jamail, R. Latif, and S. Latif: Comput. Mater. Continua (2025) 1546. https://doi.org/10.32604/cmc.2025.072426
18 Bello Musa Yakubu, Nor Shahida Mohd Jamail, Rabia Latif, Seemab Latif: Comput. Mater. Continua **86** (2026) 28. https://doi.org/10.32604/cmc.2025.072426

19 S. Lundberg and S.-I. Lee: https://arxiv.org/abs/1705.07874 (accessed May 2025).
20 K. Aas, M. Jullum, and A. Løland: https://arxiv.org/abs/1903.10464 (accessed May 2025).
21 H. Zhu, Z. Li, D. Zhong, C. Li, and Y. Yuan: Proc. 3rd IEEE Int. Conf. Digital Twins Parallel Intell. (DTPI), Orlando, FL, USA, Nov. 7–9, 2023. https://doi.org/10.1109/DTPI59677.2023.10365410
22 R. Mitchell, J. Cooper, E. Frank, and G. Holmes: https://arxiv.org/abs/2104.12199 (accessed May 2025).