Future Generation Computer Systems 166 (2025) 107706

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

FedShufde: A privacy preserving framework of federated learning for edge-based smart UAV delivery system

Aiting Yao [a], Shantanu Pal [b,*], Gang Li [b], Xuejun Li [a], Zheng Zhang [a], Frank Jiang [b], Chengzu Dong [c], Jia Xu [a], Xiao Liu [b]

[a] School of Computer Science and Technology, Anhui University, Hefei, China
[b] School of Information Technology, Deakin University, Melbourne, Victoria, Australia
[c] School of Data Science, Lingnan University, Lingnan, Hong Kong

ARTICLE INFO

Keywords:
Internet of Things
Edge computing
UAV
Smart delivery system
Differential privacy
Federated learning
Shuffle model

ABSTRACT

In recent years, there has been a rapid increase in the integration of Internet of Things (IoT) systems into edge computing. This integration offers several advantages over traditional cloud computing, including lower latency and reduced network traffic. In addition, edge computing facilitates the protection of users' sensitive data by processing it at the edge before transmitting it to the cloud using techniques such as Federated Learning (FL) and Differential Privacy (DP). However, these techniques have limitations, such as the risk of user information being obtained by attackers through the uploaded weights/model parameters in FL and the randomness of DP, which limits data availability. To address these issues, this paper proposes a framework called FedShufde (**Fed**erated Learning with a **Shuff**le Model and **D**ifferential Privacy in **E**dge Computing Environments) to protect user privacy in edge computing-based IoT systems, using an Unmanned Aerial Vehicle (UAV) delivery system as an example. FedShufde uses local differential privacy and the shuffle model to prevent attackers from inferring user privacy from information such as UAV's location, flight conditions, or delivery address. In addition, the network connection between the UAV and the edge server cannot be obtained by the cloud aggregator, and the shuffle model reduces the communication cost between the edge server and the cloud aggregator. Our experiments on a real-world edge-based smart UAV delivery system using public datasets demonstrate the significant advantages of our proposed framework over baseline strategies.

1. Introduction

In recent years, the rapid growth of Internet of Things (IoT) systems has been greatly facilitated by emerging technologies such as 5/6G, Artificial Intelligence (AI), and edge computing. Edge computing, in particular, has become a popular choice for many latency-sensitive IoT use case scenarios [1]. For example, edge-based smart delivery systems leverage the power of edge computing to enhance the efficiency and responsiveness of package delivery. These systems utilize edge devices, primarily unmanned aerial vehicle (UAV) (such as drones), to carry out the delivery process. By processing data and making decisions at the edge, these systems can minimize latency and improve real-time tracking and monitoring capabilities. Additionally, edge-based smart delivery systems can optimize route planning, resource allocation, and fleet management, leading to more streamlined and cost-effective operations [2]. Advancements in IoT technologies and edge computing have the potential to transform the delivery industry is significant, providing faster, more reliable, and environmentally friendly delivery services. However, ensuring the privacy of user data remains a pressing issue for edge-based UAV delivery systems [3].

Edge devices, located closer to the data source, collect and process vast amounts of sensitive user information, including personal, financial, and product-related data [4,5]. The decentralized nature of edge computing presents challenges in securing this data, as it is often stored and processed locally on the devices. This poses risks of unauthorized access, data breaches, and potential misuse (including, gaining insights into sensitive information such as delivery routes, customer locations, and even the type of goods being transported) [6]. In addition, the constant communication and data exchange between edge devices and the cloud increases the potential attack surface and the need for rigorous privacy protection measures [7]. For example, in October 2022, a financial firm's network experienced a security breach caused by UAVs. The administrators became aware of the attack when they noticed that the MAC address of an employee, who was later identified as compromised, was logged in from both the local network

* Corresponding author.
E-mail address: shantanu.pal@deakin.edu.au (S. Pal).

https://doi.org/10.1016/j.future.2025.107706
Received 16 August 2024; Received in revised form 14 November 2024; Accepted 5 January 2025
Available online 10 January 2025

and his home location, which was several miles away [8]. Without adequate safeguards, the privacy of user data is at risk, impacting user confidence, regulatory compliance, and the overall success and adoption of edge-based smart UAV delivery systems.

1.1. Motivation and problem statement

The edge-based smart UAV delivery system is vulnerable to several privacy threats that need to be addressed. One such threat involves attackers gaining unauthorized access to users' privacy information by establishing connections with edge servers [9]. These attackers can exploit vulnerabilities in the system's security measures to retrieve sensitive data. Additionally, privacy can be compromised when adversaries intercept and manipulate communications between UAVs and the cloud server [10]. By manipulating uploaded data or parameters, attackers can gain access to confidential information or inject malicious commands into the system. Another privacy threat is posed by individuals or automated entities that engage in jamming activities [11,12]. These attackers flood the communication network with false service requests, disrupting the normal functioning of the UAV delivery system. This jamming can affect the system's ability to securely transmit data and protect users' privacy.

To mitigate these privacy threats, robust security measures should be implemented at multiple levels within the edge-based smart UAV delivery system. These include strengthening the security of edge servers, employing encryption techniques to protect data during transmission, implementing access controls, and employing intrusion detection systems to detect and mitigate attacks. Regular security audits and updates are also essential to stay ahead of emerging privacy threats [13–16] [17].

Privacy-preserving technologies in the edge computing environment for IoT systems mainly focus on Differential Privacy (DP) [18,19], Federated Learning (FL) [20], secure multiparty computing [21], homomorphic encryption [22], etc. In particular, DP provides a method for quantifying, evaluating, and proving the privacy level of the system with certain privacy protection mechanisms. DP technology protects data privacy by adding a small amount of noise and provides rigorous proof and quantitative expression for the risk of privacy leakage, without assuming the background knowledge of the attackers. FL technology was proposed by Google in the 2016 [23] to solve the data island problem by training distributed data with a large number of terminals and learning a high-quality centralized machine learning model. Significantly, the FL does not need to obtain the raw data, which protects users' privacy, and its distributed architecture also reduces the need of storing massive data.

However, both techniques face challenges in maintaining privacy while preserving data utility, dealing with computational constraints, managing communication overhead, and accommodating the diverse capabilities of edge devices in edge-based IoT systems. For DP, limitations include the trade-off between privacy and utility, as the addition of random noise can affect data quality. Computational overhead arises from resource-intensive computations and transformations. For FL, communication overhead occurs due to frequent device-server communication, causing latency. Heterogeneity and resource constraints of edge devices pose challenges, as different hardware capabilities and limited resources (such as memory size or processing speed) affect the learning process. Furthermore, while DP and FL provide privacy-preserving mechanisms, their limitations, such as the potential risk of information leakage in FL and the impact of randomness on data availability in DP, need to be carefully considered when implementing these techniques in edge-based IoT systems [24,25]. Back-link attacks are a significant privacy threat in FL settings [26]. Such attacks exploit the sequential order of model parameter uploads or identifiable patterns in updates to trace them back to their originating devices, revealing sensitive user information. In our proposed framework, the shuffle model is a critical countermeasure against this threat. By randomizing the upload sequence of perturbed model parameters, the shuffle model ensures that an adversary cannot establish a direct link between an update and its source device. This approach disrupts the attacker's ability to exploit temporal correlations or ordering information, effectively mitigating the risk of back-link attacks and enhancing user privacy throughout the FL process.

To address the aforementioned challenges, this paper introduces a novel security framework called FedShufde (**Fed**erated Learning with a **Shuf**fle Model and **D**ifferential Privacy in **E**dge Computing Environments), which specifically aims to protect data privacy in edge-based smart UAV delivery systems. The proposed framework combines the principles of local differential privacy and the shuffle model for FL. A key objective is to minimize communication costs between the edge server and the cloud aggregator, thereby improving the overall efficiency of the system. To demonstrate the practicality and effectiveness of the proposed FedShufde framework, this paper focuses on privacy protection in the context of a smart UAV delivery system. This system, as a complex and emerging edge-based IoT application, has attracted considerable attention from both academia and industry [27]. By applying FedShufde to the smart UAV delivery system, the paper demonstrates how the framework can be practically implemented to enhance privacy protection in real-world scenarios.

1.2. Contributions

To the best of our knowledge, FedShufde is the first privacy-preserving framework implemented in a real-world edge-based smart UAV delivery system. In this paper, we focus on the privacy-preserving of users' sensitive information. In addition, the energy consumption of the FedShufde framework is considered, as calculating the energy consumption is the most effective method to evaluate the effectiveness of the framework. The main contribution of the paper can be summarized as the follows:

- We introduce a novel privacy-preserving framework for edge computing, implemented in a smart UAV delivery system. Our framework leverages local differential privacy and the shuffle model to protect users' sensitive information from attackers.
- Our framework offers three distinct layers of privacy. First, user data is kept locally on devices, never shared directly for model training. Instead, model parameters from each device are aggregated at the central server, following the principles of FL, ensuring the raw data remains private. Second, we apply local differential privacy techniques by perturbing the model parameters before transmission. This guarantees that even if intercepted, the individual parameters do not reveal sensitive information. Third, the shuffle model randomizes the upload order of perturbed model parameters, preventing attackers from performing back-link attacks, that is, tracing specific updates back to their originating devices.
- We conduct extensive experiments on a real-world smart UAV delivery system, using publicly available datasets, to evaluate the performance of our framework. Our results demonstrate that the proposed framework significantly outperforms conventional methods in both privacy protection and system performance.
- We propose an innovative Energy-Constrained Expected Improvement (ECEI) acquisition function to enhance the parameter tuning process within Bayesian optimization. Integrating an energy constraint term into the traditional expected improvement function, our approach not only seeks optimal model parameters but also explicitly minimizes energy consumption. This energy-aware optimization is particularly suited for resource-constrained IoT environments and ensures an effective balance between model performance and energy efficiency.

1.3. Paper organization

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 outlines the foundational concepts necessary for understanding this paper, including local differential privacy, shuffle mode, and FL. Section 4 presents our proposed FedShufde framework in detail. Section 5 analyzes the privacy implications of our method, including amplification privacy and utility with respect to FedShufde. Section 6 presents the experimental results of our evaluation. Finally, Section 7 concludes the paper and suggests avenues for future research.

2. Related work

There are several research studies focusing on DP, including FL [20], data analysis [28], location privacy [29], and multi-item double auction mechanism [30]. DP method can be categorized into two types: *centralized* models and *local* models. In the centralized model, a trusted server receives raw data and analyzes it using privacy mechanisms. However, issues such as abuse, theft, and loss may still occur due to the requirement of gathering all raw data in one platform. In a local model, there is no trusted server to process privacy data. Instead, user raw data is perturbed using privacy-preserving methods before being sent to a server for analysis. Nevertheless, this method requires a large amount of data to conduct effective experiments, resulting in a significant trade-off between accuracy and privacy when choosing between the centralized and local models in DP research.

However, both centralized and local models have limitations in terms of accuracy and privacy. For centralized model, data needs to be uploaded to a central server, which may collect users' personal information. Once the server is attacked or the data is leaked, a large amount of personal privacy information will be exposed, thus affecting user's privacy. For local model, some mathematical errors may occur during processing on the local device, and these mathematical errors will be transmitted to the central server, which affects the accuracy in privacy measurement. The shuffle model, on the other hand, combines centralized and local models using mathematical methods to achieve high accuracy while maintaining privacy. In 2017, Bittau et al. [31] proposed the shuffle model, which consists of Encode, Shuffle, Analyze (ESA) and uses statistical and cryptography technology to reduce precision and analyze anonymous data. Erlingsson et al. [32] proved that random shuffling amplifies the DP guarantees of local randomized data. Feldman et al. [33] achieved asymptotically optimal dependence privacy budget with an algorithm that gets close to the optimal bound. Cheu [34] proposed shuffle protocols for statistical tasks such as binary sums, counting distinct elements, and histograms, which are robustly private and more accurate.

FL protects data privacy during large-scale data exchange. This is achieved by allowing each participant to train a local model using their own data, after the global model has been shared. In a recent study, Choudhury et al. [35] proposed an FL framework that enables the training of a global model using distributed data held locally at different sites. The effectiveness of FL has been demonstrated in various edge computing scenarios, such as urban traffic flow prediction [36] and UAV delivery services [37]. It is particularly relevant in the context of UAV delivery systems, where personal and highly sensitive information is abundant and privacy is of utmost importance.

FL also offers several advantages for smart IoT systems, including improved data privacy, low-latency network communication, and enhanced learning quality [38]. For example, Umair et al. [39] leveraged the computation power of IoT devices to enable distributed data learning using FL. They used a suggest-and-improve (SAI) approach based on the solution of the Lagrangian Dual problem followed by a local optimizer based on coordinate descent to allocate tasks. Song et al. [40] proposed a joint attack detection and defense solution implemented by FL, where each IoT device runs an AI model (such as Deep Neural Networks (DNN)) to detect and defend against threats. They trained a threat model against an adversary with the privacy-enhancing features of FL. Hao et al. [41] designed a privacy-enhanced FL algorithm for Industrial IoT (IIoT) that achieve example-level DP through a distributed Gaussian mechanism, protecting the privacy of local gradients and shared parameters. However, none of these approaches can guarantee that the attacker will not launch a back-link attack to obtain the link between the model parameters and the client, thus compromising the user's private information.

To prevent the leakage of sensitive information in deep neural networks, Wei et al. [23] proposed a novel framework based on the concept of DP. They inject noise into the model parameters of the client before aggregating. Similarly, Hu et al. [22] designed a privacy-preserving approach for creating effective personalized models using distributed user data, while ensuring DP of user data. Cao et al. [42] employed local differential privacy (LDP) to address the issue of high computational resources consuming in the FL framework. These studies show how the DP and LDP can provide a strong privacy protection for user data in a FL framework depending on various factors, such as the data distribution, noise level, and model aggregation methods.

In summary, the aforementioned proposals exhibit limitations in addressing the inherent trade-off between privacy and data utility, often resulting in reduced data utility or increased computational overhead. Furthermore, the efficiency and performance of the proposed methods tend to degrade when handling large volumes of data or accommodating a high number of users. These factors hinder the practical viability of the techniques in the context of smart UAV delivery systems. Additional challenges arise from practical considerations, including the need for compatibility with existing infrastructure, ensuring usability for end-users, and seamless integration with diverse platforms. These challenges necessitate further research and development to effectively balance privacy requirements with the operational demands and constraints of edge-based smart UAV delivery systems. To address these issues, in this paper, a novel FL framework with LDP and Shuffle mode is designed to provide a higher level of privacy protection without affecting the model training effect. Additionally, through the shuffle model, the model parameters of each participant are shuffled to break the correlation between the uploaded model parameters and the participant. The LDP is applied to local model updates for each participant to protect the privacy of their data. This leads to developing our proposed FedShufde framework, which has been discussed in Section 4.

3. Preliminaries

In this section, we briefly introduce some key concepts and models related to DP, the shuffle model, and FL. Table 1 provides a summary of important notations that are used throughout this paper.

3.1. Differential privacy

Differential Privacy (DP) is a technique used to protect the privacy of individuals, especially when publishing statistical data. It ensures that individual data cannot be identified by adding random noise to the query results so that the presence or absence of individual records has very little effect on the results. In contrast, Local Differential Privacy (LDP) provides stronger privacy protection and is suitable for decentralized data collection scenarios. In the LDP model, data is randomized by the individual before it is collected, meaning that the data collector receives already protected data and the individual does not need to trust the data collector. LDP ensures that each individual's data maintains a high level of privacy throughout the process by adding random noise directly to the data collection phase.

3.1.1. DP and LDP

Compared to centralized DP, LDP is more appropriate for situations where users do not trust third parties to protect their private data.

Table 1
Notations used in this paper and their explanation.

Notations	Explanation
$[n]$	$[n] = \{1, 2, \ldots, n\}$
$\mathcal{X}$	Data Universe
P	Protocol
$\vec{x} = \{x_1, x_2, \ldots, x_n\}$	A dataset $\vec{x} \in X^n$ is an ordered tuple of n rows
S_n	A symmetric group
D	Datasets
D'	The neighbor datasets with D
ε	Privacy budget.
δ	Relaxation term
$\mathcal{M}$	A randomized algorithm
$\mathcal{P}$	A protocol
$\mathcal{R}$	A randomizer
$\mathcal{A}$	An analyzer
$\mathcal{S}$	A shuffler
σ	A permutation
$\{\mathcal{F}_i\}_{i=1}^N$	N participants in FL framework
$\{D_i\}_{i=1}^N$	The dataset of participants in FL framework
θ	The parameter of model in FL framework

Definition 1 (*Differential Privacy*)**.**[43]: For two neighboring datasets D and D' differing only in one element, a randomized algorithm function $\mathcal{M}$ guarantees ε-differential privacy for any subset of the output $S \subseteq Range(\mathcal{M})$, if satisfies:

$$Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \times Pr\left[\mathcal{M}(D') \in S\right] + \delta \tag{1}$$

where $Range(\mathcal{M})$ is the range of resultant output function $\mathcal{M}$, ε is the privacy parameter, which controls the privacy degree of the mechanism. If $\delta = 0$, $\mathcal{M}$ is degenerated into ε-differential privacy.

Definition 2 ((ε, δ)-*Local Differential Privacy [44]*)**.** For any different elements $v, v' \in D$, a randomized algorithm $\mathcal{M}$ guarantees (ε, δ)-LDP $(\varepsilon, \delta > 0)$ for any subset of the output $S \subseteq Range(\mathcal{M})$, if satisfies:

$$Pr[\mathcal{M}(v) \in S] \leq e^{\varepsilon} \times Pr\left[\mathcal{M}(v') \in S\right] + \delta \tag{2}$$

where, $Range(\mathcal{M})$ is the range of resultant output $\mathcal{M}$. ε, δ are the privacy parameter. The privacy parameter controls the privacy degree of the mechanism. If $\delta = 0$, $\mathcal{M}$ satisfies ε-LDP.

3.1.2. Laplace and Gaussian mechanism

The Laplace mechanism is applicable to various types of data and the noise added by the mechanism obeys a continuous Laplace distribution $Lap(0, \frac{\Delta f}{\varepsilon})$. The Laplace function is $f(x|0, b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$, where b is a positive number, known as the scale parameter, which controls the 'breadth' of the distribution and can also be interpreted as controlling the degree of perturbation of the data.

Definition 3 (*Laplace Mechanism [45]*)**.** Giving a query function $f : D^n \to S$, where S is all possible outputs set. For any $\varepsilon > 0$ and Laplace distribution $Lap(0, \frac{\Delta f}{\varepsilon})$, Laplace mechanism is satisfied as:

$$M(D) = f(D) + Lap(0, \frac{\Delta f}{\varepsilon}). \tag{3}$$

In the DP Gaussian mechanism, if a function f has ε-DP, then the noise introduced in its result tends to obey a Gaussian distribution with mean 0 and standard deviation $\frac{\sigma}{\varepsilon}$, where σ is the sensitivity of the query result.

Definition 4 (*Gaussian Mechanism [46]*)**.** Giving a query function $f : D^n \to S$, where S is all possible outputs set. For any $\varepsilon > 0$ and a Gaussian distribution function $\mathcal{N}(0, \frac{\Delta^2 f}{\varepsilon^2})$, $\varepsilon-$Gaussian Mechanism is satisfied as:

$$M(D) = f(D) + N(0, \frac{\Delta^2 f}{\varepsilon^2}) \tag{4}$$

3.2. Shuffle model

To prevent an attacker from linking the output to its sender, a participant must trust a secure entity to perform a shuffle operation. The shuffle model is a random operation that is used in local model to prevent the output of one participant from being linked with another. Each participant executes $\mathcal{R}$ with their data to produce an output, which is then sent to the shuffler $\mathcal{S}$.

Definition 5 (*Shuffle Model [47,48]*)**.** Protocol $\mathcal{P}$ consists of three random algorithms: randomizer $\mathcal{P}$, shuffler $\mathcal{S}$ and analyzer $\mathcal{A}$:

A randomizer is mapping data to a vector of message, the randomizer denotes $\mathcal{R}$: $\mathcal{X} \to \mathcal{Y}^*$.

A shuffler is permuting messages uniformly random; the shuffler denotes $\mathcal{S}$: $\mathcal{Y}^* \to \mathcal{Y}^*$.

An analyzer is computing a permutation of messages, the analyzer denotes $\mathcal{A}$: $\mathcal{Y}^* \to \mathcal{Z}$.

As $\mathcal{S}$ is the same in every protocol, the execution by n users on input $\vec{x} \to \mathcal{X}^n$ can be defined as follow:

$$\mathcal{P}\left(\vec{x}\right) = \mathcal{A}(\mathcal{S}\left(\mathcal{R}\left(x_1\right), \mathcal{R}\left(x_2\right), \ldots, \mathcal{R}\left(x_n\right)\right)) \tag{5}$$

Then, we introduce a permutation method - Mallows model, which is a popular two-parameter permutation model based on distance for ranking data [49]. The set of permutations of $[n], n \in \mathbb{N}$ forms a symmetric group S_n. A permutation denoted $\sigma \in S_n$ to a data sequence $\vec{x}$ of length n. The value at ith in σ denoted $\sigma(i), i \in [n]$. For example, a permutation $\sigma = (2, 6, 4, 1, 3, 5)$, the value at third and fifth are $\sigma(3) = 4$, $\sigma(5) = 3$, a dataset is $\vec{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$, then $\sigma(\vec{x}) = (x_2, x_6, x_4, x_1, x_3, x_5)$.

Definition 6 (*Mallows Model [50]*)**.** For a dispersion parameter Θ, a reference permutation $\sigma_0 \in S_n$, and a rank distance measure is Mallows Model:

$$\begin{aligned} &\partial : S_n \times S_n \mapsto \mathbb{R} \\ &\mathbb{P}_{\Theta,\partial}\left(\sigma : \sigma_0\right) = \frac{1}{\psi(\Theta, \partial)} e^{-\theta \circ (\sigma, \sigma_0)} \end{aligned} \tag{6}$$

where $\psi(\Theta, \partial) = \sum_{\sigma \in S_n} e^{-\theta \circ (\sigma, \sigma_0)}$ is a normalization term and $\sigma \in S_n$.

In this paper, we chose the Mallows model as it provides an effective trade-off between randomness and coherence, ensuring that the permutations are diverse yet retain some meaningful structure. The concentration parameter allows us to control how dispersed the permutations are, which is crucial for balancing privacy and utility in FL settings.

3.3. Federated learning

FL offers several key features, including [51]:

- FL involves multiple participants who collaborate to build a shared machine learning model using their own local training datasets.
- During the FL training process, each participant's training datasets remains local and is not shared with others.
- The FL model's relevant information is transmitted and exchanged among participants using encryption, while each participant's raw training datasets remains private and cannot be inferred by others.
- The FL model's performance is comparable to that of an ideal model trained on the combined datasets of all participants.

In general, FL involves N participants $\{\mathcal{F}_i\}_{i=1}^N$, each holding their dataset $\{D_i\}_{i=1}^N$, and a central aggregator (server). The FL process contains two phases: *model training* and *model inference*. In the training model phase, participants can exchange relevant information to

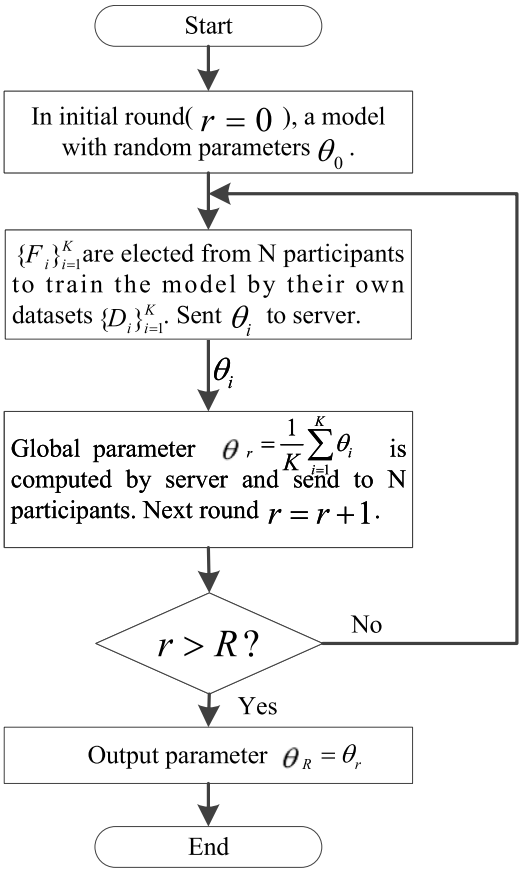


Fig. 1. The flow chart of the FL model used for our FedShufde framework.

improve the model beyond what can be achieved through individual training. In the model inference phase, the trained model is applied to new data instance. Fig. 1 illustrates the flow chart of the FL model that we used in this paper.

Firstly, the server generates a model with random parameter θ_0 in initial round ($r = 0$). Secondly, K participants are elected from N participants $\{\mathcal{F}_i\}_{i=1}^K$ to train the model by their own data $\{\mathcal{F}_i\}_{i=1}^K$ in the local, and the model parameter is sent to the server in each round r. Thirdly, the server computed the global parameter $\theta_r = \frac{1}{K}\sum_{i=1}^K \theta_i$ and sent to N participants. Finally, the train process ends and the final parameter θ_R is output until the number of rounds r reaches the set R.

4. Proposed FedShufde framework

In this section, we provide an overview of the proposed FedShufde framework. The framework is used in the edge-based smart UAV delivery system as a typical application scenario.

4.1. Framework overview

As shown in Fig. 2, the proposed framework is composed of three layers. From bottom to top, they are: end device layer, edge computing layer, and cloud computing layer. The end device layer consists of devices such as UAVs and other IoT devices that collect the raw data and communicate directly with the users. The edge computing layer comprises edge servers (such as UAV stations or 5G stations) that communicate directly with end devices and provide computing resources required for latency sensitive services and data privacy protection operations (such as LDP and Shuffling). The cloud computing layer composed with cloud servers that communicate directly with the edge servers and hosts the business system. It also provides centralized data

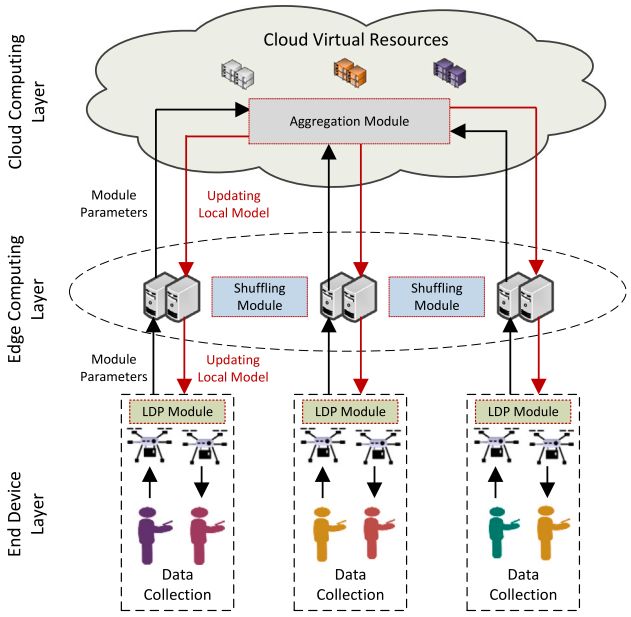


Fig. 2. An illustration of the proposed FedShufde framework for edge-based UAV delivery system.

storage and computing resources for latency non-sensitive and offline services, including the training of global machine learning models (such as the aggregation module).

Before transmitting the local model updates to the central server located in the cloud computing layer, each client, including end devices connected to the edge server, implements DP mechanism. This involves locally injecting a specified amount of noise into the model updates. Therefore, the privacy of user data is enhanced without significantly compromising the performance of the model.

4.2. System functionality

In our FedShufde framework, the process of protecting data privacy can be divided into five major phases.

(i) Data Collection and Local Model Training: User data such as face recognition, target detection, and route planning are first collected by UAVs and other end devices. The data is used to train machine learning models, a process known as local model training.

(ii) Local Differential Privacy Perturbing: After training, the local model parameters undergo LDP perturbing at the local (end device), where noise is added to the parameters. This step aims to enhance data privacy while maintaining utility, preventing the original data from being inferred from the model parameters.

(iii) Model Parameter Shuffling: Before the processed local model parameters are uploaded to the cloud server (the aggregator), they are shuffled by the shuffler model. Shuffling further enhances privacy protection by scrambling the order of the parameters, which reduces potential privacy leaks during aggregation (more discussion in Section 4.3).

(iv) Global Model Aggregating: The cloud server receives the shuffler model parameters and aggregates them to construct a global model. This aggregation is based on the model parameters provided by multiple edge servers, aiming to produce a more performant and generalized machine learning model.

(v) Model updating: After aggregation, the global model parameters are sent back to local device to update the local models. Each local model learns broader features and patterns from the global model, maintaining the periodicity and real-time nature of the updates.

Algorithm 1: FedShufde System Functionality

Input : Edge server's weight ω, epoch E, noise size $\varkappa$, the number of edge servers C, learning rate η, clipping size S, batch size B

Output : Parameters $\hat{\theta}_k$ after aggregation

1 **Edge Server:**
2 $\theta \leftarrow \theta^t$
3 **for** $e \in E$ **do**
4 **for** $b \in B$ **do**
5 compute gradient
6 $g_b(x_i) \leftarrow \nabla l(\theta; b)$
7 gradients clip
8 $\hat{g}_e(x_i) \leftarrow g_b(x_i) \min(1, \frac{S}{||g_e(x_i)||_2})$
9 $\hat{g}_e \leftarrow \hat{g}_e(x_i) + N(0, I\sigma^2)$
10 $\theta^{t+1} \leftarrow \theta_t - \eta(\hat{g}_e)$
11 **return** θ_k
12 $Shuffle \quad S$
13 Send θ^t to Cloud Server for aggregated parameters
14 **Cloud Server:**
15 $c \leftarrow$ sample participant of random select
16 Initialize model θ^0
17 $W = \sum w_k$
18 **for** $e \in E$ **do**
19 **for** $c \in C$ **do**
20 θ_t is parameters after aggregation
21 $\theta_k \leftarrow \quad EdgeServer(c, \theta^t)$
22 $\hat{\theta}_k \leftarrow w_k \theta_k$
23 $\theta^t \leftarrow \frac{\hat{\theta}_k}{W}$
24 Send $\hat{\theta}_k$ to Edge Server
25 **return** $\hat{\theta}_k$

Algorithm 1 provides an overview of the system functionality. It consists of two stages. In the first stage, the model parameters perturbation stage, we incorporate LDP into the FL framework. The method named DP stochastic gradient descent (DP-SGD) [52,53] is used to train the model with the data collected by end devices. The privacy budget can be allocated using the moment accountant. Lines 1 to 11 are used to train the model at the edge server (participant/client). Line 12 is the shuffler for the shuffle model. Finally, lines 13 to 25 are used to send the model parameters to the cloud server for aggregation of perturbation parameters.

To be more precise, we begin by performing gradient clipping at the edge server. This involves limiting the gradient values to a range of c in order to reduce their sensitivity. In generally speaking, the gradient vectors are difficult to obtain a prior sensitivity S such that $\|w_k\theta^k\| \leq S$, where w_k is weight of gradients. Therefore, the gradients need to be clipped before the phase of adding noise. In the stage of gradient clipping, we use L_2 projection as follows:

$$\pi(\Delta, S) \triangleq \Delta min(1, \frac{S}{\|\Delta\|}) \tag{7}$$

Then, we suppose that the number of gradient vectors is m, $\Delta_k = (\Delta_k(1), \ldots, \Delta_k(m))$. For any clipping parameter S, the gradient is $\Delta'_k = \pi(\Delta_k, S)$.

Further, FedShufde adds noise to the gradient and shuffles the gradient sequence. At the edge server, Gaussian noise is added to the clipping gradient. Then, the edge server parameters are updated with the noised gradient, and the updated parameters are sent back to the cloud serve to complete the aggregation operation. Finally, the perturbed gradients are transmitted to the aggregator in the cloud server.

4.3. Shuffle model mechanism

Table 2 illustrates mechanism, implementation methods, and effects of the shuffle model. Additionally, the randomizer can be categorized into single-message pattern and multi-message pattern based on the size of output messages, resulting in two types of shufflers: single mixer mode and multi mixer mode.

This section illustrates the parameter shuffling process of five models M_i, $(i = 1, \ldots, 5)$, as shown in Fig. 3, where each model's parameters are x_i, y_i, z_i, $(i = 1, \ldots, 5)$. The reference sequence of model parameters is defined as $\sigma_{0x} = (3, 5, 2, 1, 4)$, $\sigma_{0y} = (4, 1, 5, 3, 2)$, $\sigma_{0z} = (2, 4, 3, 5, 1)$ and the parameters of the new model M'_i, $(i = 1, \ldots, 5)$ are shuffled. M'_i is uploaded to the aggregator for aggregation. In the FedShufde framework, we use a multi-shuffler model for parameter shuffling. The model parameters are classified, and multiple shufflers shuffle the classified parameters. The advantage of using a multi-shuffler model is that it has higher security and is less susceptible to attacks.

In the parameter collaborative shuffling algorithm proposed in this section, the application of the Mallows model in the shuffling mechanism primarily appears during the data shuffling phase, where it serves as a guideline for the shuffling sequence of the shuffler. The Mallows model generates a randomly ordered reference permutation, which is used for shuffling the input data in the shuffler. Inspired by the use of the Mallows model and shuffling mechanisms in the literature [47,50], we define *Definition 7 Mallows Permutation Benchmark* and *Definition 8 Parameter Collaborative Shuffling Mechanism*, as follows:

Definition 7 (*Mallows Permutation Benchmark*)**.** Suppose each model has k parameter sets, denoted as $p_1, p_2, \ldots, p_k$, where each parameter set p_j $(j = 1, 2, \ldots, k)$ has a corresponding reference permutation σ_{0p_j} and dispersion parameter θ_{p_j}. The permutation σ_{p_j} for each parameter set p_j is generated according to the Mallows model, with a probability distribution given by:

$$P(\sigma_{p_j}|\sigma_{0p_j}, \theta_{p_j}) = \frac{1}{\psi(\theta_{p_j}, d_{p_j})} \exp\left(-\theta_{p_j} d(\sigma_{p_j}, \sigma_{0p_j})\right) \tag{8}$$

where $d(\sigma_{p_j}, \sigma_{0p_j})$ is the distance measure between permutation σ_{p_j} and the reference permutation σ_{0p_j}, and $\psi(\theta_{p_j}, d_{p_j})$ is a normalization constant ensuring that the probability distribution is properly normalized.

Definition 8 (*Parameter Collaborative Shuffling Mechanism*)**.** For each model M_i $(i = 1, 2, \ldots, n)$, the new model M'_i is generated as follows:

$$M'_i = \left(p_{1_{\sigma_{p_1}(i)}}, p_{2_{\sigma_{p_2}(i)}}, \ldots, p_{k_{\sigma_{p_k}(i)}}\right) \tag{9}$$

Since the permutations of each parameter set are not independent, the joint probability distribution can be expressed as the product of the individual Mallows permutation benchmarks. For k parameter sets, the joint probability distribution is given by:

$$\begin{aligned} P\left(\{\sigma_{p_j}\}_{j=1}^k\right) &= \prod_{j=1}^k P(\sigma_{p_j}|\sigma_{0p_j}, \theta_{p_j}) \\ &= \prod_{j=1}^k \frac{1}{\psi(\theta_{p_j}, d_{p_j})} \exp\left(-\theta_{p_j} d(\sigma_{p_j}, \sigma_{0p_j})\right) \end{aligned} \tag{10}$$

where each parameter set p_j is rearranged according to its generated permutation σ_{p_j}, and the newly combined parameter sets form the new model M'_i.

To increase the randomness of the sequence shuffling, we have designed a new shuffle model based on the Mallows model. The details of the shuffle model are shown in Algorithm 2. Specifically, Line 1 to 8 is the process for sequence partition, and Line 9 to 20 is to output sequence through shuffling by the reference sequence.

4.4. Energy consumption of FedShufde

In this paper, we also discuss the energy consumption problem of the FedShufde framework, which is a key performance metric for edge computing-based IoT systems [27]. The energy consumption of the

Table 2
The summary of Randomizer, Shuffler, and Analyzer.

Items	Devices		Implementation	Purpose
	End devices	Reliability		
Randomizer	Local Client	Credible	Data generalization, Data segmentation, Encryption, Adding DP noise	Privacy information is reduced or eliminated
Shuffler	Server	Semi-honest	Secure shuffle protocol, Homomorphic encryption, Secure multi-party computation	Anonymity and protection data size privacy
Analyzer	Data Collection Center	Not Credible	Analysis and correction by statistic methods	Reception the shuffle data

Algorithm 2: Shuffler $\mathcal{S}$

Input : LDP sequence $\overline{y} \leftarrow (y_1, ..., y_n)$, reference sequence $\sigma \leftarrow (\sigma_0, \sigma_1, ...\sigma_9)$;
Output : shuffled output sequence $\overline{x} \leftarrow (x_1, ..., x_n)$

$K = log_{|\sigma_i|} y$;
Unit a K dimension array M^k;
$M^k \leftarrow \overline{y}$;
for $k \in K$ **do**
  $M^{k-1} \leftarrow shuffle(M^K, \sigma_i)$;
end
$\overline{x} \leftarrow M^k$;
return $\overline{x}$; ;
$Shuffle(M^K, \sigma_i)$;
for i **do**
  for j **do**
    ...
    for k **do**
      $Shuffle\ with\ \sigma_i$;
      $M^k \leftarrow swap(M^k, \sigma_i)$
      end
    end
  end
return lower dimension M;
return M^{k-1};

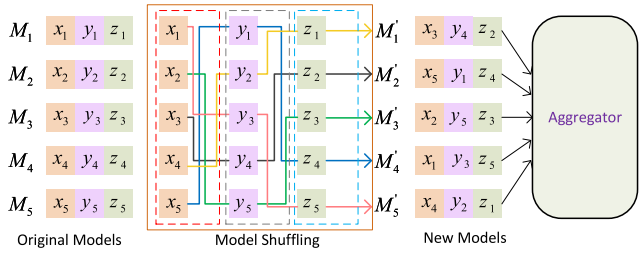


Fig. 3. The schematic diagram for the Shuffle model.

FedShufde framework can be roughly divided into two main parts: one is the *model training* energy consumption, and the other is the *model aggregation* energy consumption. These two parts represent different stages of the FL cycle, each making a unique contribution to the overall system energy consumption.

First, the energy consumption of model training, denoted as E^{ES}, refers to the energy consumed by the computing resources of edge servers during the training phase. The energy consumption can be formally defined as:

$$E^{ES} = \sum_{i=1}^{C} \left(\alpha_i \cdot (P_i^E)^{\beta_i} + \gamma_i \cdot (t_i^E)^{\delta_i} + \eta_i P_i^E t_i^E \right) \tag{11}$$

where P_i^E represents the power consumption of the ith edge server, and t_i^E represents the time required for model training by the ith edge server. The parameters α_i, β_i, γ_i, δ_i, and η_i are used to capture the nonlinear relationships and interaction effects between power and time, reflecting the complexity and variability of power consumption due to differences in hardware capability or workload intensity of each edge server, as detailed in Table 3.

Similarly, the energy consumption for model aggregation, denoted as E^{CS}, refers to the energy consumed by the cloud server during the aggregation phase. This energy consumption can be expressed as:

$$E^{CS} = \lambda \cdot (P^{CS})^{\mu} + \nu \cdot (T^{CS})^{\omega} + \kappa P^{CS} T^{CS} \tag{12}$$

where P^{CS} and T^{CS} represent the power consumption and the time required for model aggregation by the cloud server, respectively. The parameters λ, μ, ν, ω, and κ represent the nonlinear factors that describe the complex dependencies between power and time during the aggregation process, as detailed in Table 3.

Therefore, the total energy consumption E of the FedShufde framework can be defined as the sum of the energy consumed during model training and aggregation:

$$E = E^{ES} + E^{CS} \tag{13}$$

To effectively obtain the values of these parameters, we adopt an efficient approach based on Bayesian optimization. Bayesian optimization is a technique suitable for optimizing complex black-box functions, especially for parameter tuning problems in FL frameworks [54,55]. In this method, we first build a prior distribution of the parameters based on a small amount of initial experimental data, which can be obtained by collecting power consumption and training time data under different experimental conditions.

After obtaining the initial data, we use a Gaussian Process (GP) as the prior model to capture the complex relationships in the parameter space. The advantage of GP is that they can not only predict the value of the target function but also provide an uncertainty estimate for each prediction, enabling bayesian optimization to effectively balance exploration and exploitation. In each iteration, bayesian optimization selects the next experimental point by calculating the acquisition function (such as, Expected Improvement, EI) to maximize the improvement of the target function.

To further enhance the effectiveness of Bayesian optimization in energy efficiency parameter tuning in the FedShufde framework, we propose an improved acquisition function the ECEI. The traditional expected improvement function primarily considers the improvement in function value [56], whereas we add an energy constraint term to ensure that the optimization process not only finds the optimal parameters but also minimizes energy consumption as much as possible.

The improved acquisition function $A_{ECEI}(x)$ is defined as:

$$A_{ECEI}(x) = \mathbb{E}[\max(0, f(x) - f(x^+))] - \lambda_E \cdot E(x) \tag{14}$$

where $f(x^+)$ is the current best value of the function, $E(x)$ represents the energy consumption under the current parameter setting, and λ_E is a hyperparameter controlling the weight of the energy constraint. By maximizing $A_{ECEI}(x)$, we can significantly reduce energy consumption while ensuring model performance.

The Gaussian Process model in Bayesian optimization can be represented as:

$$f(x) \sim GP(\mu(x), k(x, x')) \tag{15}$$

where $\mu(x)$ is the mean function, and $k(x, x')$ is the covariance function (or kernel function) used to describe the similarity between two input

Table 3
Parameters and their names and descriptions.

Parameter	Name	Description
α_i	Nonlinear power influence factor	Reflects the basic part of edge server power consumption
β_i	Power exponent	Represents the degree of nonlinearity of power consumption with varying load
γ_i	Nonlinear training time influence factor	Reflects the weight of training time in energy consumption
δ_i	Training time exponent	Describes the nonlinear influence of training time on energy consumption
η_i	Coupling coefficient between power and training time	Reflects the interaction between power and training time on total energy consumption
λ	Nonlinear power influence factor for cloud server	Reflects the basic part of cloud server power consumption
μ	Power exponent for cloud server	Describes the nonlinearity of power consumption with varying load
ν	Nonlinear aggregation time influence factor	Reflects the weight of aggregation time in energy consumption
ω	Aggregation time exponent	Describes the nonlinear influence of aggregation time on energy consumption
κ	Coupling coefficient between power and aggregation time	Reflects the interaction between power and aggregation time on total energy consumption

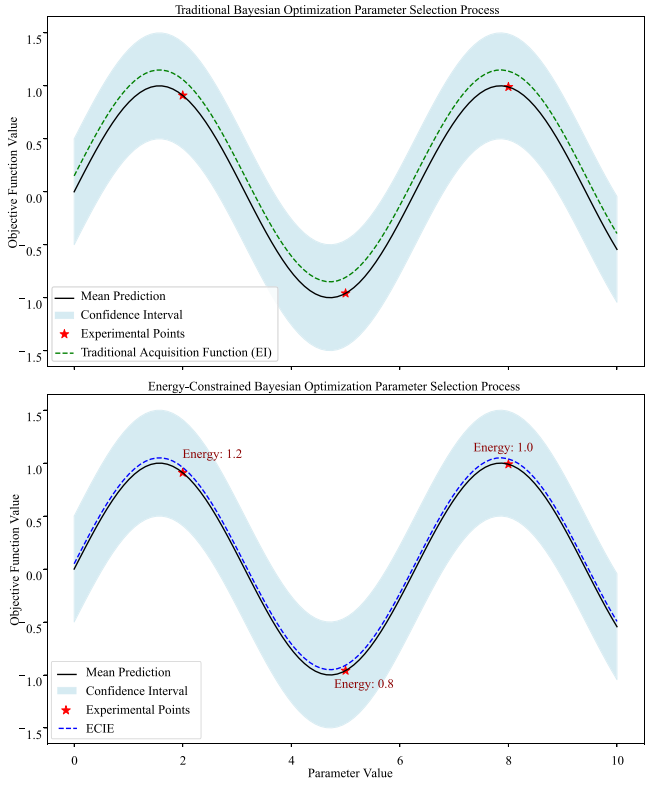


Fig. 4. Comparison of the parameter selection process between traditional Bayesian optimization and energy-constrained Bayesian optimization.

points. In each iteration, Bayesian optimization selects the next experimental point by maximizing the acquisition function. The acquisition function $A_{ECEI}(x)$ can effectively balance exploration, exploitation, and minimization of energy consumption, thereby performing better in complex energy optimization scenarios. Fig. 4 demonstrates the comparison between the traditional Bayesian optimization method and the energy-constrained Bayesian optimization method in the parameter selection process. The upper figure demonstrates the mean prediction, confidence interval and acquisition function (EI) in the traditional Bayesian optimization process, where the selection of the experimental points is mainly based on the expectation of the improvement of the objective function, and therefore more inclined to explore in the region of greater uncertainty. Whereas, the following figure demonstrates the optimization process after the introduction of energy constraints, in which the experimental points are selected through the energy constrained collection function, that is ECEI. In this approach, in addition to considering the improvement of the objective function, constraints on the energy consumption are also added, thus enabling both the improvement of the model performance and the effective reduction of energy consumption in the parameter selection. This approach is especially suitable for energy-constrained IoT scenarios. Comparing the two approaches, the significant impact of introducing energy constraints on the optimization process can be clearly seen, which helps to achieve efficient parameter tuning in energy-constrained environments.

Through this iterative process, Bayesian optimization gradually converges to the optimal values of the parameters. In each iteration, the newly acquired data is used to update the Gaussian Process model to improve understanding of the parameter space, thereby selecting more suitable experimental conditions to gain more information. Compared to traditional grid search or random search, Bayesian optimization can find near-global optimal parameter settings faster under limited experimental runs, thus significantly reducing experimental costs.

Through the above method, we finally obtained the optimal parameter set for describing the energy consumption model of the FedShufde framework. These parameter values can not only accurately describe the power consumption behavior of edge and cloud servers but also show high adaptability under different workloads and hardware conditions, thereby improving the accuracy and reliability of the overall system energy consumption modeling.

4.5. Threat model

In this section, we discuss the threat model for the proposed FedShufde framework in the context of a UAV delivery system, identifying potential adversaries and their capabilities.

4.5.1. Adversarial capabilities and attack vectors

We consider both honest-but-curious and malicious adversaries within the system:

(i) *Honest-but-curious Adversaries*: These adversaries follow the protocol but attempt to learn private information from the available data. (ii) *Malicious Adversaries:* These adversaries actively seek to manipulate the system or data to gain an advantage or cause disruption. We also consider collusion among adversaries, significantly increasing their capabilities.

The attack vectors include:

- *Model Inversion Attacks*: These attacks involve adversaries attempting to reconstruct sensitive input data by exploiting the gradients or model parameters shared during the FL process. Adversaries use the partial information present in the model to reverse-engineer details about the underlying training data. In the context of UAV delivery, attackers may try to infer user location data, item descriptions, or even behavior patterns from the shared model information. This poses a significant risk to user privacy since it allows attackers to extract sensitive information that was never explicitly shared.
- *Membership Inference Attacks*: These attacks allow adversaries to determine whether a specific data point was used in the training process. In this attack, the adversary examines a trained model's behavior to infer if a given record was part of the training dataset. This is particularly concerning for sensitive datasets, as it can reveal information about individuals participating in the training process. For instance, in a UAV delivery system, attackers could identify if a user's data contributed to the training, potentially leading to privacy violations, such as identifying purchase behaviors or delivery patterns.
- *Poisoning Attacks*: These attacks aim to compromise the integrity of the learning process by injecting malicious data or modifying model parameters. These attacks can be categorized into two types: (i) *Dataset Poisoning:* Attackers add carefully crafted, malicious data to the training dataset to alter the final model's behavior. For example, attackers may add images with noise or incorrect labels to the training dataset, causing the model to produce erroneous predictions. In a UAV delivery system, this could lead to incorrect package routing or misclassification of items. (ii) *Model Poisoning*: Attackers can manipulate model parameters during local training before they are aggregated. This type of poisoning targets the shared model to introduce biases or backdoors that alter its predictions. For instance, an attacker could modify weights in a neural network to cause the model to misclassify specific inputs. In the context of UAVs, this could lead to unintended behaviors, such as misidentifying delivery locations or modifying flight paths to cause operational disruptions.

4.5.2. Assumptions about trust boundaries

We define the trust boundaries among entities in the UAV delivery system:

- *Edge Devices:* UAVs and other edge devices are considered partially trusted but are susceptible to physical capture or software manipulation by adversaries.
- *Edge Servers:* These are assumed to have a certain level of trust but may collude with other entities to infer sensitive information.
- *Cloud Server:* Acts as an aggregator in the FL process and is assumed to be honest-but-curious, meaning it follows the protocol but attempts to learn sensitive information.

4.5.3. Quantitative security goals

We define the following quantitative security goals:

- *Information Leakage Bound:* To quantify the upper bounds on information leakage through model updates, ensuring adversaries cannot infer more than a specified amount of data. The leakage bound is crucial for understanding the worst-case privacy risk posed by sharing model updates, ensuring that even with significant adversarial effort, the privacy loss remains limited.
- *Privacy Metrics:* The DP techniques are used to limit information leakage, with metrics such as ε defining the system's privacy guarantees. The privacy parameter ε quantifies the degree of privacy provided, with smaller values indicating stronger privacy protection, ensuring that the model outputs are statistically similar whether or not any individual user's data is included.
- *Resilience to Collusion:* To quantify the system's resilience to collusion based on the number of participants needed to degrade privacy guarantees significantly. By determining how many parties need to collaborate to break the privacy barrier, we measure the robustness of the system against coordinated adversarial efforts, providing insights into the practicality of executing such attacks.

4.5.4. Malicious attacker targets and collusion threats

In a UAV delivery system, malicious attackers may aim to obtain logistics and delivery information, exposing users' private details such as addresses, phone numbers, and purchasing information. This could also lead to disruption in the delivery process. Attackers may be interested in analyzing user behaviors, purchasing habits, and movement patterns, gaining insights into private activities. The knowledge of delivery addresses, item types, and delivery schedules can also be exploited to predict users' lifestyle patterns or their economic status.

Collusion between servers and users, or between servers themselves, may significantly weaken privacy guarantees, allowing adversaries to infer otherwise protected sensitive information. Colluding entities may combine their insights to break privacy protections that are otherwise effective for individual adversaries, resulting in more successful attacks and compromising the overall system security.

5. Privacy analyses

In most privacy protection frameworks, there is a trade-off between privacy and utility to achieve optimal privacy protection and data utility. Therefore, in this section, we present the privacy analyses of the FedShufde framework, which is based on privacy amplification and utility.

5.1. Privacy amplification

To enhance the effectiveness of privacy preservation, a privacy amplification theory analysis is utilized. In the shuffle model, user data is perturbed by a randomizer that satisfies ε-LDP on the local device. The shuffler then provides the analyzer with shuffle data that satisfies ε'-DP. Typically, ε is smaller than ε'. As the users' relationship is not correlated, we apply the non-interactive mechanism privacy amplification theorem in this paper.

Theorem 1 (*Non-Interactive Mechanism Privacy Amplification*)**.**: *Let each record $x_i (i = 1, 2, \ldots, n)$ for n users be operated by random encode protocol in local randomizer $\mathcal{R}$. For $\forall n \in \mathbb{N}_+$, if $\mathcal{R}$ satisfies ε-LDP ($\varepsilon \in (0, \frac{1}{2} ln(\frac{n}{ln(1/\delta)})]$) then n outputs of protocol $\mathcal{SR}$ are satisfy (ε', δ)-DP, where*

$$\varepsilon' = O((e^{\varepsilon} - 1)\sqrt{\frac{\ln(1/\delta)}{n}}) \tag{16}$$

Proof. Let x_i be the record of user i, where $i = 1, 2, \ldots, n$, and let $\mathcal{R}$ be the local randomizer that operates on x_i to produce a randomized

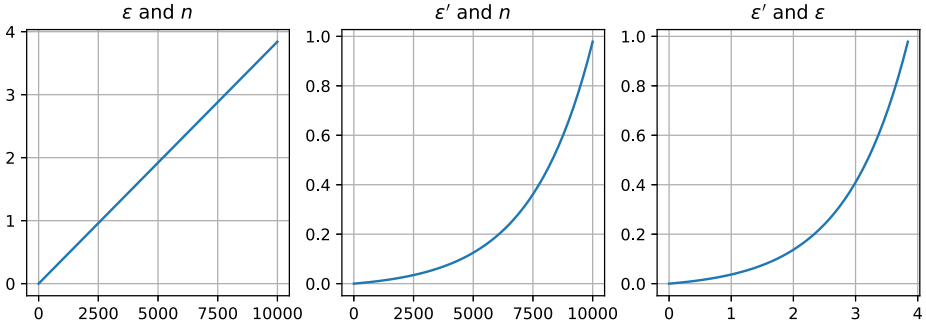


Fig. 5. The effect of amplification privacy.

output $y_i = \mathcal{R}(x_i)$. We assume that $\mathcal{R}$ satisfies ε-LDP, meaning that for any two records x_i, x_i' and any possible output y_i, we have:

$$\frac{P(\mathcal{R}(x_i) = y_i)}{P(\mathcal{R}(x_i') = y_i)} \leq e^{\varepsilon} \tag{17}$$

Consider the protocol $\mathcal{SR}$, which consists of n users each applying the local randomizer $\mathcal{R}$ independently to their own data. We denote the output of this protocol by $Y = (y_1, y_2, \ldots, y_n)$, where each y_i is the output of $\mathcal{R}(x_i)$.

To prove that the combined output Y satisfies (ε', δ)-DP, we use the concept of privacy amplification by the number of users. The key idea is that when multiple users contribute their data independently, the overall privacy loss can be reduced due to the aggregation of noise introduced by each local randomizer.

Step 1: Characterizing the Privacy Loss

The privacy loss for each individual user i is characterized by the random variable:

$$L_i = \ln\left(\frac{P(\mathcal{R}(x_i) = y_i)}{P(\mathcal{R}(x_i') = y_i)}\right) \tag{18}$$

Since $\mathcal{R}$ satisfies ε-LDP, we have $|L_i| \leq \varepsilon$ for all i. The total privacy loss for the protocol $\mathcal{SR}$ is given by the sum of the privacy losses of all users:

$$L = \sum_{i=1}^{n} L_i \tag{19}$$

Step 2: Bounding the Privacy Loss

To derive a bound on the total privacy loss L, we use the fact that the L_i are independent random variables with bounded range. By applying a concentration inequality, such as Hoeffding's inequality, we can bound the probability that the total privacy loss exceeds a certain threshold.

Specifically, for any $t > 0$, Hoeffding's inequality gives:

$$P(L > t) \leq \exp\left(-\frac{2t^2}{n(2\varepsilon)^2}\right) = \exp\left(-\frac{t^2}{2n\varepsilon^2}\right) \tag{20}$$

We want to ensure that the total privacy loss L is bounded by ε' with probability at least $1 - \delta$. Therefore, we set:

$$\exp\left(-\frac{(\varepsilon')^2}{2n\varepsilon^2}\right) = \delta \tag{21}$$

Solving for ε', we obtain:

$$\varepsilon' = \sqrt{2n\varepsilon^2 \ln\left(\frac{1}{\delta}\right)} \tag{22}$$

Step 3: Final Bound

To simplify the expression for ε', we use the fact that for small ε, $e^{\varepsilon} - 1 \approx \varepsilon$. Thus, we can approximate:

$$\varepsilon' = O\left((e^{\varepsilon} - 1)\sqrt{\frac{\ln(1/\delta)}{n}}\right) \tag{23}$$

This completes the proof, showing that the combined output of the protocol $\mathcal{SR}$ satisfies (ε', δ)-DP, where ε' is given by the stated bound. The result demonstrates the privacy amplification effect, where the effective privacy loss per user decreases as the number of users n increases.

We compare variation trend of ε and ε' under the number of users $n = 10000$, $\delta = 0.001$. As shown in Fig. 5, both ε and ε' increase as n increases, but the growth rate of ε' is noticeably slower than that ε.

5.2. Utility

In this section, we propose a metric named (α, β)-preservation to evaluate the utility of shuffle model. This metric is independent of both the data distribution and the analyst's query, as state in [50].

Theorem 2 ((α, β)-*preservation*)**.**: *Let $S \subseteq [n]$ be a set of individuals from $\mathcal{Y}$, then a shuffle mechanism provided (α, β)–preservation to S, if*

$$Pr[|S_\sigma \cap S| \geq \alpha \cdot |S|] \geq 1 - \beta \tag{24}$$

where $\sigma \in S_n$, $S_\sigma = \{\sigma(i) | i \in S\}$.

(1) For $\alpha = 0$, $\beta = 1$, we can infer at least 0% of its data values corresponding to the subset S overlapping with that of shuffled sequence $\sigma(\mathcal{Y})$ with probability 0.

(2) For $\alpha = 1$, $\beta = 0$, we can infer at least 100% of its data values corresponding to the subset S overlapping with that of shuffled sequence $\sigma(\mathcal{Y})$ with probability 1.

Proof. for $Pr[|S_\sigma \cap S| \geq \alpha \cdot |S|] \geq 1 - \beta$, we have

$$Pr[\alpha \leq \frac{|S_\sigma \cap S|}{|S|}] \geq 1 - \beta \tag{25}$$

(1) when $\alpha = 0$, then $|S_\sigma \cap S|$ which is $S_\sigma \cap S = \varnothing$.

If a sequence is output by shuffle mechanism without elements overlap with before, so $\beta = 1$.

(2) when $\alpha = 1$, then $|S_\sigma \cap S| = |S|$ which is $S_\sigma = S$.

If a sequence is output by shuffle mechanism without elements overlap with before, so $\beta = 0$.

6. Implementation and evaluation

In this section, we provide a implementation and evaluation of results of the proposed FedShufde framework using four datasets Fashion-Mnist, Mnist, Med-Mnist, and Box (self-constructed courier carton dataset).

6.1. Implementation setup

Now we discuss experimental environments, datasets description, and dataset settings.

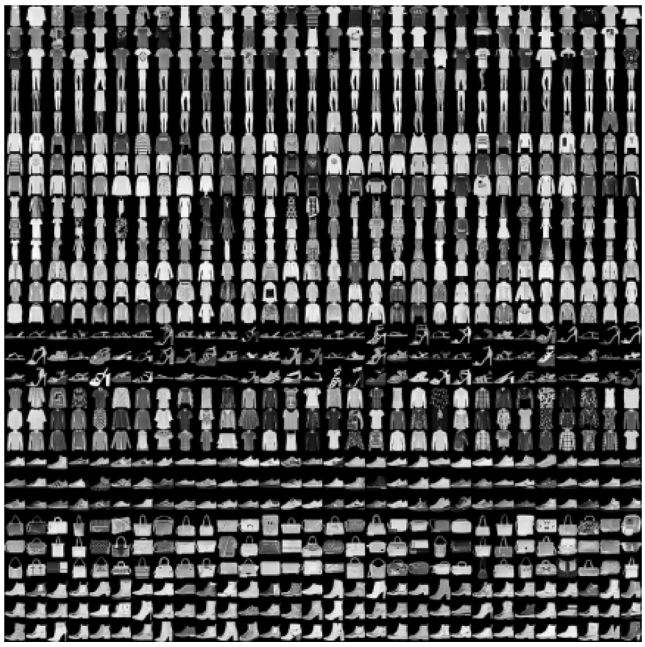


Fig. 6. Fashion-Mnist datasets example.

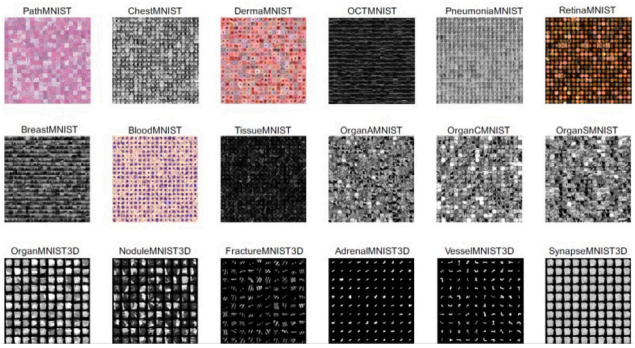


Fig. 7. Med-Mnist datasets example.

(a) Broken box datasets example.

(b) Intact box datasets example.

Fig. 8. Box datasets examples (collected by us).

6.1.1. Experimental environments

The experiments were conducted using Python 3.80 as the software environment, and opacus 0.13.0 in Pytorch was used to calculate the privacy budget. The hardware specifications for the computer user for the experiments were CPU i7-12700K, 3.61 GHz, and 64 GB of RAM. In the experiments in this paper, we use the standard energy consumption data of the Intel® Core™ i7-12700K processor to calculate the energy consumption of the edge devices. Specifically, the Processor Base Power of this processor is 125 W, while the Maximum Turbo Power is 190 W. Based on this, we consider the power consumption of the processor under different loads separately when calculating the energy consumption to ensure the accuracy and representativeness of the energy consumption calculation.

To determine the optimal privacy budget $[0.1, 2]$ for our FedShuffle framework, we follow a systematic approach to balance privacy protection and model utility. The privacy budget ϵ quantifies the trade-off between privacy and performance. Smaller values of ϵ provide stronger privacy but may degrade model performance, while larger values improve performance but weaken privacy guarantees. In our framework, the privacy budget ϵ is calculated as:

$$\epsilon = \frac{E \cdot C}{\sigma} \cdot \sqrt{2 \log\left(\frac{1.25}{\delta}\right)} \tag{26}$$

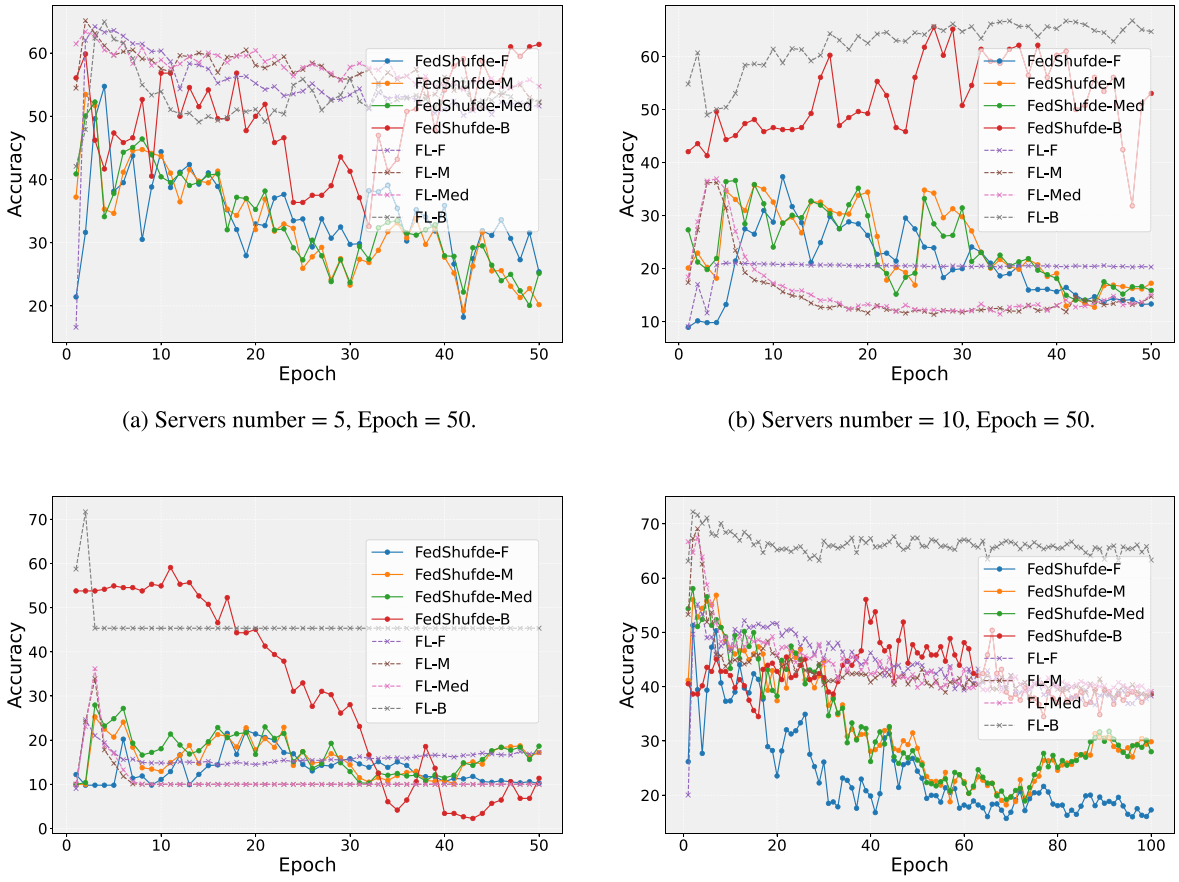
where E is the number of epochs, C is the number of edge servers participating per round, σ is the standard deviation of the Gaussian noise added to gradients, and δ is the failure probability, set to 0.001. For our experiments, we use 10 epochs and 10 participants per round. We vary the noise standard deviation (σ) with values 5, 10, and 20 to examine the effect of noise on both privacy and utility.

Using Eq. (26), the corresponding privacy budgets are calculated as follows. With $\sigma = 5$, the privacy budget ϵ is approximately 9.7; for $\sigma = 10$, it reduces to 4.8; and with $\sigma = 20$, it becomes 2.4. These calculations reveal that smaller noise levels (such as, $\sigma = 5$) result in weaker privacy, with ϵ values exceeding 9, which may not meet acceptable privacy standards. Therefore, we focus on noise settings that produce privacy budgets in the range $[0.1, 2]$, as this range offers a reasonable trade-off between privacy and utility for our system.

To validate the chosen range, we conduct experiments by varying the noise levels within this range and measure the model's performance and privacy leakage. This ensures that the final model configuration maintains both high accuracy and adequate privacy protection, suitable for deployment in our proposed smart UAV delivery system.

6.1.2. Datasets

In this paper, four widely used and representative image datasets, Fashion-Mnist, Mnist, Med-Mnist, and a self-constructed courier carton dataset, are used to validate the effectiveness and generality of the proposed method. Firstly, the Fashion-Mnist dataset contains gray-scale images of 10 types of fashion items. We use this dataset to simulate the application scenario of UAVs identifying different types of goods in a UAV delivery system in an edge computing environment. Secondly, the Mnist dataset contains handwritten numerical images, which are used by us to simulate the application of drone identification of goods numbering. Thirdly, the Med-Mnist dataset contains a variety of medical image data, which we use to simulate the need for a drone camera to identify medical items after capturing images. This is especially important for scenarios in emergency medical services or medical supply deliveries where drones need to quickly process and identify medical items or symbols. Finally, we collected some images of courier cartons from real delivery scenarios, divided into two categories: intact and broken. This dataset is used to simulate



(a) Servers number = 5, Epoch = 50.

(b) Servers number = 10, Epoch = 50.

(c) Servers number = 20, Epoch = 50.

(d) Servers number = 5, Epoch = 100.

Fig. 9. Accuracy with different number edge servers and datasets ($\varepsilon = 0.1$).

the monitoring and identification of the condition of courier cartons by UAVs during the delivery process, which helps to improve the reliability of the delivery system and customer satisfaction.

(1) Fashion-Mnist [57]: This dataset contains 10 categories, including graphic, t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot, as shown in Fig. 6. The training dataset and testing dataset each contain 6000 and 1000 samples for each category, respectively. The graphic are represented as a 28 × 28 pixel array, with each pixel having a value between 0 and 255 as an 8-bit unsigned integer (uint8). The data is stored in a three-dimensional NDArray, with the last dimension representing the number of channels.
(2) Mnist [58]: This dataset is a large collection of handwritten digit samples, with 6000 training samples and 10,000 testing samples. Each sample is a 28 × 28 pixel grayscale graphic.
(3) Med-Mnist [59]: This dataset is a collection of 10 pre-processed open medical image datasets, as shown in Fig. 7. It contains primary medical image modalities with diverse data scales.
(4) Box: The box dataset contains images of courier parcels collected by us, as shown in Fig. 8. The dataset contains of two categories: 873 images of intact boxes and 724 images of damaged boxes. The total number of image samples is 1597, and each image is uniformly cropped to size 224 × 224.

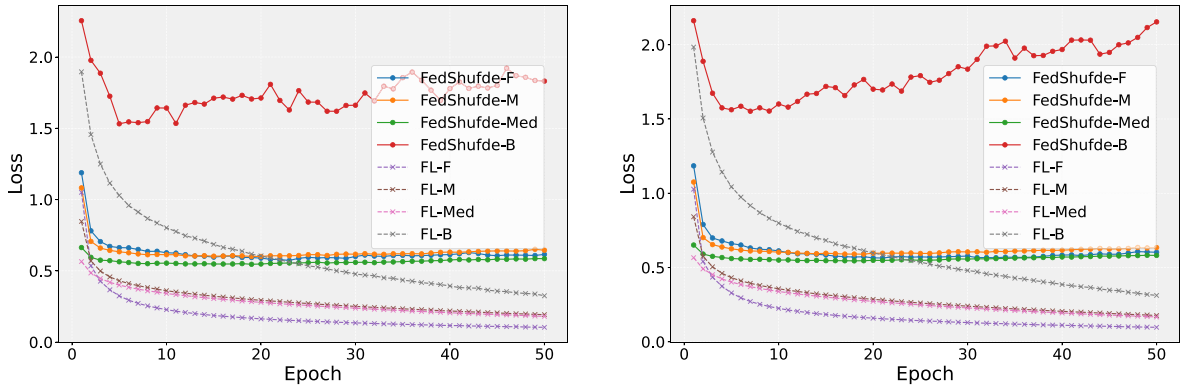
6.1.3. Datasets setting

We utilized a two-layer convolutional neural network (CNN) model to implement the tasks of graphic classification tasks on four datasets, namely Mnist dataset, Fashion-Mnist dataset, and Med-Mnist dataset. For the latter datasets, we developed a Logistic Regression model for predictive classification. To simplify the notation, "F" represents the Fashion-MNIST dataset, "M" represents MNIST, "Med" represents Med-MNIST, and "B" represents Box. To ensure fair distribution of training and testing data, we divided them equally among all edge servers. The weight and learning rate of edge servers were set to 1 and 0.01, respectively. To account for experimental variability, we conducted 10 experiments for each dataset and calculated the average value as the final result. For the Box dataset, we used 90% of the box images in both good and broken categories for training and the remaining 10% for testing.

6.2. Results

In Fig. 9, we illustrate the accuracy of the FedShufde framework using different indices, including edge servers, and datasets.

The experiments designed in this paper are compared under different number of edge servers (5, 10, 20) and training epoch (50, 100), and the main metrics are classification accuracy, loss, and energy consumption. When the epoch is 50 and the number of edge servers is 5, 10, 20, as shown in Figs. 9(a), 9(b), 9(c). the accuracy of the FedShufde method is significantly lower than that of the traditional FL method, with an accuracy of around 20%–30% and large fluctuations. The accuracy of the Unified FL method is higher, around 60% on the Fashion-Mnist dataset and between 40%–50% on the Mnist and Med-Mnist datasets. When the epoch is 100 and the number of edge servers is 5, as in Fig. 9(d). Although the FedShufde method introduces differential privacy techniques to enhance data privacy protection, this also leads to a decrease in model accuracy. For all settings of the number of servers and the number of training rounds, the accuracy of



(a) Servers number = 5, Epoch = 50.

(b) Servers number = 10, Epoch = 50.

(c) Servers number = 20, Epoch = 50.

(d) Servers number = 5, Epoch = 100.

Fig. 10. Loss with different number edge servers and datasets ($\epsilon = 0.1$).

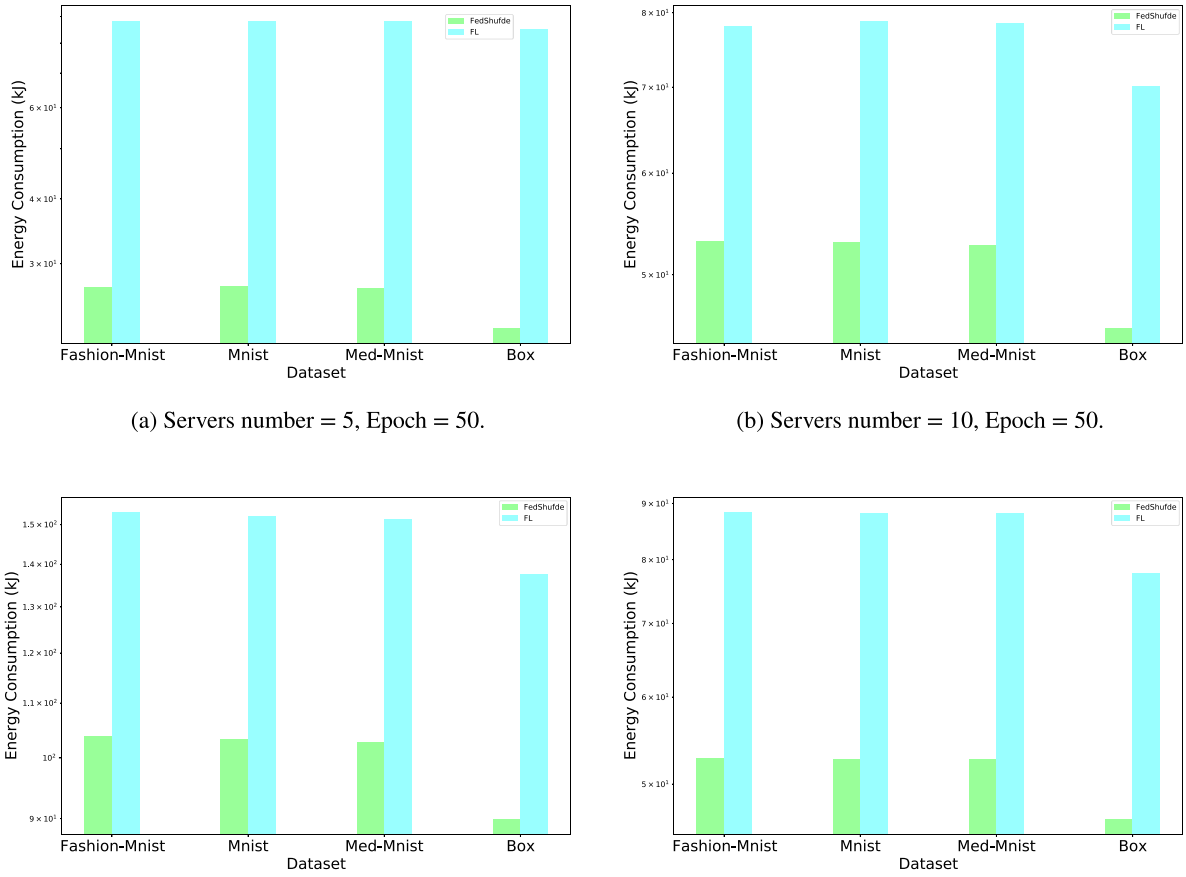
the traditional FL method is generally higher than that of the FedShufde method. This trade-off is expected because DP inevitably introduces noise and reduces model performance while protecting user privacy.

To better understand the convergence behavior of FedShufde, we conducted an empirical analysis comparing it with traditional FL. Specifically, we tracked the accuracy and loss metrics over multiple training rounds to assess convergence rates. The results, presented in Fig. 10, indicates that FedShufde initially converges more slowly compared to traditional FL. This is likely due to the increased variance introduced by the shuffling mechanism during the aggregation step, which affects the stability of model updates.

Despite the slower initial convergence, FedShufde ultimately reaches a stable state after more training rounds. However, the final accuracy achieved is lower compared to traditional FL, suggesting a potential trade-off between the benefits of shuffling (such as, improved privacy) and the efficiency of convergence. These findings provide valuable insights into the behavior of FedShufde, highlighting areas for potential improvement in future work.

Fig. 10 shows the variation of loss of various methods on multiple datasets with different number of edge servers (5, 10, 20) and training epoch (50, 100). When the epoch is 50 and the number of edge servers is 5, 10, 20, as shown in Figs. 10(a), 10(b), 10(c), the loss value of FedShufde method on dataset Fashion-Mnist stays around 1.5 throughout the training process, with large fluctuations. The loss value on Mnist and Med-Mnist datasets is around 1.0, which is relatively stable. The loss values of traditional FL methods are low, around 0.5 for dataset Fashion-Mnist and Med-Mnist, and decrease rapidly in the early stages of training, converging faster. When the epoch is 100 and the number of edge servers is 5, as shown in Fig. 10(d), the loss value decreases slightly and the fluctuation increases. The traditional FL method has lower loss values but increased volatility. The FedShufde method results in higher loss values and higher volatility due to the introduction of differential privacy technique. As the number of servers increases and the number of training rounds increases, the effect of added noise remains significant despite a decrease in the loss values.

Based on the experimental results shown in Fig. 11, we observe a nuanced performance of energy consumption across different configurations of the FedShufde and traditional FL methods: (i) *Lower Energy with FedShufde at Low Server Counts:* When the number of edge servers is 5 (Figs. 11(a) and 11(d)), the FedShufde method generally consumes less energy than the traditional FL method, particularly at both 50 and 100 epochs. This suggests that FedShufde is effective in scenarios with fewer servers, optimizing energy through its distributed shuffling and differential privacy techniques. (ii) *Slight Increase in Energy at Higher Server Counts:* For configurations with 10 and 20 servers (Figs. 11(b) and 11(c)), the FedShufde method shows a slight increase in energy consumption compared to the traditional FL method. This may be due to the overhead introduced by differential privacy and shuffling processes across a larger number of servers, impacting the energy balance. (iii) *Dataset-Specific Efficiency:* The FedShufde method demonstrates particularly lower energy consumption on the Box dataset across different server and epoch settings. This result highlights FedShufde's potential to optimize energy consumption in specific datasets with unique data distributions or processing demands. Conversely, for datasets like Fashion-Mnist, Mnist, and Med-Mnist, the FedShufde method sometimes consumes slightly more energy than traditional FL, potentially due to the added computation of privacy techniques. *(iv) Convergence of Energy Performance:* As the number of training rounds increases, FedShufde's energy efficiency improves. With 100 epochs



(a) Servers number = 5, Epoch = 50.

(b) Servers number = 10, Epoch = 50.

(c) Servers number = 20, Epoch = 50.

(d) Servers number = 5, Epoch = 100.

Fig. 11. Energy consumption with different number edge servers and datasets ($\epsilon = 0.1$).

(Fig. 11(d)), the gap in energy consumption between FedShufde and FL narrows, or even becomes more favorable to FedShufde in some cases.

In summary, the FedShufde framework not only enhances privacy protection for edge-based smart UAV delivery systems but also offers notable energy optimization benefits. By using the LDP mechanism and shuffling model, it effectively protects sensitive user data while balancing energy efficiency across various datasets and server configurations. The shuffle model, while effective in providing privacy, introduces a degree of randomness that can adversely affect convergence. Specifically, the shuffling of updates across clients may result in delayed model stabilization, particularly in highly heterogeneous data environments. We have added a discussion on strategies to mitigate these effects, such as adjusting learning rates dynamically and incorporating adaptive averaging techniques to counterbalance the added randomness.

7. Conclusion and future work

In this paper, we have proposed a novel framework named FedShufde to protect user privacy in the edge-based smart UAV delivery system. The framework consists of three layers of protection. First, FL trains the model while keeping the user data local. Secondly, LDP perturbs the model parameters to protect the privacy of the trained model. Finally, the shuffle model scrambles the connection between the data sender and receiver to prevent the attacker from launching reverse attacks. For the UAV delivery system, the first layer of guarantee is that user data does not leave the local environment to participate in model training by aggregating model parameters. The second layer of guarantee is that the model parameters receive a local differential privacy perturbation to ensure the privacy of the FL framework. The third layer of guarantee is that the shuffle model scrambles the upload order of the uploaded model parameters to prevent attackers from backlink attacks. Extensive experimental results based on public datasets and a real-world edge-based smart UAV delivery system demonstrate that our proposed FedShufde framework is effective in protecting data privacy in edge-based IoT systems. We have implemented privacy-preserving measures for users' package information. However, we recognize that other personal information, such as mobile phone numbers, addresses, shopping habits, and more, also needs to be protected. In the future, we plan to extend the FedShufde framework to protect more personal information. Additionally, we acknowledge that data heterogeneity across different edge devices is a common challenge in FL. To address this, we plan to analyze and enhance the FedShufde framework's performance under non-IID data distributions in future work.

CRediT authorship contribution statement

Aiting Yao: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Shantanu Pal:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization. **Gang Li:** Writing – review & editing, Supervision, Resources, Methodology, Conceptualization. **Xuejun Li:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization. **Zheng**

Zhang: Writing – review & editing, Software, Methodology, Investigation. **Frank Jiang:** Writing – review & editing, Supervision, Methodology, Investigation. **Chengzu Dong:** Writing – review & editing, Validation, Methodology, Data curation. **Xiao Liu:** Writing – review & editing, Supervision, Resources, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Aiting Yao reports financial support was provided by National Natural Science Foundation of China.

Acknowledgments

This work was supported by the National Natural Science Foundation of China Project (No. 62372004).

Data availability

No data was used for the research described in the article.

References

[1] L.U. Khan, I. Yaqoob, N.H. Tran, S.A. Kazmi, T.N. Dang, C.S. Hong, Edge-computing-enabled smart cities: A comprehensive survey, IEEE Internet Things J. 7 (10) (2020) 10200–10232.
[2] H. Shakhatreh, A.H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N.S. Othman, A. Khreishah, M. Guizani, Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges, Ieee Access 7 (2019) 48572–48634.
[3] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2322–2358.
[4] Y. Mekdad, A. Aris, L. Babun, A. El Fergougui, M. Conti, R. Lazzeretti, A.S. Uluagac, A survey on security and privacy issues of UAVs, Comput. Netw. 224 (2023) 109626.
[5] H. Liu, S. Li, W. Li, W. Sun, Efficient decentralized optimization for edge-enabled smart manufacturing: A federated learning-based framework, Future Gener. Comput. Syst. 157 (2024) 422–435.
[6] Y. Zhi, Z. Fu, X. Sun, J. Yu, Security and privacy issues of UAV: a survey, Mob. Netw. Appl. 25 (2020) 95–101.
[7] S.A.H. Mohsan, N.Q.H. Othman, Y. Li, M.H. Alsharif, M.A. Khan, Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends, Intell. Serv. Robot. (2023) 1–29.
[8] TechSpot, Wi-fi drones were used by hackers to penetrate a financial firm's network remotely, 2022, URL https://www.techspot.com/news/96321-drones-helped-hackers-/penetrate-financial-firm-network-remotely.html.
[9] S. Dougherty, R. Tourani, G. Panwar, R. Vishwanathan, S. Misra, S. Srikanteswara, APECS: A distributed access control framework for pervasive edge computing services, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 1405–1420.
[10] B. Bera, D. Chattaraj, A.K. Das, Designing secure blockchain-based access control scheme in IoT-enabled internet of drones deployment, Comput. Commun. 153 (2020) 229–249.
[11] I. Mistry, S. Tanwar, S. Tyagi, N. Kumar, Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges, Mech. Syst. Signal Process. 135 (2020) 106382.
[12] S. Pal, M. Hitchens, V. Varadharajan, Access control for internet of things—Enabled assistive technologies: An architecture, challenges and requirements, in: Assistive Technology for the Elderly, Elsevier, 2020, pp. 1–43.
[13] V.U. Castrillo, A. Manco, D. Pascarella, G. Gigante, A review of counter-UAS technologies for cooperative defensive teams of drones, Drones 6 (3) (2022) 65.
[14] M.E. Mkiramweni, C. Yang, J. Li, W. Zhang, A survey of game theory in unmanned aerial vehicles communications, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3386–3416.
[15] S. Pal, Internet of Things and Access Control: Sensing, Monitoring and Controlling Access in IoT-Enabled Healthcare Systems, vol. 37, Springer Nature, 2021.
[16] H. Studiawan, G. Grispos, K.-K.R. Choo, Unmanned aerial vehicle (UAV) forensics: The good, the bad, and the unaddressed, Comput. Secur. (2023) 103340.
[17] Z. Lu, N. Yu, X. Wang, Incentive mechanism and path planning for unmanned aerial vehicle (UAV) hitching over traffic networks, Future Gener. Comput. Syst. 145 (2023) 521–535.
[18] M. Bi, Y. Wang, Z. Cai, X. Tong, A privacy-preserving mechanism based on local differential privacy in edge computing, China Commun. 17 (2020) 50–65.
[19] M. Du, K. Wang, Z. Xia, Y. Zhang, Differential privacy preserving of training model in wireless big data with edge computing, IEEE Trans. Big Data 6 (2020) 283–295.
[20] J. Zhang, Y. Zhao, J. Wang, B. Chen, FedMEC: Improving efficiency of differentially private federated learning via mobile edge computing, Mob. Netw. Appl. (2020) 2421–2433.
[21] A. Jolfaei, K. Kant, Data security in multiparty edge computing environments, in: Government Microcircuit Applications & Critical Technology Conference, United States, 2019, pp. 17–22.
[22] F. Hu, B. Chen, Channel coding scheme for relay edge computing wireless networks via homomorphic encryption and NOMA, IEEE Trans. Cogn. Commun. Netw. 6 (2020) 1180–1192.
[23] K. Wei, J. Li, M. Ding, C. Ma, H.H. Yang, F. Farokhi, et al., Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469.
[24] Y. Wang, Q. Wang, L. Zhao, C. Wang, Differential privacy in deep learning: Privacy and beyond, Future Gener. Comput. Syst. (2023).
[25] H. Zhang, J. Bosch, H. Holmström Olsson, Engineering federated learning systems: A literature review, in: Software Business: 11th International Conference, ICSOB 2020, Karlskrona, Sweden, November 16–18, 2020, Proceedings 11, Springer, 2021, pp. 210–218.
[26] D. Deuber, D. Schröder, CoinJoin in the wild: An empirical analysis in dash, in: Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26, Springer, 2021, pp. 461–480.
[27] C. Lachner, Urban Sensing Environment: Exploiting the adapation space for data Protection (Ph.D. thesis), 2022.
[28] X. Nie, L.T. Yang, J. Feng, S. Zhang, Differentially private tensor train decomposition in edge-cloud computing for SDN-based internet of things, IEEE Internet Things J. 7 (2020) 5695–5705.
[29] G. Liu, Z. Tang, B. Wan, Y. Li, Y. Liu, Differential privacy location data release based on quadtree in mobile edge computing, Trans. Emerg. Telecommun. Technol. (2020) 1–17.
[30] J. Guo, W. Wu, Differential privacy-based online allocations towards integrating blockchain and edge computing, 2021, pp. 1–12, ArXiv, arXiv:2101.02834.
[31] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, Prochlo: Strong privacy for analytics in the crowd, in: Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 441–459.
[32] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, A. Thakurta, Amplification by shuffling: From local to central differential privacy via anonymity, in: Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2019, pp. 2468–2479.
[33] V. Feldman, A. McMillan, K. Talwar, Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling, 2020, pp. 1–42, arXiv preprint arXiv:2012.12803.
[34] A. Cheu, Differential Privacy in the Shuffle Model (Ph.D. thesis), Northeastern University, 2021.
[35] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, et al., Differential privacy-enabled federated learning for sensitive health data, 2019, pp. 1–6, arXiv preprint arXiv:1910.02578.
[36] X. Yuan, J. Chen, J. Yang, N. Zhang, T. Yang, T. Han, et al., FedSTN: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction, IEEE Trans. Intell. Transp. Syst. (2022) 1–11.
[37] C. Zhang, X. Liu, J. Xu, T. Chen, G. Li, F. Jiang, et al., An edge based federated learning framework for person re-identification in UAV delivery service, in: 2021 IEEE International Conference on Web Services, ICWS, 2021.
[38] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv. Tutor. 23 (2021) 1622–1658.
[39] U. Mohammad, S. Sorour, M. Hefeida, Task allocation for mobile federated and offloaded learning with energy and delay constraints, in: 2020 IEEE International Conference on Communications Workshops, ICC Workshops, 2020, pp. 1–6.
[40] Y. Song, T. Liu, T. Wei, X. Wang, Z. Tao, M. Chen, FDA [3]: Federated defense against adversarial attacks for cloud-based iIoT applications, IEEE Trans. Ind. Inform. 17 (2020) 7830–7838.
[41] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, S. Liu, Efficient and privacy-enhanced federated learning for industrial artificial intelligence, IEEE Trans. Ind. Inform. 16 (2019) 6532–6542.
[42] H. Cao, S. Liu, R. Zhao, X. Xiong, IFed: A novel federated learning framework for local differential privacy in power internet of things, Int. J. Distrib. Sens. Netw. 16 (2020) 1550147720919698.
[43] C. Dwork, Differential privacy, in: Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II, 2006, pp. 1–12.
[44] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3, Springer, 2006, pp. 265–284.

[45] R. Redberg, Y. Zhu, Y.-X. Wang, Generalized ptr: User-friendly recipes for data-adaptive algorithms with differential privacy, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2023, pp. 3977–4005.

[46] Y. Jiang, X. Chang, Y. Liu, L. Ding, L. Kong, B. Jiang, Gaussian differential privacy on Riemannian manifolds, Adv. Neural Inf. Process. Syst. 36 (2023) 14665–14684.

[47] A. Cheu, A. Smith, J. Ullman, D. Zeber, M. Zhilyaev, Distributed differential privacy via shuffling, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2019, pp. 375–403.

[48] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, e.a. D. Lie, Prochlo: Strong privacy for analytics in the crowd, in: Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 441–459.

[49] C.L. Mallows, Non-null ranking models. I, Biometrika 44 (1957) 114–130.

[50] C. Meehan, A.R. Chowdhury, K. Chaudhuri, S. Jha, A shuffling framework for local differential privacy, 2021, pp. 1–24, arXiv preprint arXiv:2106.06603.

[51] V. Mothukuri, R.M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Gener. Comput. Syst. 115 (2021) 619–640.

[52] M. Abadi, A. Chu, I. Goodfellow, H.B. McMahan, I. Mironov, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.

[53] M. Naseri, J. Hayes, E. Cristofaro, Local and central differential privacy for robustness and privacy in federated learning, 2020, pp. 1–19, arXiv preprint arXiv:2009.03561.

[54] X. Wang, Y. Jin, S. Schmitt, M. Olhofer, Recent advances in Bayesian optimization, ACM Comput. Surv. 55 (13s) (2023) 1–36.

[55] S. Ament, S. Daulton, D. Eriksson, M. Balandat, E. Bakshy, Unexpected improvements to expected improvement for bayesian optimization, Adv. Neural Inf. Process. Syst. 36 (2023) 20577–20612.

[56] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, I. Guyon, Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020, in: NeurIPS 2020 Competition and Demonstration Track, PMLR, 2021, pp. 3–26.

[57] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, pp. 1–6, arXiv preprint arXiv:1708.07747.

[58] Y. LeCun, C. Cortes, C.J.C. Burges, THE MNIST DATABASE of handwritten digits, 1998, URL http://yann.lecun.com/exdb/mnist/.

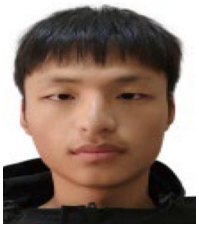
[59] J. Yang, R. Shi, B. Ni, Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis, in: 2021 IEEE 18th International Symposium on Biomedical Imaging, ISBI, 2021, pp. 191–195.

Aiting Yao received the Bachelor's in the School of Mathematics and Statistics, Chaohu College, Hefei, Anhui and the Master's in the School of Mathematics and Science, Anhui University, Hefei, Anhui, China in 2012 and 2020, respectively. She is currently pursuing the Ph.D. degree with the School of Computer Sciences and Technology, Anhui University, Hefei, Anhui, China. She is current research interests include mobile edge computing, privacy preserving, differential privacy, and federated learning.

Shantanu Pal is associated with the School of Information Technology, Deakin University, Melbourne, Australia. Shantanu has extensive research experience in Internet of Things, big data and distributed applications, access control, trust management, blockchain technology, mobile and cloud computing, and distributed networks. Shantanu has several publications in highly ranked conferences and journals, including IEEE TII, IEEE IoT journals, etc. He is a Senior Member of IEEE.

Gang Li is a Professor at the School of Information Technology Deakin University, Melbourne, Australia. He received his Ph.D. in computer science in 2005. He joined the School of Information Technology at Deakin University (Australia) as an associate lecturer (2004–2006), lecturer (2007–2011), senior lecturer (2012–2016). His research interests are in the area of data mining, machine learning, and business intelligence. He is an IEEE Senior Member.

Xuejun Li is currently a full Professor with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. He received his Ph.D. degree in computer application technology from the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, in 2008. He His major research interests include mobile edge computing, workflow systems, cloud computing, and intelligent software.



Zheng Zhang received the Master's degree in computer science and technology from the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China in 2021-2024, respectively. He is currently pursuing the Ph.D degree with the School of Computer Sciences and Technology, Anhui University, Hefei, Anhui, China, He is current research interests include privacy preserving, mobile edge comouting, data management, machine learning, and federated learning.

Frank Jiang is a Senior Lecturer at the School of Information Technology, Deakin University, Australia. He is a mature and active mid-career researcher in Australia for over 20 years, associated with the school of Information Technology, Deakin University, Australia, and holding a honourable position in University of New South Wales (UNSW), Canberra, Australia. He completed his PhD degree in communication engineering and cyber security at University of Technology, Sydney (UTS). His expertise spans cybersecurity, V2X vehicular networks, blockchain, machine learning, and data analytics. He has published in leading journals like TIFS, ACM Computing Surveys, and KBS, as well as top conferences including AsiaCCS, AAAI, and ICDM. He is an IEEE Senior Member.

Chengzu Dong is an Assistant Professor at the Division of Artificial Intelligence, School of Data Science, Lingnan University, Hong Kong, China. He received his PhD from Deakin University in Australia. He has contributed to multiple high-quality conference papers and primarily focuses his research on areas such as cybersecurity, machine learning, blockchain, unmanned aerial vehicle (UAV) delivery, and edge computing.

Jia Xu received his bachelor's, master's and Ph.D. degrees in computer science and technology from the School of Computer Science and Technology, Anhui University, Hefei, China, in 2014, 2017 and 2022, respectively. He was a software engineer focusing on industrial projects and solutions in iFLYTEK Co., Ltm from 2017-2018. From 2022 to 2024, he was a Postdoctoral Researcher at the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. He is currently a Lecturer at the School of Internet, Anhui University. His current research interests include MEC, workflow systems, cloud computing, and resource management.

Xiao Liu is an Associate Professor at School of Information Technology, Deakin University, Melbourne, Australia. Before that, he was an Associate Professor at Software Engineering Institute, East China Normal University, Shanghai, China. He received his Ph.D. degree in Computer Science and Software Engineering from the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia in 2011. He received his Master and Bachelor degrees from the School of Management, Hefei University of Technology, Hefei, China, in 2007 and 2004 respectively, all in Information Management and Information System. His research areas include Software Engineering, Distributed Computing and Service Computing, with special interests in workflow systems, cloud and edge computing, big data analytics, and human-centric software engineering. He is a Senior Member of IEEE and ACM.