FedSky: An Efficient and Privacy-Preserving Scheme for Federated Mobile Crowdsensing

Xichen Zhang, Rongxing Lu, *Fellow, IEEE*, Jun Shao, *Member, IEEE*, Fengwei Wang, *Student Member, IEEE*, Hui Zhu, *Senior Member, IEEE*, and Ali A. Ghorbani, *Senior Member, IEEE*

***Abstract*—Mobile crowdsensing (MCS) is a newly emerged sensing paradigm, where a large group of mobile workers collectively sense and share data for real-time services. However, one major problem that hinders the further development of MCS is the potential leakage of workers' data privacy. In this article, we integrate federated learning (FL) with MCS and introduce a novel sensing system, called federated MCS (F-MCS). In F-MCS, the workers can optimize the global model while keeping all the sensitive training data locally, thus ensuring their data privacy. Nevertheless, there are still two major issues in F-MCS. The first issue is that in F-MCS services, the workers are heterogeneous in terms of computational capacities and data resources. Hence, qualified workers should be appropriately selected to improve the efficiency of the training process. The second issue is that F-MCS is a *cross-device* FL system, where the platform will finally get the global model after multiple training rounds. However, most privacy-preserving techniques are designed for *cross-silo* FL platforms, which cannot be applied to real-world F-MCS scenarios. To tackle the above problems, in this article, we propose a privacy-preserving scheme for F-MCS, namely, FedSky. Mainly, by extending the classic FedAvg algorithm, FedSky selects qualified workers based on the constrained group skyline (CG-skyline) and securely aggregates model updates based on the homomorphic encryption technique. Comprehensive security analysis demonstrates the privacy preservation of FedSky. Extensive experiments are conducted on an image classification task, where the comparison results validate the proposed scheme's efficiency and effectiveness.**

***Index Terms*—FedAvg, federated learning (FL), group skyline (G-skyline), homomorphic encryption (HE), mobile crowdsensing (MCS).**

Manuscript received May 8, 2021; revised August 12, 2021; accepted August 22, 2021. Date of publication August 31, 2021; date of current version March 24, 2022. The work of Rongxing Lu was supported in part by the National Science and Engineering Research Council of Canada (NSERC) under Grant Rgpin 04009. The work of Ali A. Ghorbani was supported by NSERC through the Discovery Grant and Canada Research Chair. This work was supported in part by the NSF of Zhejiang Province under Grant LZ18F020003; in part by NSFC under Grant U1709217 and Grant 61972304; and in part by NSF of Shaanxi Province under Grant 2019ZDLGY12-02. *(Corresponding author: Rongxing Lu.)*

Xichen Zhang, Rongxing Lu, and Ali A. Ghorbani are with the Faculty of Computer Science, Canadian Institute for Cybersecurity, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: xichen.zhang@unb.ca; rlu1@unb.ca; ghorbani@unb.ca).

Jun Shao is with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: chn.junshao@gmail.com).

Fengwei Wang and Hui Zhu are with the School of Cyber Engineering, Xidian University, Xi'an 710126, China (e-mail: zhuhui@xidian.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3109058

I. Introduction

OVER the past few years, the explosions of mobile communication and the Internet of Things (IoT) have brought forth a new paradigm of sensing approach, called mobile crowdsensing (MCS) [1]. Essentially, a crowd of mobile users, namely, workers, is recruited by the MCS platform to outsource their sensing data for certain tasks, such as environmental monitoring [2], traffic density evaluation [3], urban planning [4], location navigation [5], and healthcare provisioning [6]. However, it is inevitable for the workers to share their sensing information (e.g., daily trajectories, real-time locations, and surrounding environments) to the platform during the sensing task. The information leakage may lead to serious privacy issues. For example, an attacker may infer a worker's daily behavior by analyzing his/her sensing data. As a result, protecting workers' sensitive information from being disclosed is one of the major challenges for MCS applications.

Up to now, a large and growing body of literature has been studied for solving the privacy challenges in MCS. Among all the studies, federated learning (FL) can be considered a potential and practical solution. In FL, the platform iteratively selects random workers to download a trainable model [7]. Then, the selected workers update the model with their own data and upload the updated model to the platform. After that, the platform aggregates multiple updates to further improve the model [8]. The distributed nature of FL enables the workers to optimize the shared model while keeping all the training data locally, thus ensuring their privacy. Based on [9] and [10], FL can be broadly categorized into *cross-silo* FL and *cross-device* FL. In *cross-silo* FL, the workers are functional organizations (e.g., financial agencies) with abundant computing resources. The trained model is exclusively released to these organizations, but not the FL aggregation platform. In contrast, in the *cross-device* FL, the workers are heterogeneous mobile users with limited computational powers. The FL platform will finally get access to the trained model.

In this article, by introducing *cross-device* FL into MCS, we propose a novel sensing scenario, called federated MCS (F-MCS). F-MCS allows workers to build a robust and secure machine learning model without sharing their sensing data. As a result, F-MCS can address the critical issue of workers' information leakage in traditional MCS systems and is expected to become a new hotspot of mobile sensing services.

Nevertheless, there are still two major issues if we simply integrate the FL technique with MCS services.

1) The selection of stable workers in F-MCS is challenging. In F-MCS, the mobile workers are heterogeneous with different computational capacities and data resources. For example, their devices are different in terms of hardware (CPU and memory), network connectivity (3G, 4G, 5G, Wi-Fi, and signal strength), and power status (battery level and charging capacities). The synchronous training protocol [10] is widely used in FL, where no worker can proceed to the next training round until all workers' data have been uploaded. The workers with limited computational powers require a longer time to update the model parameters. Therefore, such a problem will delay the subsequent aggregation step and make the overall training process inefficient.
2) The lack of practical privacy-preserving solutions for *cross-device* F-MCS platforms. Even though the workers can avoid leaking their sensing data in F-MCS, they still face possible information disclosure. Specifically, the shared model information (e.g., the gradient) uploaded by the workers can leak important statistical patterns of the local data sets [11]–[13]. Consequently, homomorphic encryption (HE)-based privacy-preserving data aggregation techniques have been well-studied for enhancing the privacy of FL platforms [14]. However, most of the approaches are designed for *cross-silo* FL setting, where the final global model can only be accessed by the participants (i.e., workers) [10], [11], [15]–[19]. In contrast, F-MCS is a *cross-device* FL system, where the final model will be released to the F-MCS platform for certain real-time services. Hence, the existing methods cannot be applied in F-MCS. There is an immediate necessity for designing practical and privacy-preserving solutions for F-MCS applications.

In this article, aiming at solving the above challenges, we propose a novel and secure data aggregation scheme for F-MCS applications, called FedSky. By extending the classic FedAvg algorithm [20], FedSky is characterized by employing group skyline (G-skyline) for selecting qualified workers and HE for securely aggregating data. Specifically, the main contributions of this article are threefold as follows.

1) We propose a novel and effective worker selection mechanism in FedSky for F-MCS. Concretely, in each communication round, instead of choosing a group of workers at random, we select a skyline group of workers in terms of workers' local data sizes and their mobile devices' computational powers. In this way, compared to the traditional FedAvg algorithm, our approach can significantly reduce workers' computational time and the platform's waiting time and, therefore, significantly maximize the efficiency of the FL process.
2) We propose a novel privacy-preserving data aggregation scheme for the F-MCS platform based on the HE technique. The scheme is designed for the *cross-device* FL setting such that the final global model can be accessed by the F-MCS platform. In addition, the overall training process requires no interaction between the selected workers. Except for the left neighbor ID and right neighbor ID, each selected worker knows nothing about other workers' information.
3) We build a custom simulator for performance evaluation. Extensive experimental results validate the effectiveness and efficiency of the proposed schemes. In particular, by introducing a novel worker selection mechanism and a privacy-preserving data aggregation protocol, we can greatly improve the efficiency of the FL process without affecting its accuracy.

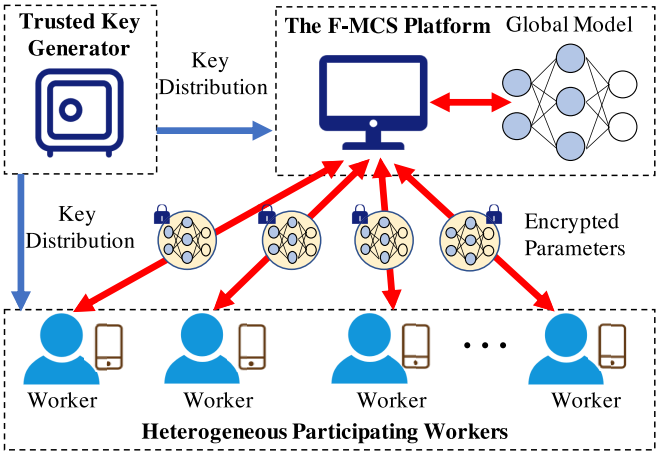


Fig. 1. System model under consideration.

The remainder of this article is organized as follows. In Section II, we introduce our system model, security model, and design goals. In Section III, we describe some preliminaries. In Section IV, we propose our scheme in details. Then, in Section V and Section VI, we present the security analysis and performance evaluation, followed by the related work in Section VII. Finally, we recap the conclusions in Section VIII.

II. Models and Designed Goals

In this section, we first formalize our system model and security model, and then identify our design goals.

A. System Model

In our system model (see in Fig. 1), we mainly consider a typical F-MCS scenario, including three entities, namely, a trusted key generator $\mathcal{TKG}$, an F-MCS platform $\mathcal{P}$, and a group of heterogeneous workers $\mathcal{W} = \{w_1, w_2, \ldots\}$.

1) *Trusted Key Generator ($\mathcal{TKG}$):* $\mathcal{TKG}$ is a trusted authority that generates and distributes proper keys to the corresponding entities so that a certain F-MCS task can be completed cooperatively.
2) *F-MCS Platform ($\mathcal{P}$):* $\mathcal{P}$ is the trusted platform for providing F-MCS services and is responsible for performing an F-MCS task, including registering workers, initializing the task model, selecting workers, and training the model. More specifically, upon registration, $\mathcal{P}$ assigns an unique identifier $\mathtt{ID}_{w_i}$ to each registered worker w_i, and then broadcasts the list of workers' identifiers to $\mathcal{TKG}$. In the training process, $\mathcal{P}$ first initializes the global model and selects a fraction of qualified workers from $\mathcal{W}$ (details of worker selection will be introduced

in Section IV-B). Next, $\mathcal{P}$ distributes the model to the selected workers and trains the model by aggregating the worker-provided model parameters in each training round. Multiple rounds of interactions between $\mathcal{P}$ and the selected workers are demanded to achieve the final global model.

3) *Participating Workers* $\mathcal{W} = \{w_1, w_2, \ldots\}$: $\mathcal{W}$ are the participating workers who wish to conduct a certain F-MCS task. If a worker in $\mathcal{W}$ is selected by $\mathcal{P}$, at first, he/she needs to collect the raw sensing data using his/her mobile device(s). After receiving the global model, each selected worker needs to update the model parameters by training the model with his/her local data, encrypt the model parameters, and exchange the ciphertexts with $\mathcal{P}$. To be selected in each training round, the workers need to periodically send their current training capacities and computational statuses to $\mathcal{P}$, e.g., the size of the local data set, the current CPU share/battery/memory of their mobile devices.

B. Security Model

First, we consider $\mathcal{P}$ and $\mathcal{TKG}$ are fully trusted. Notably, some malware have been installed in $\mathcal{P}$ by an adversary $\mathcal{A}$ without being detected. Therefore, $\mathcal{A}$ can monitor $\mathcal{P}$'s database and eavesdrop on the communication information between $\mathcal{P}$ and $\mathcal{W}$. Essentially, $\mathcal{A}$ is interested in the model updates that each worker uploads to the platform. With those updates, $\mathcal{A}$ may infer workers' real-time spatial–temporal information. Besides, we consider the workers $\mathcal{W}$ are honest-but-curious. So, all the workers strictly follow the designed protocols for updating, encrypting, and uploading model parameters but may be interested in other workers' data resources. Moreover, we assume that there is no collusion among workers in $\mathcal{W}$, and workers cannot collude with $\mathcal{P}$ either. It is worth noting that outside attackers may exploit other vulnerabilities of the F-MCS platform. As this work focuses on privacy preservation, those attacks are beyond this article's scope and will be discussed in our future work.

C. Design Goals

Based on the system model and security model mentioned above, our design goal is to develop an efficient and privacy-preserving scheme for F-MCS applications. Specifically, the following desirable properties should be achieved.

1) *Efficiency:* The proposed FedSky scheme should be efficient in terms of training models and uploading the encrypted model parameters in each communication round. Consequently, compared to the traditional FedAvg algorithm, $\mathcal{P}$'s waiting time should be shortened and the overall FL training efficiency should be improved in our scheme.
2) *Privacy Preservation:* We plan to design a privacy-preserving F-MCS framework, which can prevent the confidentiality of workers' sensitive information from being disclosed. More specifically, for $\forall w_i$, let $\mathcal{D}_i$ denote w_i's local data and x_i denote w_i's updated model parameters after each training round, both $\mathcal{D}_i$ and x_i should be kept secret to other workers. Moreover, even if $\mathcal{A}$ eavesdrops $\mathcal{P}$'s database and steals the communication data between $\mathcal{W}$ and $\mathcal{P}$, it still cannot identify each worker's uploaded model parameters x_i.

TABLE I
SUMMARY OF NOTATIONS

Notation	Definition
System Notations	
$\mathcal{TKG}$	The trusted key generator
$\mathcal{P}$	The F-MCS platform
$\mathcal{W} = \{w_1, w_2, ..., w_K\}$	The participating workers
Skyline Notations	
G-skyline	Group skyline
C-skyline	Constrained skyline
$\mathcal{C} = \{\mathtt{Con}_1, ..., \mathtt{Con}_d\}$	A set of constrains
$w_{ui} \preceq w'_{vi}$	$w_{ui} \prec w'_{vi}$ or $w_{ui} = w'_{vi}$
CG-skyline	Constrained group skyline
FL Notations	
$\mathcal{D}_i$	Worker w_i's local data
x_i	Worker w_i's model parameter
y_i	Weight for x_i
g_i	Worker w_i's local gradient for x_i
Model Notations	
N_i	Worker w_i's data size, i.e., $N_i = \lvert\mathcal{D}_i\rvert$
P_i	Worker w_i's computational power
$\mathcal{W}_s = \{w_1, w_2, ..., w_k\}$	The group of selected k workers

III. PRELIMINARIES

This section briefly introduces the background about FedAvg, constrained G-skyline (CG-skyline), Bilinear Pairing, and Paillier Cryptosystem. For reference, a summary of frequently used notations is given in Table I.

A. Fedavg Algorithm

The FedAvg [20] is a federated optimization algorithms for training ML/DL models in FL. Essentially, there are two entities in a typical FL setting, they are: 1) the data aggregator and 2) K participating workers $\{w_1, w_2, \ldots, w_K\}$. Let N_i denote the number of data sets that owned by worker w_i. In FedAvg, at the beginning of each round t, the data aggregator randomly selects k workers from all the K candidates, and then broadcasts the current global model x^t (at round t) to each of the k workers. After receiving the current model, worker w_i computes the average gradient on its local data set, which is denoted by g_i. Next, given a fixed learning rate η, w_i calculates $x_i \leftarrow x^t - \eta \cdot g_i$ and uploads x_i back to the data aggregator. Finally, the data aggregator updates the global model by $x^{t+1} \leftarrow \sum_{i=1}^{k}(N_i/N)x_i$, where $N = N_1 + N_2 + \ldots N_k$. The details of FedAvg are shown in Algorithm 1.

B. Different Variants of Skyline Queries

In this section, we use MCS workers as examples to illustrate skyline [21], G-skyline [22], [23], and constrained skyline (C-skyline) [24]. After that, we propose a novel skyline variant, called CG-skyline, which can retrieve skyline groups with specific dimension restrictions on the data.

Definition 1 (Skyline): Given a set of K participating workers $\mathcal{W} = \{w_1, w_2, \ldots, w_K\}$, each worker can be represented as a d-dimensional point, i.e., $w_i = (w_i[1], w_i[2], \ldots, w_i[d])$ for $i \in [1, K]$. Without loss of generality, we assume that for

Algorithm 1: FedAvg Algorithm [14]

Data aggregator executes:
for each training round $t = 1, 2, ...$ **do**
  Randomly selects k workers from all the K candidates
  for each selected worker w_i **in parallel do**
    $x_i \leftarrow$ **WorkerUpdate**(i, x^t)
  Takes the weighted average of the received model parameters from the k workers: $x^{t+1} \leftarrow \sum_{i=1}^{k} \frac{N_i}{N} x_i$
  if The model parameters converge **then**
    Training finishes.
  else
    Broadcasts the aggregated model parameters x^{t+1} to all the workers

WorkerUpdate(i, x^t): `// Executed by each selected worker`
for each local epoch e from 1 to E **do**
  batches $\leftarrow$ randomly divides dataset $\mathcal{D}_k$ into batches of size S
  for each batch b in batches **do**
    Computes the batch gradient g_i^b
    Updates model parameter by $x_i = x^t - \eta \cdot g_i^b$
Send the local model parameters x_i to the data aggregator

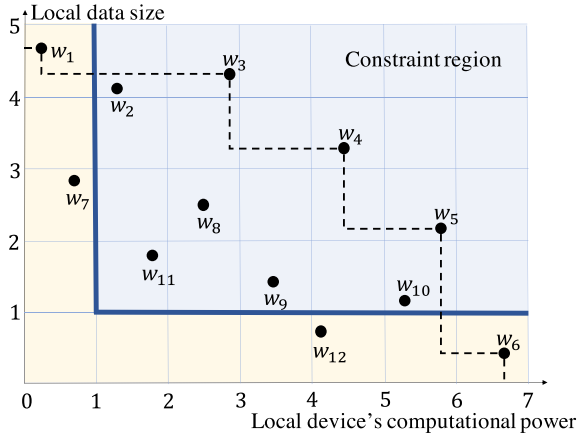


Fig. 2. Simple example of skyline, G-skyline, C-skyline, and CG-skyline with 12 workers in terms of local data size and device's computational power.

each dimension, larger values are more preferred. Let w_a and w_b denote two workers in $\mathcal{W}$, where $a, b \in [1, K]$ and $a \neq b$. We define w_a dominates w_b, denoted as $w_a \prec w_b$, if for all $j \in [1, d]$, $w_a[j] \geq w_b[j]$, and there exists at least one j such that $w_a[j] > w_b[j]$. Specifically, We define w_a equals w_b, denoted as $w_a = w_b$, if for all $j \in [1, d]$, $w_a[j] = w_b[j]$. The skyline workers of $\mathcal{W}$ are all the workers that are not dominated by others in $\mathcal{W}$.

Example 1: Fig. 2 shows a simple example of 12 workers. Each worker is represented by two attributes: 1) local data size and 2) local device's computational power. Based on Definition III-B, w_1, w_3, w_4, w_5, and w_6 are the skyline workers.

Definition 2 (k-Point G-Skyline): Given a set of workers $\mathcal{W}$, let $G = \{w_1, w_2, \ldots, w_k\}$ and $G' = \{w'_1, w'_2, \ldots, w'_k\}$ be two different groups with size k, where $k \leq K$ and $G, G' \subseteq \mathcal{W}$. Based on [22], we say G *g-dominates* G', denoted by $G \prec_g G'$, if we can find two permutations of k points for G and G', i.e., $G = \{w_{u1}, w_{u2}, \ldots, w_{uk}\}$ and $G' = \{w'_{v1}, w'_{v2}, \ldots, w'_{vk}\}$, such that $w_{ui} \preceq w'_{vi}$ for all $i \in [1, k]$, and there exists at least one i for $w_{ui} \prec w'_{vi}$. Consequently, the k-point G-skyline consists of all the skyline groups with k workers that are not g-dominated by other groups with the same size.

Example 2: In Fig. 2, if we consider $k = 3$, then for two groups $G = \{w_3, w_4, w_5\}$ and $G' = \{w_2, w_9, w_{10}\}$, we have $G \prec_g G'$ due to $w_3 \prec w_2$, $w_4 \prec w_9$, and $w_5 \prec w_{10}$. Moreover, $G = \{w_3, w_4, w_5\}$ is a 3-point G-skyline group since there exists no other group with the same size g-dominates G.

Definition 3 (C-Skyline): Given a set of d-dimensional workers $\mathcal{W}$, let $\mathcal{C} = \{\texttt{Con}_1, \texttt{Con}_2, \ldots, \texttt{Con}_d\}$ denote a set of d constraints, where each $\texttt{Con}_i$ for $i \in [1, d]$ is either a range $[\min_i, \max_i]$ along a dimension or an empty set $\emptyset$ indicating no constraint in that dimension. Typically, the conjunction of all constrains in $\mathcal{C}$ forms a *constraint region* in the d-dimensional attribute space. C-skyline [24]–[26] retrieves all the skyline points in the constraint region.

Definition 4 (k-Point CG-Skyline): Given a set of d-dimensional workers $\mathcal{W}$ and a set of constraints $\mathcal{C} = \{\texttt{Con}_1, \texttt{Con}_2, \ldots, \texttt{Con}_d\}$, the k-point CG-skyline consists of all the skyline groups in the constraint region with k workers that are not g-dominated by other groups with the same size.

Example 3: In Fig. 2, $\mathcal{C} = \{\texttt{Con}_1, \texttt{Con}_2\}$ is a constraint set, where $\texttt{Con}_1 = [1, 5]$ and $\texttt{Con}_2 = [1, 7]$. The corresponding constraint region is the light blue area in the figure. The C-skyline workers are w_3, w_4, and w_5 (since w_1 and w_6 are not in the constraint region). Besides, group $G = \{w_1, w_4, w_5\}$ is a 3-point G-skyline group but not a 3-point CG-skyline group because worker w_1 is not in the constraint region.

C. Bilinear Pairing

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of the same prime order q. g is a generator of group $\mathbb{G}$. Assume that $\mathbb{G}$ and $\mathbb{G}_T$ are equipped with a pairing, and a nondegenerated and efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has the following properties [27], [28].

1) *Bilinearity:* For all $x, y \in \mathbb{G}$, then $e(x^a, y^b) = e(x, y)^{ab}$ for $a, b \in \mathbb{Z}_q^*$.
2) *Nondegeneracy:* $e(g, g) \neq 1_{\mathbb{G}_T}$.
3) *Computability:* For all $x, y \in \mathbb{G}$, there exists an efficient algorithm to compute $e(x, y) \in \mathbb{G}_T$.

D. Paillier Cryptosystem

The Paillier cryptosystem consists of key generation $\texttt{KeyGen}(\kappa)$, encryption $\texttt{E}(pk, m)$, and decryption $\texttt{D}(sk, c)$.

1) $\texttt{KeyGen}(\kappa)$*:* Given a security parameter κ, two big primes p and q are first chosen such that $|p| = |q| = \kappa$. Then, $n = pq$ and $\lambda = lcm(p - 1, q - 1)$ are computed. Given a function $L(u) = (u - 1)/n$, let g denote a generator of $\mathbb{Z}_{n^2}^*$, $\mu = (L(g^\lambda \mod n^2))^{-1}$. The public key is $pk = (n, g)$, and the corresponding private key is $sk = (\lambda, \mu)$.
2) $\texttt{E}(pk, m)$*:* Given a plaintext $m \in \mathbb{Z}_n$ and a random number $r \in \mathbb{Z}_n^*$, the ciphertext can be calculated as $c = \texttt{E}(pk, m) = g^m \cdot r^n \mod n^2$.
3) $\texttt{D}(sk, c)$*:* Given the ciphertext $c \in \mathbb{Z}_{n^2}^*$, m can be recovered as $m = \texttt{D}(sk, c) = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

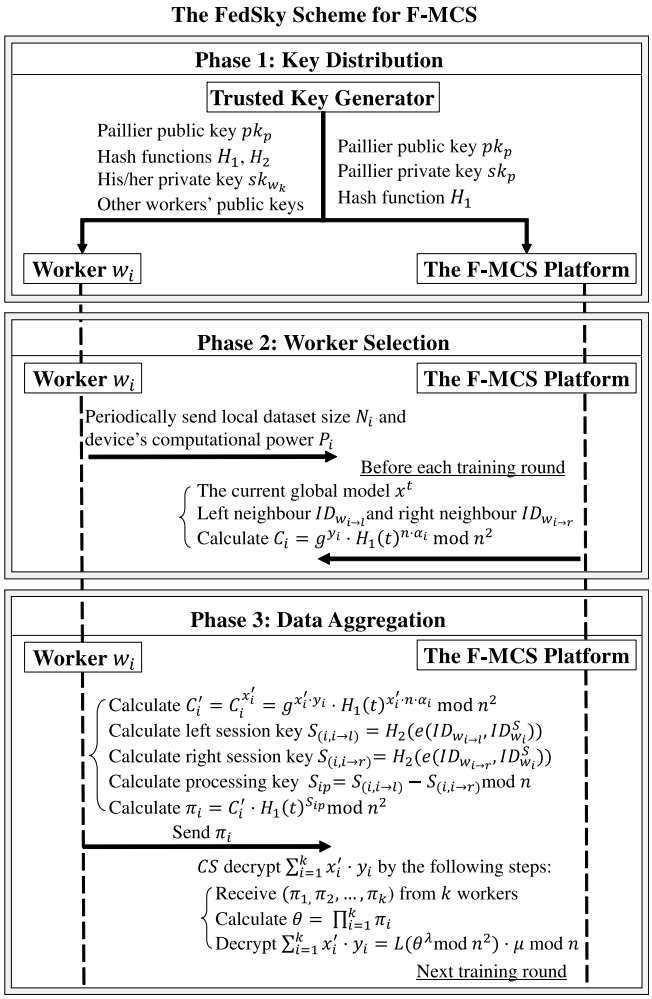


Fig. 3. Overall workflow of the FedSky scheme.

In the Paillier cryptosystem, the product of two ciphertexts can be decrypted to the sum of their corresponding plaintexts, i.e.,

$$\mathtt{D}\Big(sk, \mathtt{E}(pk, m_1) \cdot \mathtt{E}(pk, m_2) \bmod n^2\Big) = m_1 + m_2 \bmod n. \quad (1)$$

IV. Proposed Scheme

In this section, we propose a novel data aggregation scheme for F-MCS called FedSky, which mainly consists of three phases: 1) key distribution (KeyDist); 2) worker selection (WorkSel); and 3) data aggregation (DataAgg). In the KeyDist phase, $\mathcal{TKG}$ will generate keys and then distribute them to the correct entities. In the WorkSel phase, $\mathcal{P}$ will select an optimal group of workers by k-point CG-skyline. Finally, in the DataAgg phase, the selected workers need to train the model, encrypt the updated model parameter x_i, and securely send it to $\mathcal{P}$. Finally, $\mathcal{P}$ decrypts the ciphertexts to get the final aggregation results. The overall workflow of FedSky is shown in Fig. 3.

A. KeyDist Phase

First, $\mathcal{TKG}$ computes (p, q, n, λ, μ) by running $\mathtt{KeyGen}(\kappa)$ and generates the public key $pk_p = (n, g)$ and the private key $sk_p = (\lambda, \mu)$. Next, $\mathcal{TKG}$ chooses an uniform cryptographic hash function H_1, where $H_1 : \{0,1\}^* \to \mathbb{Z}_{n^2}$ and a bilinear pairing map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. After that, $\mathcal{TKG}$ broadcasts $(pk_p, H_1, e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T)$ to $\mathcal{P}$ and all the workers in the system, and securely sends sk_p to $\mathcal{P}$.

In the next step, $\mathcal{TKG}$ chooses another cryptographic hash function H_2 such that $H_2 : \{0,1\}^* \to \mathbb{Z}_n$. Then, $\mathcal{TKG}$ chooses a random number $S \in \mathbb{Z}_n^*$ as the master key. For $\forall w_i \in \mathcal{W}$, given w_i's identifier $\mathtt{ID}_{w_i}$, where $\mathtt{ID}_{w_i} = H_0(w_i)$ and $H_0 : \{0,1\}^* \to \mathbb{G}$, $\mathcal{TKG}$ calculates $\mathtt{ID}_{w_i}^S$. Last, $\mathcal{TKG}$ shares H_2 to all the workers and securely sends $\mathtt{ID}_{w_i}^S$ to w_i.

B. WorkSel Phase

In this article, each worker w_i has two attributes: 1) the size of w_i's local data set N_i and 2) w_i's mobile device's computational power P_i. Same as [8], we assumed that P_i is measured by how many data samples a worker can process per minute for the F-MCS task. Each worker w_i is required to periodically send N_i and P_i to platform $\mathcal{P}$. Before worker selection, $\mathcal{P}$ first defines a set of 2-D constraints $\mathcal{C} = \{\mathtt{Con}_1, \mathtt{Con}_2\}$ based on the F-MCS task requirements. For higher requirements (e.g., higher accuracy and less training latency), the constraints $\mathtt{Con}_1$ and $\mathtt{Con}_2$ should be chosen more strictly. Specifically, for $\mathtt{Con}_1 = [\min_1, \max_1]$ and $\mathtt{Con}_2 = [\min_2, \max_2]$, we say $\mathtt{Con}_1$ is more strictly than $\mathtt{Con}_2$ if $\min_1 > \min_2$ and $\max_1 > \max_2$. After that, $\mathcal{P}$ selects the set of workers $\mathcal{W}'$ for $\mathcal{W}' \subseteq \mathcal{W}$ whose current information matches the constraints.

Before each training round, $\mathcal{P}$ will select an optimal group of k workers from $\mathcal{W}'$ by CG-skyline in terms of N_i and P_i. The details of the proposed approach are discussed as follows.

For each worker w_i in $\mathcal{W}'$, $\mathcal{P}$ first calculates $\mathtt{Sum}(w_i)$ where $\mathtt{Sum}(w_i) = N_i + P_i$. Then, all the workers are added in a preinitialized Max Priority Queue $\mathtt{Q}_\mathcal{W}$ by $\mathtt{Sum}(w_i)$, where all the workers are sorted by $\mathtt{Sum}(w_i)$ in a descending order and the workers with larger $\mathtt{Sum}(w_i)$ values will be removed earlier from $\mathtt{Q}_\mathcal{W}$ than the workers with smaller $\mathtt{Sum}(w_i)$ values. It is worth noting that the magnitudes of P_i and N_i should be in the same level, such that the sum of P_i and N_i is not dominated by any of the two attributes. Based on the definition of skyline discussed in Section III-B, we have the following theorems.

Theorem 1: Let w_i denote a worker with N_i as the number of data samples owned by w_i and P_i as w_i's computational power. If w_i is the first worker being removed from $\mathtt{Q}_\mathcal{W}$, then w_i is a skyline worker.

Proof: Assume that there exists another worker w_i' such that $w_i' \prec w_i$. Let N_i' and P_i' denote the number of data samples owned by w_i' and w_i''s computational power, respectively. Based on the definition of skyline, w_i' is better than w_i for at least one attribute, and not worse than w_i for the rest of attributes. It is easy to know that $N_i' + P_i' > N_i + P_i$, i.e., $\mathtt{Sum}(w_i') > \mathtt{Sum}(w_i)$. Therefore, w_i' should be removed first from $\mathtt{Q}_\mathcal{W}$, which contradicts our assumption that w_i is the first worker to be removed. Therefore, such w_i' does not exist and our theorem holds. ■

Theorem 2: For $\forall w_j \in \mathcal{W}'$, all the workers who dominate w_j will be removed earlier from $\mathtt{Q}_\mathcal{W}$ than w_j.

Proof: Let w_j' denote one of the workers who dominate w_j. Again, based on the definition of skyline, it is easy to have

Algorithm 2: CG-Skyline-Based Worker Selection

Input: A worker database $\mathcal{W} = \{w_1, w_2, ..., w_K\}$ where $w_i = (N_i, P_i)$ for $i \in [1, K]$, a given set of two constraints $\mathcal{C} = \{\mathtt{Con}_1, \mathtt{Con}_2\}$

Output: A set of selected workers with size k

$\mathcal{P}$ finds the set of workers $\mathcal{W}'$ based on $\mathcal{C} = \{\mathtt{Con}_1, \mathtt{Con}_2\}$
$\mathcal{P}$ initializes an empty Max Priority Queue $\mathtt{Q}_{\mathcal{W}}$
for $\forall w_i \in \mathcal{W}'$ **do**
  $\mathcal{P}$ calculates $\mathtt{Sum}(w_i) = N_i + P_i$
  $\mathcal{P}$ adds w_i into $\mathtt{Q}_{\mathcal{W}}$ by the value of $\mathtt{Sum}(w_i)$
end
$\mathcal{P}$ initializes another empty Max Priority Queue $\mathtt{Q}_{non}$ and an empty list $\mathtt{Q}_{sky}$
$\mathcal{P}$ removes the first worker in $\mathtt{Q}_{\mathcal{W}}$ and adds this worker into $\mathtt{Q}_{sky}$
for $\forall w_i \in \mathtt{Q}_{\mathcal{W}}$ **do**
  $\mathcal{P}$ removes w_i from $\mathtt{Q}_{\mathcal{W}}$
  $\mathcal{P}$ compares the dominance relationship between w_i and all the workers in $\mathtt{Q}_{sky}$
  if $\exists\ w_j \in \mathtt{Q}_{sky}$ *s.t.* $w_j \prec w_i$ **then** $\mathcal{P}$ adds w_i into $\mathtt{Q}_{non}$;
  else $\mathcal{P}$ adds w_i into $\mathtt{Q}_{sky}$;
end
if $k' < k$ *where* k' *is number of workers in* $\mathtt{Q}_{sky}$ **then**
  $\mathcal{P}$ adds the first $k - k'$ workers from $\mathtt{Q}_{non}$ into $\mathtt{Q}_{sky}$
end
return *The workers in* $\mathtt{Q}_{sky}$

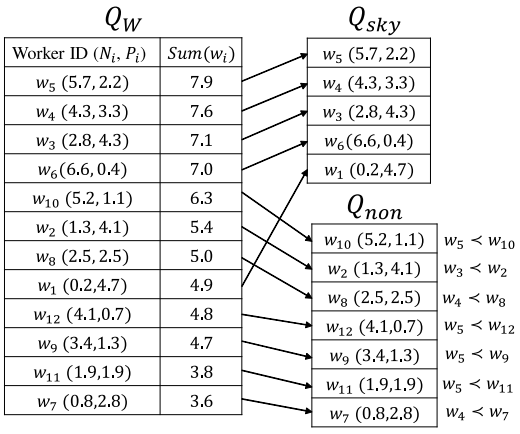


Fig. 4. Illustration example of the proposed worker selection approach for CG-skyline computation.

$\mathtt{Sum}(w_j') > \mathtt{Sum}(w_j)$. Consequently, w_j' will be removed earlier than w_j from $\mathtt{Q}_{\mathcal{W}}$. ■

Next, $\mathcal{P}$ initializes another empty Max Priority Queue $\mathtt{Q}_{\text{non}}$ and an empty list $\mathtt{Q}_{\text{sky}}$. Then, $\mathcal{P}$ goes through the workers in $\mathtt{Q}_{\mathcal{W}}$. By Property 1, the first worker will be removed from $\mathtt{Q}_{\mathcal{W}}$ and directly added into $\mathtt{Q}_{\text{sky}}$. The next removed worker from $\mathtt{Q}_{\mathcal{W}}$ will be compared with the workers(s) in $\mathtt{Q}_{\text{sky}}$. If he/she is not dominated by all the worker(s) in $\mathtt{Q}_{\text{sky}}$, then this worker will be added in $\mathtt{Q}_{\text{sky}}$. Otherwise, based on Property 2, $\mathcal{P}$ can safely add this worker in $\mathtt{Q}_{\text{non}}$. $\mathcal{P}$ repeats the steps above for the remaining workers until either the number of workers in $\mathtt{Q}_{\text{sky}}$ equals k or $\mathtt{Q}_{\mathcal{W}}$ is empty. Let k' denote the final number of workers in $\mathtt{Q}_{\text{sky}}$. If $k' < k$, then $\mathcal{P}$ removes the first $k - k'$ workers from $\mathtt{Q}_{\text{non}}$ and adds them into $\mathtt{Q}_{\text{sky}}$. In this way, we can select skyline workers as much as possible. The proposed worker selection approach is shown in Algorithm 2.

Example 4: Fig. 4 shows a simple example of the proposed worker selection approach for the CG-skyline query. The 12 workers in Example 1 are added in the Max Priority Queue $\mathtt{Q}_{\mathcal{W}}$ by $\mathtt{Sum}(w_i)$. After that, they are removed one by one from $\mathtt{Q}_{\mathcal{W}}$ and added into either $\mathtt{Q}_{\text{sky}}$ or $\mathtt{Q}_{\text{non}}$. In particular, w_5, w_4, w_3, w_6, and w_1 are added into $\mathtt{Q}_{\text{sky}}$ because they are skyline workers. The rest of the workers (i.e., $w_{10}, w_2, w_8, w_{12}, w_9, w_{11}, w_7$) are added into $\mathtt{Q}_{\text{non}}$ because they are dominated by at least one worker in $\mathtt{Q}_{\text{sky}}$. In this example, $k' = 5$. If $k \leq k'$, then the first k workers are the results for the k-point CG-skyline query. For example, if $k = 3$, then $\{w_5, w_4, w_3\}$ are selected. However, if $k > k'$, then all the workers in $\mathtt{Q}_{\text{sky}}$ plus the first $k-k'$ workers are the final results. For example, if $k = 7$, then $\{w_5, w_4, w_3, w_6, w_1, w_{10}, w_2\}$ are selected.

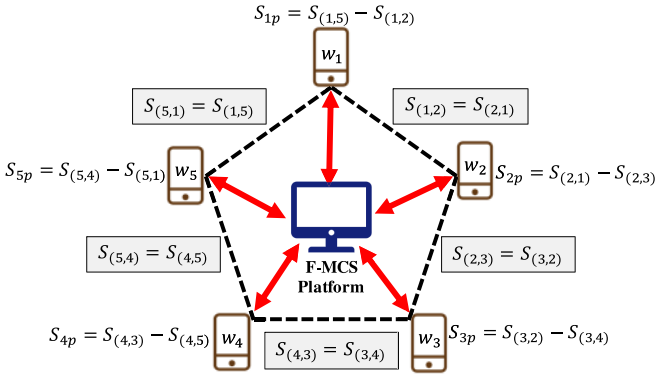


Fig. 5. Example of 5 selected workers in a loop, where the workers have no interactions with each other. In the example, each worker w_i calculates S_{ip}, then we can simply get $\sum_{i=1}^{5} S_{ip} \bmod n = (S_{1p} + S_{2p} + S_{3p} + S_{4p} + S_{5p}) \bmod n = 0$.

It is worth noting that the proposed CG-skyline technique can only select one G-skyline workers. It cannot find all the G-skyline workers that are not dominated by other groups. After worker selection, the MCS platform $\mathcal{P}$ broadcasts the global model's hyperparameters (e.g., the learning rate) to all the selected workers, which can be used by the workers to train and optimize the model in the next step.

C. DataAgg *Phase*

Let $\mathcal{W}_s$ denote a group of k selected workers, i.e., $\mathcal{W}_s = \{w_1, w_2, \ldots, w_k\}$. At the beginning of each training round, for $\forall w_i \in \mathcal{W}_s$, $\mathcal{P}$ calculates $y_i = N_i/(N_1 + N_2 + \cdots + N_k)$ as the weight for w_i's model parameter. Each y_i is rounded to three decimal places. Next, $\mathcal{P}$ amplifies each y_i by 1000 to convert it as a positive integer, i.e., $y_i \leftarrow y_i \cdot 1000$. The detailed steps of the DataAgg phase are introduced as follows.

Step 1: $\mathcal{P}$ puts all the k workers from $\mathcal{W}_s$ in a loop. In the loop, w_{i+1} is w_i's right neighbor and w_{i-1} is w_i left neighbor for $i \in [2, k-1]$. Particularly, w_k is w_1's left neighbor and w_1 is w_k's right neighbor. Fig. 5 shows an example of such a loop with five workers.

Step 2: $\mathcal{P}$ generates a random number $\alpha_i \in \mathbb{Z}_n^*$. Next, $\mathcal{P}$ sends the following information to w_i:

$$\begin{cases} \text{The current global model } x^t \text{(at round } t\text{)} \\ \text{the ID of } w_i\text{'s left neighbor } \mathtt{ID}_{w_{i\to l}} \\ \text{the ID of } w_i\text{'s right neighbor } \mathtt{ID}_{w_{i\to r}} \\ C_i = g^{y_i} \cdot H_1(t)^{n\cdot\alpha_i} \bmod n^2 \end{cases} \tag{2}$$

where $i \to l$ and $i \to r$ are indices of w_i's left neighbor and right neighbor, respectively. Moreover, t is the current training round (e.g., $t = 0, 1, 2$, etc.). Therefore, the value of t changes in each training round.

Step 3: Upon receiving the above information, w_i first calculates the average gradient g_i based on his/her local data set. Next, given a fixed learning rate η, w_i computes the updated local model parameter x_i by $x_i \leftarrow x^t - \eta \cdot g_i$. Besides, to keep the original information as much as possible, we assume that each worker w_i retains x_i to eight decimal places. So, after each local training round, w_i needs to convert x_i to a positive integer $\bar{x}_i$ in $\mathbb{Z}_n$ by $\bar{x}_i = 10^8 \cdot x_i \bmod n$.

Step 4: Next, w_i computes $\bar{C}_i$, the left session key $S_{(i,i\to l)}$, the right session key $S_{(i,i\to r)}$, and the processing key S_{ip} as

$$\begin{cases} \bar{C}_i = C_i^{\bar{x}_i} = g^{\bar{x}_i \cdot y_i} \cdot H_1(t)^{\bar{x}_i \cdot n \cdot \alpha_i} \bmod n^2 \\ S_{(i,i\to l)} = H_2\left(e\left(\mathtt{ID}_{w_i}^S, \mathtt{ID}_{w_{i\to l}}\right)\right) = H_2\left(e\left(\mathtt{ID}_{w_i}, \mathtt{ID}_{w_{i\to l}}\right)^S\right) \\ S_{(i,i\to r)} = H_2\left(e\left(\mathtt{ID}_{w_i}^S, \mathtt{ID}_{w_{i\to r}}\right)\right) = H_2\left(e\left(\mathtt{ID}_{w_i}, \mathtt{ID}_{w_{i\to r}}\right)^S\right) \\ S_{ip} = S_{(i,i\to l)} - S_{(i,i\to r)} \bmod n. \end{cases} \tag{3}$$

After that, w_i calculates π_i as follows and sends π_i to $\mathcal{P}$:

$$\begin{aligned} \pi_i &= \bar{C}_i \cdot H_1(t)^{S_{ip}} \bmod n^2 \\ &= g^{\bar{x}_i \cdot y_i} \cdot H_1(t)^{\bar{x}_i \cdot n \cdot \alpha_i} \cdot H_1(t)^{S_{ip}} \bmod n^2. \end{aligned} \tag{4}$$

Step 5: $\mathcal{P}$ needs to first check if $\exists\ w_i$ who does not submit π_i. If yes, then $\mathcal{P}$ aborts the protocol and starts a new training round. If all the k workers submit their π_i(s), $\mathcal{P}$ calculates $\theta = \prod_{i=1}^k \pi_i \bmod n^2$, which can be represented as

$$\theta = g^{\sum_{i=1}^k \bar{x}_i \cdot y_i} \cdot H_1(t)^{n \cdot \sum_{i=1}^k \bar{x}_i \cdot \alpha_i} \cdot H_1(t)^{\sum_{i=1}^k S_{ip}} \bmod n^2. \tag{5}$$

Letting $\bar{m} = \sum_{i=1}^k \bar{x}_i \cdot y_i$, $\mathcal{P}$ can decrypt $\bar{m}$ by $\mathtt{D}(sk, c)$ as

$$\bar{m} = \sum_{i=1}^k \bar{x}_i \cdot y_i \bmod n = L\left(\theta^\lambda \bmod n^2\right) \cdot \mu \bmod n. \tag{6}$$

The correctness of (6) is given as follows. First, given $\mathcal{W}_s$, we know that the sum of all the k workers' the processing keys S_{ip} is $\sum_{i=1}^k S_{ip} = S_{1p} + S_{2p} + \cdots + S_{kp} = S_{(1,k)} - S_{(1,2)} + S_{(2,1)} - S_{(2,3)} + \cdots + S_{(k,k-1)} - S_{(k,1)} \bmod n$. Based on (3) and the property of bilinear pairing, we know $S_{(i,j)} = S_{(j,i)} = H_2(e(\mathtt{ID}_{w_i}, \mathtt{ID}_{w_j})^S)$. Therefore, $\sum_{i=1}^k S_{ip} = 0 \bmod n$ as long as $k > 1$ and we can write $\sum_{i=1}^k S_{ip} = \gamma \cdot n$ for an integer γ.

Then, (5) can be written as

$$\begin{aligned} \theta &= g^{\sum_{i=1}^k \bar{x}_i \cdot y_i} \cdot H_1(t)^{n \cdot \sum_{i=1}^k \bar{x}_i \cdot \alpha_i} \cdot H_1(t)^{\gamma \cdot n} \bmod n^2 \\ &= g^{\sum_{i=1}^k \bar{x}_i \cdot y_i} \cdot H_1(t)^{n \cdot \left(\sum_{i=1}^k \bar{x}_i \cdot \alpha_i + \gamma\right)} \bmod n^2. \end{aligned} \tag{7}$$

If we consider $H_1(t)^{\sum_{i=1}^k \bar{x}_i \cdot \alpha_i + \gamma} \bmod n^2$ as a random number r, $\theta = g^{\bar{m}} \cdot r^n \bmod n^2$ is still a valid Paillier ciphertext and can be decrypted by $\mathtt{D}(sk, c)$. Since m can be either positive (less than $n/2$) or negative (larger than $n/2$), $\mathcal{P}$ can then recover m by

$$\begin{cases} m = 10^{-11} \cdot \bar{m}, & \text{if } \bar{m} < \frac{n}{2} \\ m = 10^{-11} \cdot (\bar{m} - n), & \text{otherwise.} \end{cases} \tag{8}$$

Note that as the original values of x_i and y_i have been, respectively, amplified by 10^3 and 10^8, the multiplication of 10^{-11} here is to recover the true value of $m = \sum_{i=1}^k x_i \cdot y_i$.

After recovering m, $\mathcal{P}$ can update the global model for round $t+1$ by $x^{t+1} \leftarrow \sum_{i=1}^k (N_i/N) x_i = m$. Next, $\mathcal{P}$ will start the next training round by repeating all the above-mentioned steps until the global model achieves a desired performance.

V. Security Analysis

In this section, we analyze the security properties of the FedSky scheme. Essentially, given $\forall w_i \in \mathcal{W}_s$ and his/her uploaded model parameter x_i, our analysis will focus on how FedSky is privacy preserving in protecting each w_i' uploaded model parameter x_i from being disclosed in the whole process of F-MCS. Particularly, based on the assumption that $\mathcal{P}$ does not collude with the workers, $\mathcal{P}$ can get a list of intermediate variables $(\pi_1, \pi_2, \ldots, \pi_k)$ from the k workers where $\pi_i = g^{\bar{x}_i \cdot y_i} \cdot H_1(t)^{\bar{x}_i \cdot n \cdot \alpha_i} \cdot H_1(t)^{S_{ip}} \bmod n^2$ for $i \in [1, k]$. Here, we first prove that the intermediate variable π_i is semantically secure against the chosen-plaintext attack (CPA) and then prove the security of our scheme.

A. Semantic Security of π_i

Assume that there exist a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a hypothetical challenger $\mathcal{C}$. The semantic security of π_i can be proved by a game between $\mathcal{A}$ and $\mathcal{C}$ as follows.

First, $\mathcal{C}$ runs the key generation algorithm (in Section IV-A) to generate $pk_p = (n, g)$, $sk_p = (\lambda, \mu)$ and the hash function H_1. Then, $\mathcal{C}$ sends sk_p to $\mathcal{A}$.

Upon receiving sk_p, $\mathcal{A}$ chooses two messages $x_0, x_1 \in \mathbb{Z}_n$, two random numbers $y, \alpha \in \mathbb{Z}_n^*$, and a random integer $t \in [1, 100]$. Then, $\mathcal{A}$ sends (x_0, x_1, y, α, t) to $\mathcal{C}$.

After receiving (x_0, x_1, y, α, t), $\mathcal{C}$ first flips a bit $b \in \{0, 1\}$ and generates a random number $S_p \in \mathbb{Z}_n$. Then, $\mathcal{C}$ computes $\pi_b = g^{x_b \cdot y} \cdot H_1(t)^{x_b \cdot n \cdot \alpha} \cdot H_1(t)^{S_p} \bmod n^2$ and returns π_b as a ciphertext back to $\mathcal{A}$.

Finally, $\mathcal{A}$ returns $\mathcal{C}$ a bit $b' \in \{0, 1\}$ as the guess of b. As g is the generator of $\mathbb{Z}_{n^2}^*$, there must exist an unique integer $h \in Z_n^*$ such that $H_1(t) = g^h \bmod n^2$ and h is unknown to $\mathcal{A}$. Consequently, π_b can be written as

$$\begin{aligned} \pi_b &= g^{x_b \cdot y} \cdot H_1(t)^{x_b \cdot \alpha \cdot n} \cdot g^{h \cdot S_p} \bmod n^2 \\ &= g^{x_b \cdot y + h \cdot S_p} \cdot H_1(t)^{x_b \cdot \alpha \cdot n} \bmod n^2. \end{aligned} \tag{9}$$

Here, π_b can be considered as a valid Paillier ciphertext; thus, $\mathcal{A}$ can decrypt $\tilde{m} = (x_b \cdot y + h \cdot S_p) \bmod n$ from π_b.

Next, with (x_0, x_1, y), $\mathcal{A}$ can put x_0 or x_1 into $\tilde{m}$ and calculate $\tilde{m} = x_b \cdot y + h \cdot S_p = x_{1-b} \cdot y + h' \cdot S_p \bmod n$ for $h' \in Z_n^*$. Furthermore, $\mathcal{A}$ knows that h' can be represented as $h' = (\tilde{m} - x_{1-b} \cdot y / S_p) \bmod n$. However, since S_p is generated by $\mathcal{C}$, $\mathcal{A}$ has no idea about the value of S_p. Therefore, h' is also an unknown and random value to $\mathcal{A}$. So $\mathcal{A}$ cannot distinguish between h' and h. Consequently, $\mathcal{A}$ cannot correctly decide if $b' = 0$ or 1 with a probability higher than 1/2 (i.e., the probability of a random guess). As a result, π_i is semantically secure against $\mathcal{P}$ in the proposed FedSky scheme.

B. Security of FedSky

In this section, we first define our scheme's leakage function from the perspective of $\mathcal{P}$. Next, we prove the security of the proposed scheme FedSky in the real/ideal world model.

In the FedSky scheme, the information leaked to $\mathcal{P}$ includes: 1) the Paillier public key pk_p and secret key sk_p; 2) the hash function H_1; 3) the bilinear pairing map e; 4) the list of local data set sizes $(N_1, N_2, \dots)$ and computational powers $(P_1, P_2, \dots)$; 5) the list of parameter weights $(y_1, y_2, \dots, y_k)$ from the k selected workers; and 6) $\bar{m}$ and m where $\bar{m} = \sum_{i=1}^{k} \bar{x}_i \cdot y_i \bmod n$ and $m = \sum_{i=1}^{k} x_i \cdot y_i \bmod n$. Let $\mathtt{L}(\mathcal{P})$ denote the leakage function regarding $\mathcal{P}$, $\mathtt{L}(P) = \{pk_p, sk_p, H_1, e, (N_1, N_2, \dots), (P_1, P_2, \dots), (y_1, y_2, \dots, y_k), \bar{m}, m\}$. Based on $\mathtt{L}(\mathcal{P})$, we define the real/ideal world model as follows.

Real World: In the real world, there exist a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. $\mathcal{C}$ interacts with $\mathcal{A}$ as follows.

1) KeyDist *Phase:* The challenger $\mathcal{C}$ runs the key generation algorithm in Section IV-A to create $pk_p = (n, g)$, $sk_p = (\lambda, \mu)$, the hash function H_1, and the bilinear pairing map e. Next, $\mathcal{C}$ publishes (pk_p, H_1, e), and sends sk_p to $\mathcal{A}$.
2) WorkSel *Phase:* $\mathcal{C}$ sends the list of local data set sizes $(N_1, N_2, \dots)$ and device's computational powers $(P_1, P_2, \dots)$ to $\mathcal{A}$. After that, $\mathcal{A}$ runs the k-point CG-Skyline algorithm to select k workers. Next, $\mathcal{A}$ calculates $C_i = g^{y_i} \cdot H_1(t)^{n \cdot \alpha_i} \bmod n^2$ for each worker w_i, where $i \in [1, k]$ and sends $(C_1, C_2, \dots, C_k)$ to $\mathcal{C}$.
3) *Challenge Phase:* Upon receiving $(C_1, C_2, \dots, C_k)$, $\mathcal{C}$ calculates $(\pi_1, \pi_2, \dots, \pi_k)$ based on the FedSky scheme and sends $(\pi_1, \pi_2, \dots, \pi_k)$ to $\mathcal{A}$.
4) DataAgg *Phase:* After receiving $(\pi_1, \pi_2, \dots, \pi_k)$, $\mathcal{A}$ decrypts $\bar{m} = \sum_{i=1}^{k} \bar{x}_i \cdot y_i \bmod n$ by (7) and calculates $m = \sum_{i=1}^{k} x_i \cdot y_i \bmod n$ based on (8).

Let $\mathtt{View}_{(\mathcal{A},\mathtt{Real})}$ denote $\mathcal{A}$'s view in the real world. We have $\mathtt{View}_{(\mathcal{A},\mathtt{Real})} = (\pi_1, \pi_2, \dots, \pi_k)$, where $\pi_i = g^{\bar{x}_i \cdot y_i} \cdot H_1(t)^{\bar{x}_i \cdot n \cdot \alpha} \cdot H_1(t)^{S_{ip}} \bmod n^2$ for $i \in [1, k]$.

Ideal World: In the ideal world, there exist a PPT adversary $\mathcal{A}$ and a simulator $\mathcal{S}$. Besides, $\mathcal{S}$ can get access to the leakage function $\mathtt{L}(P)$. The simulator $\mathcal{S}$ interacts with the adversary $\mathcal{A}$ as follows.

1) KeyDist *Phase:* Based on the leaked information in $\mathtt{L}(\mathcal{P})$, $\mathcal{S}$ publishes (pk_p, H_1, e) and sends sk_p to $\mathcal{A}$.
2) WorkSel *Phase:* Based on $\mathtt{L}(\mathcal{P})$, $\mathcal{S}$ sends the list of local data set sizes $(N_1, N_2, \dots)$ and device's computational powers $(P_1, P_2, \dots)$ to $\mathcal{A}$. Then, $\mathcal{A}$ runs the CG-Skyline algorithm to select k workers. Next, $\mathcal{A}$ calculates $C_i = g^{y_i} \cdot H_1(t)^{n \cdot \alpha_i} \bmod n^2$ for each worker w_i, where $i \in [1, k]$. Finally, $\mathcal{A}$ sends $(C_1, C_2, \dots, C_k)$ to $\mathcal{S}$.
3) *Challenge Phase:* In this phase, $\mathcal{S}$'s main goal is to generate a list of $(\pi'_1, \pi'_2, \dots, \pi'_k)$ to replace the list of $(\pi_1, \pi_2, \dots, \pi_k)$. Specifically, the list of $(\pi'_1, \pi'_2, \dots, \pi'_k)$ should be generated with random values based on $\mathtt{L}(P)$. Moreover, using $(\pi'_1, \pi'_2, \dots, \pi'_k)$, $\mathcal{A}$ should decrypt the same value of m as using $(\pi_1, \pi_2, \dots, \pi_k)$ in the real world. $\mathcal{S}$ generates $(\pi'_1, \pi'_2, \dots, \pi'_k)$ as follows.
   a) First, $\mathcal{S}$ randomly selects $k-1$ numbers $(x'_1, x'_2, \dots, x'_{k-1})$ such that $x'_i \in \mathbb{Z}_n^*$ for $i \in [1, k-1]$. Then, given $m = \sum_{i=1}^{k} x_i \cdot y_i \bmod n$ and $(y_1, y_2, \dots, y_k) \in \mathtt{L}(P)$, $\mathcal{S}$ calculates x'_k as
   $$x'_k = \frac{\left(m - \sum_{j=1}^{k-1} x'_j \cdot y_j\right)}{y_k} = \frac{\left(\sum_{i=1}^{k} x_i \cdot y_i - \sum_{j=1}^{k-1} x'_j \cdot y_j\right)}{y_k} \bmod n. \quad (10)$$
   In this case, it is easy to have $m = \sum_{i=1}^{k} x_i \cdot y_i = \sum_{i=1}^{k} x'_i \cdot y_i \bmod n$. Next, $\mathcal{S}$ calculates a list of $(\bar{x_1}', \bar{x_2}', \dots, \bar{x_k}')$, where $\bar{x_i}' = 10^8 \cdot x'_i \bmod n$ for $i \in [1, k]$.
   b) $\mathcal{S}$ randomly chooses a sequence of k numbers $(\alpha_1, \alpha_2, \dots, \alpha_k)$ and another sequence of $k-1$ numbers $(S'_{1p}, S'_{2p}, \dots, S'_{(k-1)p})$, where $\alpha_i, S'_{jp} \in \mathbb{Z}_n^*$ for $i \in [1, k], j \in [1, k-1]$. Next, $\mathcal{S}$ calculates S'_{kp} as
   $$S'_{kp} = -\left(S'_{1p} + S'_{2p} + \cdots + S'_{(k-1)p}\right) \bmod n. \quad (11)$$
   Therefore, it is easy to have $\sum_{i=1}^{k} S'_{ip} = 0 \bmod n$. Hence, we can write $\sum_{i=1}^{k} S'_{ip} = \gamma' \cdot n$ for an integer γ'.
   c) $\mathcal{S}$ can finally generate the list of $(\pi'_1, \pi'_2, \dots, \pi'_k)$, where $\pi'_i = g^{\bar{x_i}' \cdot y_i} \cdot H_1(t)^{\bar{x_i}' \cdot \alpha_i \cdot n} \cdot H_1(t)^{S'_{ip}} \bmod n^2$ for $i \in [1, k]$. Finally, $\mathcal{S}$ sends $(\pi'_1, \pi'_2, \dots, \pi'_k)$ to $\mathcal{A}$.
4) DataAgg *Phase:* After receiving $(\pi'_1, \pi'_2, \dots, \pi'_k)$, $\mathcal{A}$ first calculates $\theta' = \prod_{i=1}^{k} \pi'_i \bmod n^2$ as
   $$\begin{aligned}\theta' &= g^{\sum_{i=1}^{k} \bar{x_i}' \cdot y_i} \cdot H_1(t)^{n \cdot \sum_{i=1}^{k} \bar{x_i}' \cdot \alpha_i} \cdot H_1(t)^{\sum_{i=1}^{k} S'_{ip}} \\ &= g^{\sum_{i=1}^{k} \bar{x_i}' \cdot y_i} \cdot H_1(t)^{n \cdot \sum_{i=1}^{k} \bar{x_i}' \cdot \alpha_i} \cdot H_1(t)^{\gamma' \cdot n} \\ &= g^{\sum_{i=1}^{k} \bar{x_i}' \cdot y_i} \cdot H_1(t)^{n \cdot \left(\sum_{i=1}^{k} \bar{x_i}' \cdot \alpha_i + \gamma'\right)} \bmod n^2. \quad (12)\end{aligned}$$
   Similar to the methods in Section IV-C, by considering θ' as a valid Paillier ciphertext, $\mathcal{A}$ can decrypt $\bar{m}' = \sum_{i=1}^{k} \bar{x_i}' \cdot y_i \bmod n$. Finally, by (8), $\mathcal{A}$ can calculate $m' = \sum_{i=1}^{k} x'_i \cdot y_i = \sum_{i=1}^{k} x_i \cdot y_i \bmod n$, i.e., $\mathcal{A}$ can get the same result as in the real world.

Let $\mathtt{View}_{(\mathcal{A},\mathtt{Ideal})}$ denote $\mathcal{A}$'s view in the ideal world. We have $\mathtt{View}_{(\mathcal{A},\mathtt{Ideal})} = (\pi'_1, \pi'_2, \dots, \pi'_k)$, where $\pi'_i = g^{\bar{x_i}' \cdot y_i} \cdot H_1(t)^{\bar{x_i}' \cdot n \cdot \alpha} \cdot H_1(t)^{S'_{ip}} \bmod n^2$ for $i \in [1, k]$.

In Section V-A, we have already proved the semantic security of π_i against CPA. Therefore, $\mathcal{A}$'s view in real world, i.e., $\mathtt{View}_{(\mathcal{A},\mathtt{Real})} = (\pi_1, \pi_2, \dots, \pi_k)$, is computationally indistinguishable from its view in the ideal world, i.e., $\mathtt{View}_{(\mathcal{A},\mathtt{Ideal})} = (\pi'_1, \pi'_2, \dots, \pi'_k)$. As a result, the proposed FedSky scheme is secure with the leakage function $\mathtt{L}(P)$.

VI. Performance Evaluation

In this section, we study the performance of FedSky in terms of time consumption and model accuracy. Particularly, first, we simulated 1000 heterogeneous workers with different computational powers in an F-MCS environment. Then,

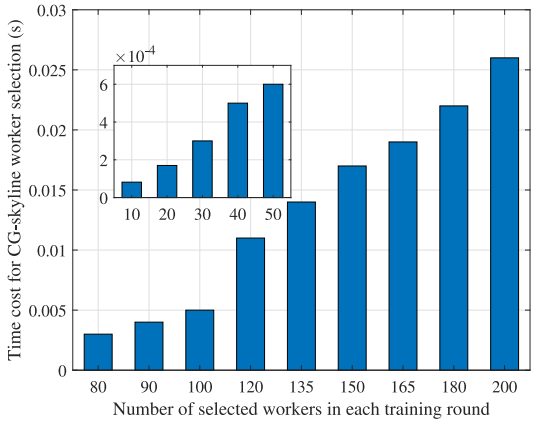


Fig. 6. Running time for CG-skyline-based worker selection with different number of selected workers in each training round.

we compare the time consumption for the FedSky scheme across various experimental settings, including the number of selected workers and the number of training rounds. Second, with the simulated F-MCS environment, we deployed an object classification task on the MNIST data set with FedSky and compared the model accuracy between independent and identical distribution (IID) data and non-independent and identical distribution (Non-IID) data. The details of performance evaluation are presented as follows.

A. Experimental Setting

In our experiment, we simulated an F-MCS environment including 1000 heterogeneous workers (i.e., $K = 1000$) with different computational powers. We assumed that P_i follows the Gaussian distribution with 50 samples/min as the mean value and sd the as the standard deviation, where $sd = 5$ or 15 in the experiment.

$\mathcal{P}$ initialized an object classification task on the MNIST data set. The MNIST data set is a large database of handwritten digits commonly used for training image processing systems. In the simulation, we assumed that every worker has N_i handwritten digits images, where N_i also follows the Gaussian distribution with 500 as the mean value as 50 as the standard deviation. Then, by following the experimental setup in [8] and [29], we designed two ways for distributing the training samples to the workers.

1) *IID:* There are ten classes (e.g., digits 0–9) in the MNIST data set. In the IID setting, each worker is randomly sampled N_i images across the ten classes from the training data set.
2) *Non-IID:* In the Non-IID setting, each worker is still sampled N_i image samples, yet 70% of which come from a dominant class and the remaining 30% belong to other classes. For instance, if $N_i = 500$ for a worker w_i, then she owned 350 images from class "0," while the remaining 150 images are randomly distributed among class "1" to "9."

In each training round, $\mathcal{P}$ selects k workers from 1000 candidates by CG-skyline, where k varies from 10 to 200 in the experiment. There are two constraints $\texttt{Con}_1$ and $\texttt{Con}_2$ for N_i and P_i, respectively. We set $\texttt{Con}_1 = [100, +\infty]$ and $\texttt{Con}_2 = [10, +\infty]$. In the experiment, the security parameter κ is set to 512, i.e., $|p| = |q| = 512$ and $n = 1024$.

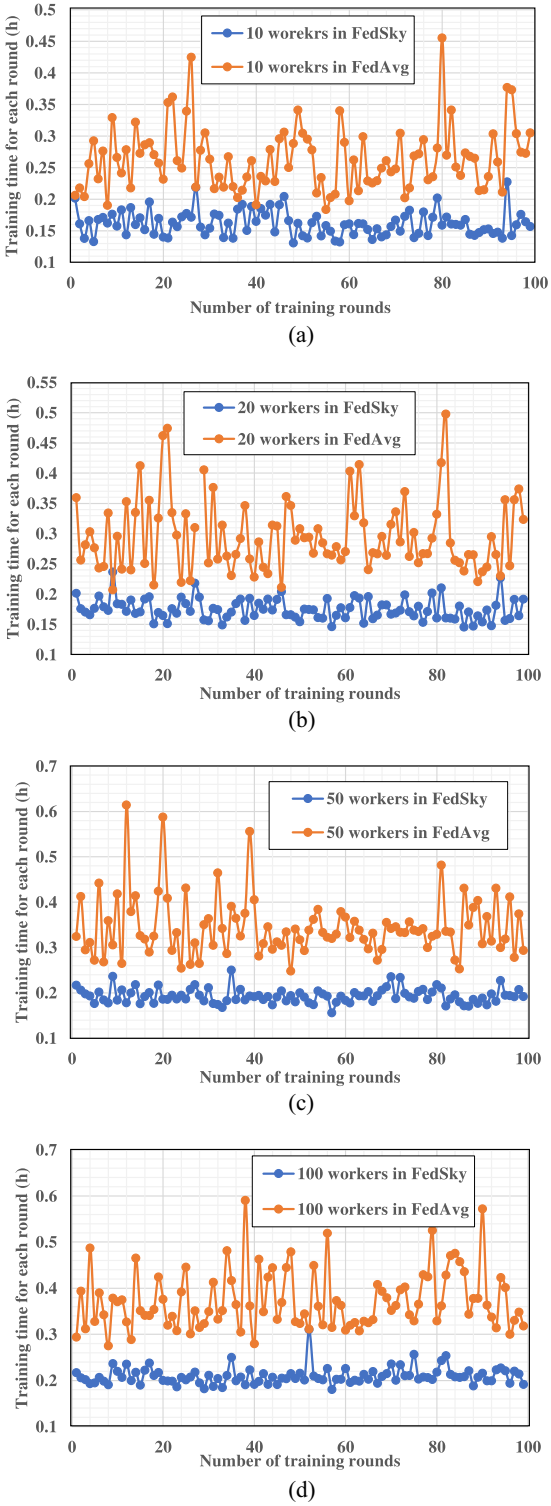
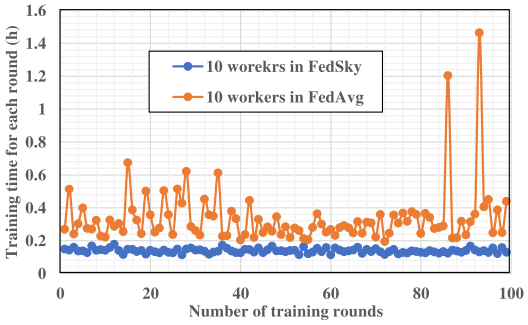
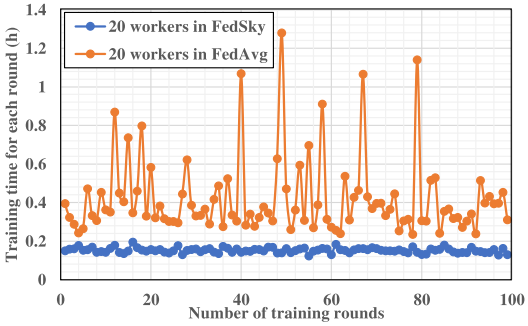


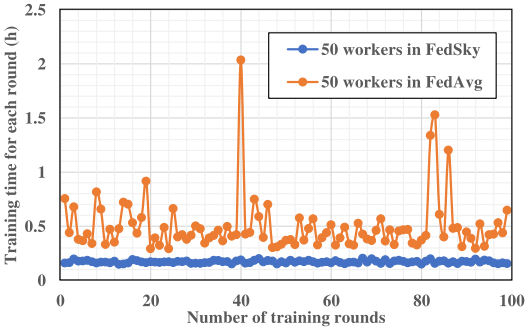
Fig. 7. Comparison of training time (h) per round between FedSky and FedAvg for different numbers of workers with $sd = 5$. (a) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 5$ for 10 workers. (b) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 5$ for 20 workers. (c) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 5$ for 50 workers. (d) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 5$ for 100 workers.



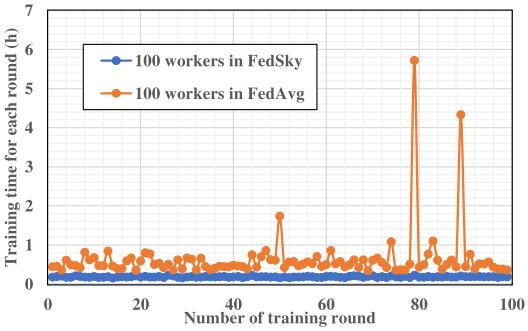
(a)



(b)



(c)



(d)

Fig. 8. Comparison of workers' local training time (h) per round between FedSky and FedAvg for different numbers of workers with $sd = 15$. (a) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 15$ for 10 workers. (b) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 15$ for 20 workers. (c) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 15$ for 50 workers. (d) Comparison of workers' local training time (h) per round between FedSky and FedAvg with $sd = 15$ for 100 workers.

Moreover, we deployed a standard convolutional neural network (CNN) as the global model. Particularly, in the CNN model, there are three convolution layers with ReLU as the activation function, followed by one fully connected layer activated by a Softmax function. The hyperparameters for training the model are as follows: the minibatch size is 50, the number of epochs in each round is 10, the initial learning rate is 0.01 and 0.9 for learning rate decay, and the number of training rounds is 100. The designed CNN model is implemented using the PyTorch API for the FL framework.

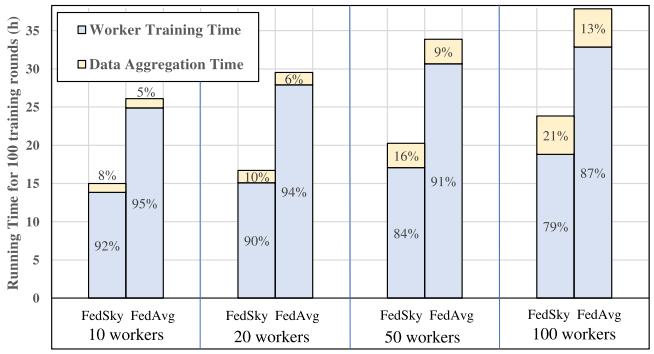


Fig. 9. Breakdowns of overall running time for 100 training rounds in terms of local training and data aggregation $k = 10, 20, 50,$ and 100.

All the experiments are implemented with Python programming language conducted experiments on a machine with an Intel Core i7-6700 CPU @3.60-GHz, 16-GB RAM, and Windows 10 operating system.

B. Evaluation Results

In this section, we describe the evaluation results in the experiments. Fig. 6 compares the running time (in second) for worker selection based on the CG-skyline technique with different k. The CG-skyline is performed in terms of N_i and P_i. For each value of k, the experiment is conducted 10 000 times and the average result is reported. From Fig. 6, we can see that the running time increases as the increasing of k. Clearly, worker selection can be finished in a very short period of time (in the order of 10^{-4} s to 10^{-2} s). Compared to the traditional FedAvg scheme, although we add an extra worker selection phase in each training round, the efficiency of the overall process is not affected.

Figs. 7 and 8 compare the running time (in hour) for each training round between FedSky and FedAvg for different number of workers with $sd = 5$ and 15, respectively. As mentioned in Section I, we consider the synchronous training protocol, where no worker can proceed to the next training round until all the k workers' data have been uploaded in the current round. Each training round's running time is restricted by the worker who has the poorest computational power. Therefore, in Figs. 7 and 8, the time required by the worker with the poorest computational power is the running time for each training round.

Notably, the standard deviation sd represents the uncertainty and heterogeneity of workers' computational powers. The larger sd indicates the higher heterogeneity among workers' computational capacities. From Figs. 7 and 8, we can summarize that, under all the experimental settings, the running time for each training round in FedSky is lower than that in FedSky, indicating the efficiency of FedSky. Specifically, we can see that for FedAvg, the variance of running time is proportional to both the number of selected workers k and the standard

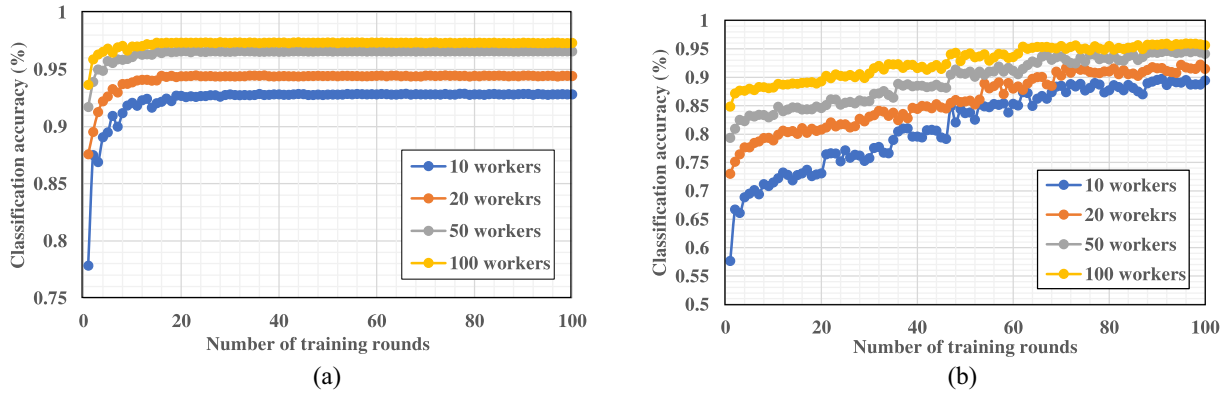


Fig. 10. Performance comparison of the deep learning model between the IID and Non-IID data settings with $k = 10, 20, 50$, and 100. (a) Accuracy of the deep learning model with different number of workers on IID MNIST data. (b) Accuracy of the deep learning model with different number of workers on non-IID MNIST data.

deviation for computational power *sd*. Under the experimental setting of $k = 100$ and $sd = 15$, the maximum training time for FedSky can be as high as 6 h. On the other hand, we note that neither the number of selected workers k nor the standard deviation *sd* can greatly affect the performance of FedSky. For all the experimental settings, FedSky takes about 0.2 h to finish one training round, indicating the reliability and stability of FedSky in terms of time consumption. The performance difference between FedSky and FedAvg is even more significant as the increase of k and *sd*.

In Fig. 9, we compare the breakdowns of the overall running time for 100 training rounds in terms of workers' local training time and the platform's aggregation time. Same as [10], we found that for both FedSky and FedAvg, workers' local training time dominates the overall time consumption of the FL process. Especially, in FedAvg, the F-MCS platform can perform data aggregation very quickly. However, most of its time is wasted on waiting for the last worker to send the encrypted model updates. For instance, in FedAvg, the actual computation time for the platform $\mathcal{P}$ to aggregate and decrypt data only accounts for 5% to 13% of the whole time span. In contrast, FedSky can greatly increase workers' training efficiency and reduce $\mathcal{P}$'s idle time. For instance, when $k = 100$, $\mathcal{P}$'s idle time will be decreased from 87% (in FedAvg) to 79% (in FedSky). Overall speaking, by introducing an extra worker selection phase, FedSky can select qualified workers in terms of local data size and computational power. The time consumption of worker selection in FedSky is quite low (in the order of 10^{-2} to 10^{-4} s). But the overall efficiency of the training process can be greatly improved.

Fig. 10 shows the comparison of model performance with the different numbers of selected workers for both IID and Non-IID settings. We can see that for both cases, selecting more workers can lead to better classification accuracies. Besides, we note that when $k = 100$, it takes only about ten rounds to achieve 96% target accuracy in the IID setting. However, it takes about 100 rounds to achieve similar accuracy in the Non-IID setting. Notably, data samples across mobile workers are usually Non-IID in the real-world F-MCS scenarios. Hence, the F-MCS service needs more training rounds to achieve a satisfactory result.

VII. Related Work

A. Worker Selection in MCS

Worker selection is always a fundamental problem in MCS applications. Tong *et al.* [30] proposed a microtask allocation problem in spatial crowdsourcing. In their study, they focused on the simple tasks any individual can perform with the smart devices. Specifically, they used the online maximum weighted bipartite matching problem as the baseline algorithm and introduced a two-phase-based framework to improve the baseline's efficiency. Zhao *et al.* [31] studied a destination-aware task assignment problem that concerns the optimal strategy of assigning each task to proper workers. Consequently, the total number of completed tasks can be maximized while all workers can reach their destinations before deadlines after performing the assigned tasks. Zhao *et al.* [32] focused on a novel spatial crowdsourcing problem, namely, the predictive task assignment, which aimed to maximize the number of assigned tasks by considering both current and future workers in a dynamic way. Zhang *et al.* [33] proposed a privacy-preserving worker selection scheme for the MCS platform based on the probabilistic skyline query over sliding windows. Their work can continuously select reliable workers in terms of working experience, expiry time, and trustability without revealing workers' sensitive information. Unlike the above studies, in this article, we considered the worker selection problem under the F-MCS circumstance, where the task is complicated (e.g., image classification or object detection) and the model needs to be trained across workers' local data sets.

B. Privacy-Preservation Federated Learning

Zhang *et al.* [10] proposed a HE-based scheme for *cross-silo* FL. Instead of encrypting individual gradients, they encoded a batch of quantized gradients into a long integer and encrypted it in one go. The experimental results showed that the proposed method can significantly increase the training speedup while reducing the communication overhead. Li *et al.* [15] designed a *cross-silo* FL scheme to detect cyberthreats in industrial cyber–physical systems based on the Paillier cryptosystem. Their system model contains one cloud server and multiple physical infrastructures (i.e., workers). The

cloud server aggregates the encrypted model updates and sends the encrypted model to the workers. Each worker holds the private key and decrypts the model to build a local intrusion detection system. Mandal and Gong [13] presented a privacy-preserving system for training linear and logistic regression models. Their proposed model can guarantee data and model privacy as well as ensuring robustness to users dropping out in the network. The core of their training protocols is a secure multiparty global gradient computation on alive users' data. Their security goal is to protect the trained model from being leaked to the user, but we do not guarantee this assumption in our work. Overall, most of the existing solutions focus on privacy-preserving data aggregation in the *cross-silo* FL system for different applications.

VIII. Conclusion

In this article, we have proposed a privacy-preserving scheme for F-MCS applications, called FedSky. By extending the classic FedAvg algorithm, FedSky selects qualified workers based on the CG-skyline technique and securely aggregate model updates for training the global model. Particularly, compared to FedAvg, our scheme considers the workers' dynamics and heterogeneous natures. It can significantly improve the efficiency of model training process in F-MCS. Besides, we have designed a novel privacy-preserving data aggregation protocol based on the additive HE system. This protocol is exclusively designed for the *cross-device* FL setting and requires no interactions between workers during the model training process. Security analysis demonstrated that the proposed scheme is privacy preserving. Extensive experiments have been conducted on a real-world image classification task. The comparison results validated the efficiency of worker selection and the robustness of FedSky against heterogeneous workers. For future work, we will extend FedSky in the real-world object detection problem.

References

[1] X. Zhang, R. Lu, J. Shao, H. Zhu, and A. A. Ghorbani, "Secure and efficient probabilistic skyline computation for worker selection in MCS," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11524–11535, Dec. 2020.

[2] A. Overeem, J. C. R. Robinson, H. Leijnse, G.-J. Steeneveld, B. P. K. Horn, and R. Uijlenhoet, "Crowdsourcing urban air temperatures from smartphone battery temperatures," *Geophys. Res. Lett.*, vol. 40, no. 15, pp. 4081–4085, 2013.

[3] C. Zhang, L. Zhu, C. Xu, X. Du, and M. Guizani, "A privacy-preserving traffic monitoring scheme via vehicular crowdsourcing," *Sensors*, vol. 19, no. 6, p. 1274, 2019.

[4] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surveys*, vol. 48, no. 1, pp. 1–31, 2015.

[5] S. Tiwari and S. Kaushik, "Information enrichment for tourist spot recommender system using location aware crowdsourcing," in *Proc. IEEE 15th Int. Conf. Mobile Data Manage.*, vol. 2, 2014, pp. 11–14.

[6] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2419–2465, 3rd Quart., 2019.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[8] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–7.

[9] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: arXiv:1912.04977.

[10] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.

[11] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6532–6542, Oct. 2020.

[12] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1333–1345, 2018.

[13] K. Mandal and G. Gong, "PrivFL: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks," in *Proc. ACM SIGSAC Conf. Cloud Comput. Security Workshop*, 2019, pp. 57–68.

[14] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," in *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13. San Rafael, CA, USA: Morgan Claypool, 2019, pp. 1–207.

[15] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.

[16] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[17] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha: An efficient approach for privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Security*, 2019, pp. 13–23.

[18] Y. Dong, X. Chen, L. Shen, and D. Wang, "EaSTFly: Efficient and secure ternary federated learning," *Comput. Security*, vol. 94, Jul. 2020, Art. no. 101824.

[19] W. Schneble and G. Thamilarasu, "Attack detection using federated learning in medical cyber-physical systems," in *Proc. 28th Int. Conf. Comput. Commun. Netw.*, 2019, pp. 1–8.

[20] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," 2016. [Online]. Available: http://arxiv.org/abs/1602.05629.

[21] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 421–430.

[22] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang, "Finding pareto optimal groups: Group-based skyline," *Proc. VLDB Endowment*, vol. 8, no. 13, pp. 2086–2097, 2015.

[23] J. Liu, L. Xiong, J. Pei, J. Luo, H. Zhang, and W. Yu, "Group-based skyline for pareto optimal groups," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 7, pp. 2914–2929, Jul. 2021.

[24] L. Chen, B. Cui, and H. Lu, "Constrained skyline query processing against distributed data sites," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 2, pp. 204–217, Feb. 2011.

[25] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2003, pp. 467–478.

[26] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 546–555.

[27] H. Zhu, Q. Wei, X. Yang, R. Lu, and H. Li, "Efficient and privacy-preserving online fingerprint authentication scheme over outsourced data," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 576–586, Apr.–Jun. 2018.

[28] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 9, pp. 1621–1631, Sep. 2012.

[29] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1698–1707.

[30] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, 2016, pp. 49–60.

[31] Y. Zhao, Y. Li, Y. Wang, H. Su, and K. Zheng, "Destination-aware task assignment in spatial crowdsourcing," in *Proc. ACM Conf. Inf. Knowl. Manag.*, 2017, pp. 297–306.

[32] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou, "Predictive task assignment in spatial crowdsourcing: A data-driven approach," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, 2020, pp. 13–24.

[33] X. Zhang, R. Lu, J. Shao, H. Zhu, and A. A. Ghorbani, "Continuous probabilistic skyline query for secure worker selection in mobile crowdsensing," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11758–11772, Jul. 2021.

Xichen Zhang received the B.E. degree from Changsha University of Science and Technology, Changsha, China, in 2010, the M.S. degree in computer science from the Canadian Institute for Cybersecurity (CIC), Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, in 2018, where he is currently pursuing the Ph.D. degree.

He worked as a Research Assistant with CIC. His research interests are data mining in cybersecurity, privacy-enhancing technologies, and IoT-big data security and privacy.

Rongxing Lu (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012.

He worked as a Postdoctoral Fellow with the University of Waterloo, from May 2012 to April 2013. He is currently an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Fredericton, NB, Canada. Before that, he worked as an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from April 2013 to August 2016. His research interests include applied cryptography, privacy-enhancing technologies, and IoT-big data security and privacy. He has published extensively in his areas of expertise.

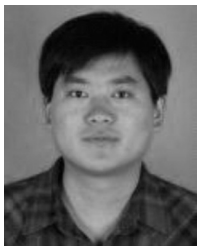
Dr. Lu was the recipient of the eight best (student) paper awards from some reputable journals and conferences. He was awarded the most prestigious Governor General's Gold Medal. He won the 8th IEEE Communications Society Asia–Pacific Outstanding Young Researcher Award in 2013. He is the Winner of 2016–2017 Excellence in Teaching Award from FCS, UNB. He serves as the Vice-Chair (Conferences) for IEEE Communications and Information Security Technical Committee.

Jun Shao (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008.

He was a Postdoctoral with the School of Information Sciences and Technology, Pennsylvania State University, State College, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His research interests include network security and applied cryptography.

Fengwei Wang (Student Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 2016, where he is currently pursuing the Ph.D. degree with the School of Cyber Engineering.

His research interests include the areas of applied cryptography, cybersecurity, and privacy.



Hui Zhu (Senior Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, China, in 2005, and the Ph.D. degree from Xidian University in 2009.

He was a Research Fellow with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2013. Since 2016, he has been a Professor with the School of Cyber Engineering, Xidian University. His current research interests include applied cryptography, data security, and privacy.

Ali A. Ghorbani (Senior Member, IEEE) is currently a Professor of Computer Science, a Tier 1 Canada Research Chair of Cybersecurity, and the Director of the Canadian Institute for Cybersecurity, University of New Brunswick, Fredericton, NB, Canada, which he established in 2016. He has held a variety of academic positions for the past 39 years. He served as the Dean of the Faculty of Computer Science, University of New Brunswick from 2008 to 2017. He has spent over 29 years of his 39-year academic career, carrying out fundamental and applied research in machine learning, cybersecurity, and critical infrastructure protection. He is the Co-Inventor on three awarded and one filed patent in the fields of cybersecurity and Web intelligence and has published over 280 peer-reviewed articles during his career. He has supervised over 190 research associates, postdoctoral fellows, and students during his career. He developed several technologies adopted by high-tech companies and co-founded three startups, Sentrant Security, EyesOver Technologies, and Cydarien Security in 2013, 2015, and 2019. His book *Intrusion Detection and Prevention Systems: Concepts and Techniques*, was published by Springer in October 2010.

Prof. Ghorbani is the recipient of the 2017 Startup Canada Senior Entrepreneur Award and the Canadian Immigrant Magazine's RBC top 25 Canadian immigrants of 2019. He is also the founding Director of the Laboratory for Intelligence and Adaptive Systems Research. He is the Co-Founder of the Privacy, Security, Trust Network in Canada and its annual international conference and served as the Co-Editor-in-Chief of *Computational Intelligence: An International Journal* from 2007 to 2017.