Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.085946

ARTICLE

Privacy-Preserving Collaborative Task Allocation for Multi-Skill Mobile Crowdsensing

Jie Li, Fuyuan Song* and Qin Jiang

School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, China

*Corresponding Author: Fuyuan Song. Email: fysong@nuist.edu.cn

Received: 21 May 2026; Accepted: 17 July 2026; Published: 13 August 2026

ABSTRACT: Mobile crowdsensing enables large-scale sensing tasks through smart devices carried by users and has been widely applied in intelligent transportation and environmental monitoring. With the increasing complexity of sensing tasks, many tasks require the collaboration of multiple workers with different skills. However, both task-required skills and worker skills are privacy-sensitive, and directly exposing them to the platform may reveal task intentions and workers' capability profiles. To address this issue, this paper proposes Dual-Fog Privacy-Preserving Multi-skill Task Allocation (DPMTA), a privacy-preserving task allocation scheme for multi-skill collaborative tasks. DPMTA adopts a dual-fog architecture to separately protect location privacy and skill privacy. Specifically, task and worker locations are perturbed by differential privacy, while skill information is split by XOR secret sharing and distributed to two non-colluding fog servers. With Beaver triple-assisted secure Boolean computation, DPMTA enables skill matching, coverage updating, and collaborative worker selection without revealing plaintext skills. In addition, a random permutation mechanism is introduced to hide the direct mapping between skill bit positions and semantic skill labels. Security analysis shows that DPMTA effectively protects skill and location privacy. Meanwhile, the experimental evaluation demonstrates that DPMTA maintains a high task allocation success rate while keeping communication and computation overhead within acceptable limits.

KEYWORDS: Privacy-preserving; task allocation; secret sharing; mobile crowdsensing (MCS)

1 Introduction

With the rapid development of mobile Internet, the Internet of Things, and intelligent terminal devices, Mobile CrowdSensing (MCS) has gradually emerged as an important paradigm for large-scale data acquisition and environmental sensing [1]. Compared with traditional sensing approaches that rely on the deployment of fixed sensors, MCS fully exploits smart devices carried by ordinary users to achieve dynamic perception of target areas, events, or objects through distributed and crowdsourced data collection. Benefiting from its advantages of low deployment cost, wide coverage, and strong scalability, MCS has demonstrated broad application prospects in fields such as intelligent transportation, environmental monitoring, and urban governance [2–5]. As a key component of MCS systems, task allocation directly affects task completion efficiency, sensing quality, and the overall service capability of the platform, and has therefore remained a major research focus in this field [6,7]. In recent years, with the continuous expansion of MCS application scenarios, task types have gradually evolved from relatively simple single-objective sensing tasks to more complex composite tasks. Many real-world tasks can no longer be accomplished independently by a single worker, but instead require multiple participants with different capabilities or

skills to collaborate. For example, a complex inspection, monitoring, or emergency sensing task may simultaneously involve image acquisition, equipment inspection, anomaly reporting, and environmental recording, while an individual worker usually possesses only a subset of these required skills. Consequently, the platform must select a collaborative worker set from the candidate pool and accomplish the task through the complementary capabilities of multiple workers. This implies that the task allocation problem is no longer confined to conventional single-worker matching, but is gradually shifting toward multi-skill joint selection for collaborative task fulfillment [8].

Nevertheless, existing studies on task allocation in Mobile CrowdSensing still mainly focus on factors such as location [9–11], user preference [12,13] and budget cost [14]. In such studies, spatial reachability, the preference matching relationship between workers and tasks, and budget constraints are typically regarded as the primary decision-making criteria, whereas the skill structure of workers and its impact on task completion are often insufficiently modeled. For simple tasks that can be completed by a single worker, this modeling paradigm may be feasible to some extent. However, for complex tasks that require the collaboration of multiple workers, considering only location, preference, or cost is no longer sufficient to accurately capture the actual requirements of task allocation. This is because whether such a task can be successfully assigned no longer depends solely on whether a particular worker is suitable, but rather on whether the skill union of the selected collaborative worker set can cover the task requirements. In other words, in complex collaborative task scenarios, skill-related factors have become a critical dimension affecting the outcome of task allocation.

To address the above issues, a limited number of recent studies have begun to consider multi-skilled participants or collaborative task scenarios. Han et al. [15] investigated the problem of online organizing large-scale heterogeneous tasks and multi-skilled participants, and pointed out that the complex heterogeneity of tasks and participants across temporal, spatial, and skill dimensions significantly increases the difficulty of task allocation. To improve the efficiency of online sharing and matching, they further designed data organization mechanisms based on hierarchical trees and time-series queues. This work demonstrates that multi-skilled participants have become an important factor that cannot be neglected in Mobile CrowdSensing task allocation, although its primary focus remains on improving organization and matching efficiency in large-scale settings. In addition, Wei et al. [16] studied the problem of group task recommendation in Mobile CrowdSensing and observed that many collaborative tasks require multiple participants with different sensing capabilities to work jointly. Accordingly, the platform needs to first construct a participant group that satisfies the task requirements and then perform task recommendation. From the perspectives of group formation and preference modeling, this study reveals a fundamental characteristic of collaborative tasks, namely, that the decision object in task allocation has expanded from an individual worker to a worker set with complementary capabilities. Furthermore, Fang et al. [17] considered the multi-skill task allocation scenario and proposed a skill-aware task allocation method under local differential privacy, which introduces skill diversity indicators and skill contribution values to improve allocation performance while preserving location privacy.

Although existing studies have examined skill-related factors and collaboration requirements from different perspectives, the issue of skill privacy preservation in multi-skill collaborative tasks remains insufficiently explored. In multi-skill collaborative task allocation, the platform needs to select workers according to the correspondence between task-required skills and worker skills. However, the set of required skills may reflect the task publisher's business objectives or service intentions, while a worker's skill vector may reveal the participant's professional competence structure and long-term service expertise. Therefore, such information is inherently privacy-sensitive [18]. If the platform directly accesses plaintext skill data, privacy leakage may occur. In contrast, if skill information cannot be utilized, the effective allocation of multi-skill collaborative tasks will be hindered. Therefore, realizing collaborative worker selection without

exposing plaintext task-required skills and worker skills remains an important problem in multi-skill task allocation research.

To address the above issue, this paper investigates the problem of privacy-preserving task allocation for multi-skill collaborative tasks in Mobile CrowdSensing and proposes a dual-fog collaborative privacy-preserving task allocation scheme, termed Dual-Fog Privacy-Preserving Multi-skill Task Allocation (DPMTA). Centered on skill privacy preservation in multi-skill collaborative tasks, the proposed scheme supports candidate screening and collaborative worker selection while preventing task-required skills and worker skills from being exposed to the platform in plaintext form. By separating the processing of location information and skill information and incorporating a dual-fog collaborative computation mechanism, the proposed scheme enables effective multi-skill collaborative task allocation under skill privacy protection.

The main contributions of this paper are summarized as follows:

- In view of the fact that complex tasks in Mobile CrowdSensing often require the collaboration of multiple workers, we propose a privacy-preserving task allocation model for multi-skill collaborative tasks. Specifically, the conventional single-worker matching problem is extended to a multi-skill collaborative coverage scenario, where the coverage relationship between task-required skills and worker skills is formally characterized, and privacy protection is incorporated into the task allocation process, thereby laying the problem-modeling foundation for subsequent privacy-preserving algorithm design.
- We propose a dual-fog collaborative privacy-preserving task allocation scheme, namely DPMTA. The scheme employs XOR secret sharing and Beaver-triple-assisted secure Boolean computation to realize skill matching, coverage determination, and collaborative worker selection without exposing plaintext task-required skills or worker skills. In addition, by integrating a consistent random permutation mechanism and location perturbation, the scheme hides the semantic mapping between skill indices and original skill labels in intermediate coverage results and supports candidate worker screening under privacy constraints.
- We evaluate the proposed scheme from the perspectives of privacy preservation and allocation performance. Security analysis shows that, under the assumption that the two fog servers do not collude, DPMTA can effectively protect task-required skills, worker skill information, and location information. Experimental results demonstrate that the proposed scheme can maintain a relatively high task allocation success rate while preserving privacy, and is feasible in terms of both communication overhead and computational cost.

The remainder of this paper is organized as follows: Section 2 introduces the system and security models along with design objectives. Section 3 presents the proposed DPMTA scheme. Sections 4 and 5 provide security analysis and performance evaluation, respectively. Finally, Section 6 concludes this paper.

2 Models and Problem Formulation

In this section, we first introduce the system model and the threat model; then we formally model the privacy-preserving task allocation problem for multi-skill collaborative tasks in the Mobile CrowdSensing; finally, we present the design objectives of this paper.

2.1 System Model

The DPMTA scheme targets the problem of multi-skill collaborative task allocation in Mobile CrowdSensing, as illustrated in Fig. 1. The system model mainly consists of the Task Requester (TR), Workers (W), the Cloud Server Platform (CSP), a Trusted Authority (TA), and two fog servers (Fog Servers), denoted as FSA, FSB.

- Cloud Server Platform (CSP): The CSP acts as the scheduling and management entity of the system, and is responsible for task publication, information collection, candidate screening, and final task allocation decisions. Since the CSP occupies a central position in the workflow, direct access to sensitive information of workers or tasks may lead to privacy leakage risks. Therefore, in DPMTA, the information accessible to the CSP is strictly limited to a schedulable yet non-invertible scope. Specifically, the CSP only receives perturbed locations of tasks and workers, along with necessary auxiliary information, and uses them to perform candidate set filtering and approximate distance-based ranking. During the skill matching and collaborative selection stage, the CSP does not access any plaintext skill data or secret shares. Instead, it only obtains a small amount of statistical outputs from the two fog servers (e.g., the current coverage size, the coverage after adding a candidate worker, and the corresponding marginal gain), and performs greedy selection based on these statistics, thereby determining the collaborative worker set and completing task allocation for multi-skill collaborative tasks.
- Task Requester (TR): TR is the publisher of multi-skill collaborative sensing tasks, which can be an organization, enterprise, or individual user that requires sensing data from specific scenarios. When initiating a task, the TR specifies the target location and the set of required skills, which describe the capabilities needed to collaboratively complete the task, such as equipment inspection, event sampling, or multi-type data collection within a certain area. In DPMTA, to prevent the leakage of sensitive task information during transmission, the TR first perturbs the task location locally using differential privacy before uploading it to the CSP, where it is used for subsequent candidate screening and approximate distance estimation. Meanwhile, the TR splits the task-required skill vector into two random shares using XOR secret sharing and sends them separately to the two fog servers, FSA and FSB.
- Worker (W): Workers are the entities that actually perform sensing tasks. They typically carry mobile smart devices equipped with positioning and communication capabilities (e.g., smartphones, in-vehicle devices, or wearable devices) and possess one or more skills or capabilities required for task execution. In the system, workers need to report their location and skill information to participate in task matching and allocation. In DPMTA, each worker first perturbs their true location locally using differential privacy and then uploads the perturbed location to the CSP, so that the platform can only obtain approximate location information for candidate screening. Meanwhile, each worker splits their skill vector into two random shares using XOR secret sharing and sends them separately to the two fog servers, FSA and FSB.
- Fog Servers (FSA, FSB): The fog servers are key entities responsible for privacy-preserving computation in DPMTA. This paper adopts a dual-fog collaborative computation architecture, in which the task-required skill vector and worker skill vectors are stored in the form of XOR secret shares on FSA and FSB, respectively. Without reconstructing the plaintext skills, the two fog servers jointly perform the necessary secure computations and provide the cloud platform with the statistical results required for decision-making.
- Trusted Authority (TA): TA is responsible for system initialization, entity registration, and key material distribution. Specifically, the TA generates and distributes long-term public/private key pairs for the cloud platform and the two fog servers, which are used for key encapsulation in subsequent hybrid encryption communications. In addition, during the system initialization phase, the TA generates the Beaver triples required for secure computation offline and distributes them to the two fog servers. After completing the initialization process, the TA goes offline and does not participate in the subsequent online task allocation and scheduling process.

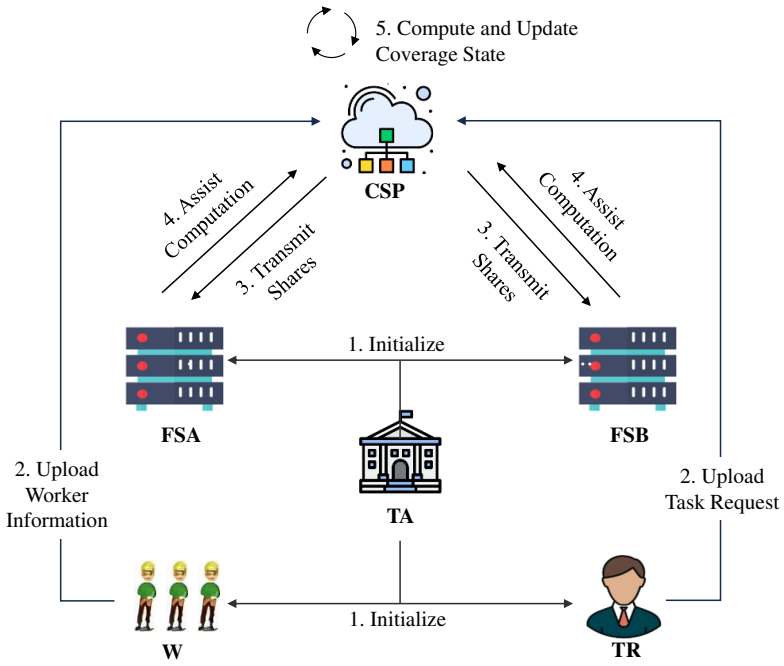


Figure 1: System model.

2.2 Threat Model

In our threat model, TA is assumed to be fully trusted and is responsible for system initialization and key distribution. TR and Workers are assumed to be honest and will not tamper with task or worker information. The CSP and the two fog servers FSA and FSB are considered semi-honest participants, meaning that they follow the prescribed protocols but may attempt to infer sensitive information from the data they observe. Specifically, the CSP may try to infer private information of tasks or workers based on perturbed location data and other observable outputs, while the fog servers may attempt to recover plaintext skill vectors by analyzing the intermediate results generated during the protocol execution. In addition, FSA and FSB are assumed to be non-colluding.

The above assumptions define the basic security boundary of DPMTA. The TA is assumed to be fully trusted because it is only responsible for system initialization, entity registration, key distribution, and offline Beaver triple generation. After the initialization phase, the TA goes offline and does not participate in the online task allocation process. TRs and workers are assumed to honestly generate and upload their perturbed locations and secret shares, since this paper focuses on protecting location privacy and skill privacy during task allocation, rather than verifying the correctness of submitted information. If malicious TRs or workers are considered, additional mechanisms such as authentication, commitment, verifiable secret sharing, or incentive-compatible verification should be incorporated. The non-collusion between FSA and FSB is a necessary assumption for XOR secret sharing. In practical deployment, the two fog servers can be managed by different administrative domains or independent edge service providers, which helps reduce the risk of collusion. If FSA and FSB collude, they can reconstruct plaintext skill vectors by combining their shares. If the CSP colludes with one fog server, plaintext skill vectors still cannot be directly recovered without the other fog server's shares, but the inference risk may increase.

2.3 Problem Definition and Design Goals

In this section, we present the problem definition and design objectives. To facilitate formal description and subsequent privacy-preserving computation, the skill set is encoded as a binary vector. Let the system define a universal skill set $S = \{1, 2, \ldots, m\}$, where m denotes the total number of skills. Table 1 summarizes the main notations used in the proposed scheme.

Table 1: Notations in the DPMTA scheme.

Symbol	Description
S	Universal skill set, $S = \{1, 2, \ldots, m\}$
m	Total number of skills
$W = \{w_i\}_{i=1}^{n}$	Worker set
n	Number of workers
$k = 1, \ldots, m$	Skill index
$\mathcal{T}$	Task request
tid	Task identifier
P	Random permutation rule
L_r, L_i	True locations of task and worker w_i
$\tilde{L}_r, \tilde{L}_i$	Perturbed locations of task and worker w_i
$r \in \{0,1\}^m$	Task-required skill vector
$s_i \in \{0,1\}^m$	Skill vector of worker w_i
ℓ	Iteration round
w_{i_ℓ}	Worker selected in the ℓ-th round
$s_{i_\ell} \in \{0,1\}^m$	Skill vector of selected worker in round ℓ
$t_\ell \in \{0,1\}^m$	Skill coverage vector after round ℓ
t_ℓ^P	Permuted coverage vector after applying rule P
$HW(\cdot)$	Hamming weight (number of 1s)
$h_\ell = HW(t_\ell)$	Number of covered skills after round ℓ
$R = HW(r)$	Total number of required skills
u, v, w	Beaver triples (single-bit), satisfying $w = u \wedge v$
$(K_A, k_A), (K_B, k_B)$	Public/private key pairs
r^A, r^B	XOR secret shares of task-required skill vector
s_i^A, s_i^B	XOR secret shares of worker skill vector
I_{TR}	Task request information
I_{w_i}	Worker report information
$E_{k_A}(\cdot), E_{k_B}(\cdot)$	Public-key encryption functions
Dis	Distance threshold
$\tilde{d}_i = dist(\tilde{L}_i, \tilde{L}_r)$	Distance between perturbed worker and task locations
$C \subseteq W$	Candidate worker set
$x_{i_\ell}[k]$	Contribution of worker w_{i_ℓ} on skill k
$y_{i_\ell}[k]$	Intermediate result in secure Boolean computation

Definition 1 (Task Request): *task request is represented as* $\mathcal{T} = (tid, L_r, r, R, Dis, P)$*, where* tid *denotes the task identifier,* L_r *is the true location of the task,* $r \in \{0,1\}^m$ *is the required skill vector, and* R *denotes the total number of required skills. If* $r[k] = 1$*, it indicates that the task requires skill* k*.* Dis *is the distance threshold for candidate filtering, and* P *denotes the random permutation rule.*

Definition 2 (Worker Information): *Let the set of workers be denoted as* $W = \{w_i\}_{i=1}^{n}$. *Each worker* w_i *is associated with a true location* L_i *and a skill vector* $s_i \in \{0,1\}^m$, *where* $s_i[k] = 1$ *indicates that worker* w_i *possesses skill* k.

Definition 3 (Skill Coverage): *The skill coverage state of a task is represented as an iteratively updated binary vector. Suppose that the worker selected in the* ℓ*-th round is* w_{i_ℓ}, *with skill vector* $s_{i_\ell} \in \{0,1\}^m$. *Let* $t_\ell \in \{0,1\}^m$ *denote the effective skill coverage vector after the* ℓ*-th round, with the initial state* $t_0 = \mathbf{0}$. *The update rule is defined as:*

$$t_\ell = t_{\ell-1} \vee (s_{i_\ell} \wedge r), \quad \ell = 1, 2, \ldots, |S|. \tag{1}$$

This recursion indicates that, after selecting a new worker, only the useful skill positions for the task (i.e., $s_{i_\ell} \wedge r$*) are incorporated into the current coverage state, thereby progressively accumulating the coverage of required skills. Let the coverage size be defined as* $h_\ell = HW(t_\ell)$, *and let* $R = HW(r)$ *denote the total number of required skills. The task is considered successfully covered if and only if* $h_\ell = R$, *in which case the selection process terminates. Otherwise, if* $h_\ell < R$ *after* ℓ *rounds, the task allocation fails.*

The specific design goals of this paper are concluded as follows:

- *Privacy Protection:* During the task allocation process, the required skill information and location data of both the task requester and workers are considered sensitive. The proposed scheme should complete task allocation while preventing the platform and any unauthorized entities from accessing the plaintext skills and location information of tasks and workers, thereby ensuring effective protection of both skill privacy and location privacy.
- *Functionality:* Without exposing the plaintext skill requirements or location data of tasks and workers, the system should be able to accurately determine whether a candidate worker set satisfies the task-required skills. Moreover, by incorporating the spatial relationships between tasks and workers, the scheme should achieve efficient multi-skill collaborative task allocation, ensuring both the correctness of allocation results and the overall usability of the system.

3 The Proposed Scheme

In this section, we present the detailed construction of the DPMTA scheme, which mainly consists of four phases: (1) initialization; (2) information uploading; (3) share distribution; (4) iterative update computation. Fig. 2 illustrates the sequence diagram of the proposed scheme.

3.1 Initialization

The system initialization is performed by the TA. In the offline phase, the TA generates a pool of Beaver triples and distributes them to the two fog servers. Specifically, the preprocessing module generates a large number of triples (u, v, w), where u and v are random bits and $w = u \wedge v$. These triples are then split into two shares using XOR secret sharing and distributed to FSA and FSB respectively. Each fog server locally maintains the triple pool and synchronization pointers. During the online phase, each secure bitwise AND operation consumes one triple to ensure that triples are not reused. In addition to generating Beaver triples, the TA also generates two pairs of asymmetric keys (K_A, k_A) and (K_B, k_B) during the initialization phase, which are distributed to FSA and FSB, respectively. Newly joined task requesters and workers are required to register with the TA. The public keys K_A and K_B are made available to all legitimate entities. After distributing the keys and preprocessing materials, the TA enters an offline state and does not participate in the subsequent online task allocation process.

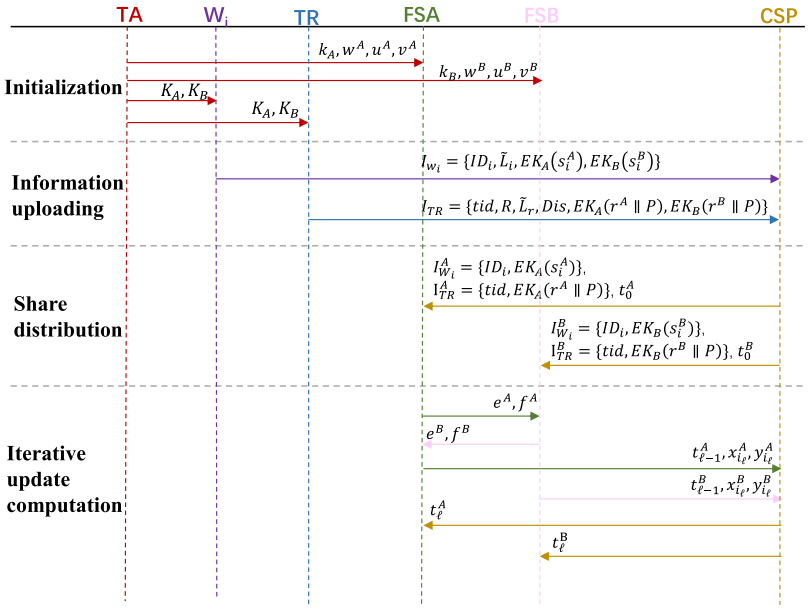


Figure 2: Sequence diagram of the DPMTA.

3.2 Information Uploading

3.2.1 Location Perturbation

The true location information of both task requesters and workers is highly sensitive. If directly uploaded to the cloud platform, it may enable the platform to infer participants' activity regions, spatial distribution patterns, and even mobility behaviors, thereby leading to potential location privacy leakage. To mitigate this risk, before uploading task and worker information, the proposed scheme perturbs the original two-dimensional location coordinates and only reports the perturbed locations to the cloud platform.

Considering that location data exhibit continuous spatial properties, this work adopts the planar Laplace mechanism for location perturbation by Wang et al. [19]. Specifically, random noise is added to the true location in a two-dimensional plane such that the perturbed locations are distributed around the original position. Locations closer to the true position have a higher probability of being reported, while those farther away have a lower probability, thereby achieving a balance between privacy protection and data utility.

Formally, let the true location of a task requester or a worker be denoted as $L = (L_x, L_y)$. Given a privacy budget ε, a random perturbation distance ρ and direction θ are first generated in polar coordinates, where θ is uniformly distributed over $[0, 2\pi]$, and ρ is sampled according to the probability distribution defined by the planar Laplace mechanism. The noise in polar coordinates is then converted into Cartesian offsets $(\Delta x, \Delta y) = (\rho \cos\theta, \rho \sin\theta)$, which are added to the true location, yielding the perturbed location: $\tilde{L} = (L_x + \rho \cos\theta,\ L_y + \rho \sin\theta)$.

After perturbation, only the obfuscated location $\tilde{L}$ is uploaded to the cloud platform, while the true location L remains private. The cloud platform then performs approximate distance computation and candidate worker selection based on the perturbed locations, thereby reducing the risk of location privacy leakage while preserving the feasibility of subsequent task allocation.

3.2.2 Task Request Uploading

TR initiates a task with identifier tid, and generates the true task location L_r and the required skill vector $r \in \{0,1\}^m$. To protect location privacy, TR locally perturbs L_r using differential privacy to obtain $\tilde{L}_r$. To protect skill privacy, TR performs XOR-based secret sharing on the required skill vector as follows: it randomly generates $r^A \in \{0,1\}^m$ and computes $r^B = r \oplus r^A$. In addition, TR needs to randomly generate a permutation rule denoted by P, which is used to shuffle the bit positions of the skill encoding vector. Specifically, P is defined as a sequence $(P_1, P_2, \ldots, P_m)$, which is a random permutation of $(1, 2, \ldots, m)$. Since it is necessary to protect the information from irrelevant entities, this scheme adopts an asymmetric encryption method. Specifically, TR constructs the task request as:

$$I_{TR} = \{tid, R, \tilde{L}_r, Dis, E_{K_A}(r^A \parallel P), E_{K_B}(r^B \parallel P)\} \tag{2}$$

and sends it to the cloud service platform (CSP).

3.2.3 Worker Information Uploading

Each worker w_i possesses a true location L_i and a skill vector $s_i \in \{0,1\}^m$. The worker first perturbs its location locally using differential privacy to obtain $\tilde{L}_i$. To protect skill privacy, the worker performs XOR-based secret sharing on the skill vector as follows: it randomly generates $s_i^A \in \{0,1\}^m$ and computes $s_i^B = s_i \oplus s_i^A$. Similarly, worker w_i constructs its information as:

$$I_{w_i} = \{ID_i, \tilde{L}_i, E_{K_A}(s_i^A), E_{K_B}(s_i^B)\} \tag{3}$$

and sends it to the cloud service platform (CSP).

3.3 Share Distribution

The CSP collects the perturbed task location $\tilde{L}_r$ and the perturbed worker locations $\tilde{L}_i$, and computes the perturbed distance $\tilde{d}_i = dist(\tilde{L}_i, \tilde{L}_r)$. Based on the distance threshold Dis, the CSP selects a candidate worker set $C \subseteq W$ according to proximity. In addition, the CSP computes $t_0^B = t_0 \oplus t_0^A$. Subsequently, the CSP assists in distributing the shares to FSA and FSB. Specifically, the CSP sends $I_{TR}^A = \{tid, E_{K_A}(r^A \parallel P)\}$, $I_{w_i}^A = \{ID_i, E_{K_A}(s_i^A)\}$, and t_0^A to FSA; and sends $I_{TR}^B = \{tid, E_{K_B}(r^B \parallel P)\}$, $I_{w_i}^B = \{ID_i, E_{K_B}(s_i^B)\}$, and t_0^B to FSB.

Furthermore, the CSP also sends the candidate worker ID list to both fog servers in order.

3.4 Secure Multi-Party Computation Protocol for Skill Coverage

This section presents the core protocol of DPMTA for computing the effective skill contribution and updating the coverage state without revealing the plaintext of the task-required and worker skill vectors. All vector Boolean operations are performed in a bitwise manner, i.e., each skill position $k = 1, \ldots, m$ is processed individually and the results are concatenated to form the final vector. According to the skill coverage definition in Section 2.3, the system maintains a coverage state vector $t_\ell \in \{0,1\}^m$ at round ℓ, which is updated iteratively. Based on Eq. (1), the update involves both bitwise AND and OR operations. In the following, we present secure protocols for these operations.

3.4.1 Secure Bitwise AND Computation

Let the effective matching bit between worker and task at skill position k be defined as:

$$x_{i_\ell}[k] = s_{i_\ell}[k] \wedge r[k], \tag{4}$$

and we expect the two fog servers to output its XOR shares $[x_{i_\ell}[k]] = (x_{i_\ell}^A[k], x_{i_\ell}^B[k])$, such that:

$$x_{i_\ell}[k] = s_{i_\ell}[k] \wedge r[k] = x_{i_\ell}^A[k] \oplus x_{i_\ell}^B[k], \tag{5}$$

while neither party can obtain the plaintext values of $s_{i_\ell}[k]$ or $r[k]$.

To realize Eq. (5), the DPMTA scheme relies on Beaver triples generated in the offline phase. For each computation (i.e., each invocation on a skill position k), the two fog servers fetch a triple $(u[k], v[k], w[k])$, where $u[k], v[k] \in \{0,1\}$ and satisfy: $w[k] = u[k] \wedge v[k]$.

The triple is shared between the two fog servers using XOR secret sharing, satisfying:

$$u[k] = u^A[k] \oplus u^B[k], \tag{6}$$
$$v[k] = v^A[k] \oplus v^B[k], \tag{7}$$
$$w[k] = w^A[k] \oplus w^B[k]. \tag{8}$$

During the online computation phase, the two fog servers first mask the input bits. Specifically, let:

$$e[k] = s_{i_\ell}[k] \oplus u[k], \quad f[k] = r[k] \oplus v[k]. \tag{9}$$

Since each fog server only holds shares of the inputs and the Beaver triple, they locally compute:

$$e^A[k] = s_{i_\ell}^A[k] \oplus u^A[k], f^A[k] = r^A[k] \oplus v^A[k], \tag{10}$$
$$e^B[k] = s_{i_\ell}^B[k] \oplus u^B[k], f^B[k] = r^B[k] \oplus v^B[k]. \tag{11}$$

Then, the two fog servers exchange their shares to reconstruct:

$$e[k] = e^A[k] \oplus e^B[k], \quad f[k] = f^A[k] \oplus f^B[k]. \tag{12}$$

It should be noted that $e[k]$ and $f[k]$ are only reconstructed between FSA and FSB and will not be sent to the cloud service platform (CSP). Since $u[k]$ and $v[k]$ are uniformly random bits, $e[k] = s_{i_\ell}[k] \oplus u[k]$ and $f[k] = r[k] \oplus v[k]$ are statistically indistinguishable from random bits. Therefore, even if the fog servers obtain $e[k]$ and $f[k]$, they still cannot infer the true values of $s_{i_\ell}[k]$ and $r[k]$.

After obtaining $e[k]$ and $f[k]$, the fog servers use the Beaver triple shares to construct the output shares. The proposed scheme adopts a common construction by assigning the public term $e[k] \wedge f[k]$ to the FSA side. Thus, the two fog servers output:

$$x_{i_\ell}^A[k] = w^A[k] \oplus (e[k] \wedge v^A[k]) \oplus (f[k] \wedge u^A[k]) \oplus (e[k] \wedge f[k]), \tag{13}$$
$$x_{i_\ell}^B[k] = w^B[k] \oplus (e[k] \wedge v^B[k]) \oplus (f[k] \wedge u^B[k]). \tag{14}$$

Since $e[k]$ and $f[k]$ are plaintext bits, operations such as $e[k] \wedge v^A[k]$ and $f[k] \wedge u^A[k]$ are bitwise ANDs between a plaintext bit and a share bit, and can therefore be computed locally by each fog server without additional interaction. Since the final result is represented in the form of XOR secret sharing, it remains to verify that the two output shares can be correctly combined to recover the target bitwise AND value, i.e., $x_{i_\ell}^A[k] \oplus x_{i_\ell}^B[k] = x_{i_\ell}[k] = s_{i_\ell}[k] \wedge r[k]$.

The correctness is shown as follows. XORing the two output shares yields:

$$\begin{aligned} x_{i_\ell}^A[k] \oplus x_{i_\ell}^B[k] = {} & (w^A[k] \oplus w^B[k]) \\ & \oplus \left(e[k] \wedge (v^A[k] \oplus v^B[k])\right) \\ & \oplus \left(f[k] \wedge (u^A[k] \oplus u^B[k])\right) \\ & \oplus (e[k] \wedge f[k]). \end{aligned} \tag{15}$$

Since $w[k] = u[k] \wedge v[k]$, and by using the Boolean identity:

$$(x \oplus y) \wedge (z \oplus t) = x \wedge z \oplus x \wedge t \oplus y \wedge z \oplus y \wedge t, \tag{16}$$

we can derive:

$$\begin{aligned} & w[k] \oplus (e[k] \wedge v[k]) \oplus (f[k] \wedge u[k]) \oplus (e[k] \wedge f[k]) \\ & \quad = (e[k] \oplus u[k]) \wedge (f[k] \oplus v[k]). \end{aligned} \tag{17}$$

Combining $e[k] = s_{i_\ell}[k] \oplus u[k]$ and $f[k] = r[k] \oplus v[k]$, we have:

$$e[k] \oplus u[k] = s_{i_\ell}[k], \quad f[k] \oplus v[k] = r[k]. \tag{18}$$

Therefore:

$$x_{i_\ell}[k] = x_{i_\ell}^A[k] \oplus x_{i_\ell}^B[k] = s_{i_\ell}[k] \wedge r[k]. \tag{19}$$

3.4.2 Secure Bitwise OR Computation

Since the bitwise OR operation satisfies the following identity:

$$a \vee b = a \oplus b \oplus (a \wedge b) \tag{20}$$

Eq. (1) can be transformed into:

$$t_\ell = t_{\ell-1} \vee (s_{i_\ell} \wedge r) = t_{\ell-1} \oplus (s_{i_\ell} \wedge r) \oplus \left[t_{\ell-1} \wedge (s_{i_\ell} \wedge r)\right], \tag{21}$$

By Eqs. (19), (21) can be further rewritten as:

$$t_\ell = t_{\ell-1} \oplus x_{i_\ell} \oplus (t_{\ell-1} \wedge x_{i_\ell}), \tag{22}$$

In Eq. (22), the term $t_{\ell-1} \wedge x_{i_\ell}$ can be computed again by applying the secure bitwise AND protocol, yielding:

$$y_{i_\ell}[k] = y_{i_\ell}^A[k] \oplus y_{i_\ell}^B[k] = t_{\ell-1}[k] \wedge x_{i_\ell}[k]. \tag{23}$$

3.4.3 Iterative Computation

After the ℓ-th round of computation, FSA holds $t_{\ell-1}^A$, $x_{i_\ell}^A$, and $y_{i_\ell}^A$, while FSB holds $t_{\ell-1}^B$, $x_{i_\ell}^B$, and $y_{i_\ell}^B$. The two fog servers upload their respective shares to the CSP. The CSP then performs bitwise XOR on these shares to obtain the effective coverage state vector t_ℓ in the ℓ-th round, as follows:

$$
\begin{aligned}
t_\ell &= t_{\ell-1}^A \oplus x_{i_\ell}^A \oplus y_{i_\ell}^A \oplus t_{\ell-1}^B \oplus x_{i_\ell}^B \oplus y_{i_\ell}^B \\
&= t_{\ell-1}^A \oplus t_{\ell-1}^B \oplus x_{i_\ell}^A \oplus x_{i_\ell}^B \oplus y_{i_\ell}^A \oplus y_{i_\ell}^B \\
&= t_{\ell-1} \oplus x_{i_\ell} \oplus y_{i_\ell} \\
&= t_{\ell-1} \oplus (s_{i_\ell} \wedge r) \oplus \left[t_{\ell-1} \wedge (s_{i_\ell} \wedge r) \right].
\end{aligned} \tag{24}
$$

It should be noted that, if the two fog servers directly send their shares to the CSP, the CSP may further infer which specific skill positions have been newly covered, thereby causing leakage of skill structure information. To mitigate this risk, before uploading, the two fog servers apply the random permutation rule P generated by the task requester to consistently permute their vector shares (i.e., perform the same bitwise reordering on the shares), thereby obtaining the permuted effective coverage state vector t_ℓ^P. Since the permutation does not change the number of 1s in the vector, it follows that $HW(t_\ell^P) = HW(t_\ell)$. Therefore, at the end of each round, the CSP only determines the stopping condition based on the quantity h_ℓ: if $h_\ell \geq R$, where $R = HW(r)$ denotes the total number of required skill positions, the task-required skills are considered fully covered and the iteration terminates; otherwise, the protocol proceeds to the next round to continue evaluating and selecting a new candidate worker.

3.5 Worker Selection Rule

After candidate filtering and secure skill matching, DPMTA performs greedy worker selection over the candidate worker set C. Let $\mathcal{A}$ denote the selected worker set, where $\mathcal{A} = \varnothing$ initially. At the beginning of the ℓ-th selection round, the current skill coverage state is denoted as $t_{\ell-1}$. For each unselected candidate worker $w_i \in C \backslash \mathcal{A}$, the two fog servers securely compute the effective skill coverage vector $x_i = s_i \wedge r$, where s_i denotes the skill vector of worker w_i, and r denotes the task-required skill vector. Then, the marginal skill coverage gain of worker w_i is defined as:

$$
\Delta_i = HW(t_{\ell-1} \vee x_i) - HW(t_{\ell-1}). \tag{25}
$$

Here, $HW(\cdot)$ denotes the Hamming weight, i.e., the number of covered skills. Therefore, Δ_i indicates the number of newly covered required skills if worker w_i is selected.

In each round, the CSP selects the worker w^* with the largest marginal gain according to the statistical coverage results returned by the fog servers. If multiple workers have the same marginal gain, the worker with the smaller perturbed distance $\widetilde{d}_i = dist(\widetilde{L}_i, \widetilde{L}_r)$ is selected.

After selecting w^*, the selected worker set is updated as: $\mathcal{A} \leftarrow \mathcal{A} \cup \{w^*\}$.

The task allocation process terminates successfully when all required skills are covered. If the remaining candidate workers cannot provide any additional skill coverage, the task allocation fails. The complete worker selection procedure is shown in Algorithm 1.

Algorithm 1: Worker selection in DPMTA

1: **Input:** Candidate worker set C, task-required skill vector r, number of required skills R, perturbed distances $\{\widetilde{d}_i\}_{w_i \in C}$
2: **Output:** Selected worker set $\mathcal{A}$ or failure
3: Initialize $\mathcal{A} \leftarrow \varnothing$, $t_0 \leftarrow \mathbf{0}$, and $\ell \leftarrow 1$
4: **while** $HW(t_{\ell-1}) < R$ and $|\mathcal{A}| < \min(R, |C|)$ **do**
5: **for** each $w_i \in C \backslash \mathcal{A}$ **do**
6: FSA and FSB securely compute the effective skill coverage vector $x_i = s_i \wedge r$

(Continued)

Algorithm 1 (continued)

FSA and FSB securely compute the updated coverage state $t_{\ell-1} \vee x_i$
The CSP obtains the corresponding coverage size and computes $\Delta_i = HW(t_{\ell-1} \vee x_i) - HW(t_{\ell-1})$
end for
if $\max_{w_i \in C \setminus \mathcal{A}} \Delta_i = 0$ **then**
return failure
end if
Let $\Delta_{\max} = \max_{w_i \in C \setminus \mathcal{A}} \Delta_i$
Let $\mathcal{B} = \{w_i \in C \setminus \mathcal{A} \mid \Delta_i = \Delta_{\max}\}$
Select $w^* = \arg\min_{w_i \in \mathcal{B}} \widetilde{d}_i$
$\mathcal{A} \leftarrow \mathcal{A} \cup \{w^*\}$
FSA and FSB securely update the skill coverage state $t_\ell \leftarrow t_{\ell-1} \vee x_{w^*}$
$\ell \leftarrow \ell + 1$
end while
if $HW(t_{\ell-1}) = R$ **then**
return $\mathcal{A}$
else
return failure
end if

The online computational complexity mainly comes from candidate filtering and iterative secure skill matching. Candidate filtering requires traversing all workers, and its complexity is $O(N)$, where N denotes the total number of workers. In each selection round, DPMTA evaluates the marginal coverage gain of unselected candidate workers over m-dimensional skill vectors. Therefore, the computational complexity of secure skill matching and coverage updating in each round is $O(|C|m)$.

Let ℓ denote the actual number of selection rounds. Then, the overall online computational complexity of DPMTA is $O(N + \ell|C|m)$.

4 Security Analysis

In this section, we provide a more formal security analysis of DPMTA under the semi-honest adversarial model. As defined in the threat model, the CSP, FSA, and FSB honestly follow the prescribed protocol, but may try to infer private information from their observed data. The two fog servers are assumed to be non-colluding. The security goal of DPMTA is to protect the plaintext task-required skill vector, worker skill vectors, and true locations during the task allocation process. Specifically, the CSP should not obtain plaintext skill information, and any single fog server should not reconstruct complete skill vectors from its own shares and intermediate values.

4.1 Leakage Definition and Adversarial Views

We first define the information visible to each potentially adversarial party. For the CSP, its real execution view is defined as

$$View_{CSP} = \{tid, R, \widetilde{L}_r, \widetilde{L}_i, Dis, C, ID_i, h_\ell, \Delta_i, \ell_{\text{stop}}, E_{K_A}(r^A \| P), E_{K_B}(r^B \| P), E_{K_A}(s_i^A), E_{K_B}(s_i^B)\}. \tag{26}$$

where tid is the task identifier, R is the number of required skills, $\widetilde{L}_r$ and $\widetilde{L}_i$ denote the perturbed task and worker locations, Dis is the distance threshold, C is the candidate worker set, ID_i is the worker identifier, h_ℓ is the coverage size in round ℓ, Δ_i is the marginal gain of candidate worker w_i, and ℓ_{stop} is the stopping round.

Moreover, $E_{K_A}(r^A \| P)$ and $E_{K_B}(r^B \| P)$ denote the ciphertexts of the task-required skill shares, while $E_{K_A}(s_i^A)$ and $E_{K_B}(s_i^B)$ denote the ciphertexts of the worker skill shares. The CSP can observe these ciphertexts during the upload and forwarding phases, but cannot decrypt them without the corresponding private keys.

Accordingly, the leakage function of the CSP is defined as

$$\mathcal{L}_{CSP} = \{tid, R, \widetilde{L}_r, \widetilde{L}_i, Dis, C, ID_i, h_\ell, \Delta_i, \ell_{\text{stop}}\}. \tag{27}$$

The above information is necessary for candidate filtering, distance-based ranking, and greedy worker selection. However, the leakage function does not include plaintext skill shares or plaintext skill vectors. Specifically, the CSP cannot access the true task location L_r, the true worker location L_i, the task-required skill vector r, the worker skill vector s_i, or their plaintext shares r^A, r^B, s_i^A, and s_i^B.

For a single fog server, taking FSA as an example, its real execution view is defined as

$$View_{FSA} = \{r^A, s_i^A, t_\ell^A, u^A, v^A, w^A, e, f, P\}, \tag{28}$$

where r^A, s_i^A, and t_ℓ^A are the task-required skill share, worker skill share, and coverage-state share held by FSA, respectively. u^A, v^A, and w^A are the Beaver triple shares, e and f are the opened masked values in the secure bitwise AND protocol, and P is the random permutation rule. Similarly, the view of FSB is

$$View_{FSB} = \{r^B, s_i^B, t_\ell^B, u^B, v^B, w^B, e, f, P\}. \tag{29}$$

The leakage function of a single fog server is defined as

$$\mathcal{L}_{FSA} = \{r^A, s_i^A, t_\ell^A, u^A, v^A, w^A, e, f, P\}, \tag{30}$$

and $\mathcal{L}_{FSB}$ is defined analogously. This leakage function indicates that a single fog server can only observe its own random shares and masked intermediate values, but cannot obtain the complete task-required skill vector r, worker skill vector s_i, or coverage state t_ℓ.

4.2 Security Against the CSP

For the CSP, we construct a simulator Sim_{CSP} that only takes $\mathcal{L}_{CSP}$ as input. The simulator outputs the task identifier, perturbed locations, distance threshold, candidate worker identifiers, coverage sizes, marginal gains, and stopping round according to $\mathcal{L}_{CSP}$. For the encrypted skill shares, the simulator generates random ciphertexts with the same length as the real ciphertexts.

In the real protocol, the CSP receives only encrypted skill shares rather than plaintext shares. Under the semantic security of the adopted public-key encryption scheme, the real ciphertexts are computationally indistinguishable from the simulated ciphertexts. Moreover, the CSP only receives the perturbed locations generated by the planar Laplace mechanism instead of the true locations. Therefore, the real view of the CSP and the simulated view are computationally indistinguishable: $\text{View}_{\text{CSP}}^{\text{real}} \approx_c \text{Sim}_{\text{CSP}}(\mathcal{L}_{\text{CSP}})$. Thus, the CSP cannot infer plaintext task-required skills, worker skills, or true locations beyond the defined leakage.

4.3 Security against a Single Fog Server

We take FSA as an example, and the analysis for FSB is the same. For FSA, we construct a simulator $\mathsf{Sim}_{\mathsf{FSA}}$. The simulator randomly samples $\hat{r}^A, \hat{s}_i^A, \hat{t}_\ell^A \leftarrow \{0,1\}^m$, and randomly generates Beaver triple shares $\hat{u}^A, \hat{v}^A, \hat{w}^A \leftarrow \{0,1\}$. It also samples masked values $\hat{e}$ and $\hat{f}$ from $\{0,1\}$ with the same distribution as those in the real protocol execution.

In XOR secret sharing, a single share is uniformly random and independent of the plaintext. For any bit $x[k] \in \{0,1\}$, the share $x^A[k]$ is randomly sampled from $\{0,1\}$, and the other share is computed as $x^B[k] = x[k] \oplus x^A[k]$. Therefore, observing only $x^A[k]$ reveals no information about $x[k]$. The same argument holds for r^A, s_i^A, and t_ℓ^A.

In the Beaver-triple-based secure bitwise AND protocol, the opened masked values are $e[k] = s_{i\ell}[k] \oplus u[k]$, $f[k] = r[k] \oplus v[k]$. Since $u[k]$ and $v[k]$ are uniformly random bits, $e[k]$ and $f[k]$ are also uniformly distributed and statistically independent of $s_{i\ell}[k]$ and $r[k]$. Thus, the masked values do not reveal the plaintext input bits.

Therefore, the simulated view and the real view have the same distribution: $\mathrm{View}_{\mathrm{FSA}}^{\mathrm{real}} \equiv \mathrm{Sim}_{\mathrm{FSA}}\left(\mathcal{L}_{\mathrm{FSA}}\right)$.

The same conclusion holds for FSB. Hence, under the non-colluding fog-server assumption, any single fog server cannot reconstruct plaintext task-required skills or worker skills.

4.4 Leakage of Random Permutation

The random permutation mechanism is used to hide the direct mapping between skill bit positions and original semantic skill labels. Since permutation preserves the Hamming weight, the CSP can still obtain the coverage size h_ℓ and marginal gain Δ_i required for greedy worker selection, but it cannot directly determine which original skill label corresponds to a specific covered position. However, random permutation does not hide all structural relationships between workers and skills. The CSP may still observe candidate worker identifiers, selected workers, coverage sizes, marginal gains, and stopping conditions. Across multiple tasks, such information may reveal statistical patterns such as worker similarity or capability distribution. Therefore, these observable values are explicitly included in the leakage function $\mathcal{L}_{CSP}$. Accordingly, DPMTA uses random permutation mainly to hide the semantic mapping between skill positions and skill labels, rather than to completely hide all skill-structure relationships.

4.5 Security Boundary

The security guarantees of DPMTA rely on the semi-honest model and the non-colluding assumption between the two fog servers. If FSA and FSB collude, they can combine their XOR shares to reconstruct the plaintext task-required skill vector and worker skill vectors. Therefore, DPMTA cannot resist skill privacy leakage caused by collusion between the two fog servers.

When the CSP colludes with one fog server, plaintext skills still cannot be directly recovered because the shares held by the other fog server are unavailable. However, combining the scheduling leakage observed by the CSP with the shares and intermediate values held by one fog server may increase inference risks. Therefore, DPMTA mainly guarantees skill privacy against the CSP alone and against any single non-colluding fog server.

Moreover, DPMTA is designed under the semi-honest model. The current scheme cannot detect malicious deviations from the protocol, such as reusing Beaver triples, sending incorrect shares, or returning incorrect computation results. Defending against malicious adversaries requires additional mechanisms, such as verifiable secret sharing, message authentication codes, or zero-knowledge proofs, which will be investigated in future work.

5 Performance Evaluation

In this section, we provide a comprehensive experimental evaluation of the proposed DPMTA scheme, systematically assessing its performance in three key factors: communication overhead, computational cost, and success rate of task allocation.

5.1 Experimental Setup

5.1.1 Experimental Environment

The proposed DPMTA scheme is implemented in Python and evaluated under the following experimental environment. The experiments are conducted on a Windows 11 operating system, with hardware including a 13th Gen Intel® Core™ i5-13500H processor, 16 GB RAM, and an NVIDIA GeForce RTX 4050 GPU.

5.1.2 Dataset

A real-world dataset, namely the NYC check-in dataset [20], is adopted in the experiments. This dataset contains approximately 227,000 real-world user check-in records, reflecting users' check-in behaviors in the New York City area. For ease of analysis, a 5 km × 5 km area around the city center is selected as the service region. Within this region, 40 locations are randomly selected as task locations, and 400 locations are chosen as worker locations. In addition, 40 skills are randomly generated to form the skill set. Asymmetric encryption is implemented using RSA-2048 to encrypt skill shares.

5.1.3 Baselines

To verify the effectiveness of DPMTA in the multi-skill collaborative task allocation scenario and further analyze the performance overhead introduced by privacy-preserving mechanisms, we select PUGR, TCRS, and Plain-Greedy as comparison baselines. The comparison metrics include task allocation success rate, computation overhead, and communication overhead. Among them, Plain-Greedy is used as a plaintext greedy allocation baseline without privacy protection, which provides a reference for evaluating the utility difference and additional system overhead introduced by location perturbation, secret sharing, and secure computation in DPMTA.

- PUGR [17]: It is a privacy-preserving multi-skill task allocation scheme. It protects location privacy through local differential privacy and improves task completion rate and allocation efficiency through greedy worker selection based on skill contribution values and pruning strategies. In this paper, PUGR is selected as one representative baseline of privacy-preserving task allocation methods.
- TCRS: It is a threshold-constrained random selection baseline designed in this paper. Under the distance threshold constraint, this method randomly selects R workers from the candidate worker set to execute the collaborative task, without introducing skill-contribution ranking or optimized screening mechanisms. It is used to evaluate the task completion performance and system overhead under a random allocation strategy.
- Plain-Greedy: It is a non-private plaintext greedy task allocation method. It directly uses real locations and plaintext skill information for candidate filtering and worker selection. Specifically, the platform first constructs the candidate worker set according to real distances, then computes the plaintext skill coverage of each candidate worker with respect to the task requirements, and selects the worker with the maximum newly added skill coverage in each round until all required skills are covered or no worker can provide positive marginal gain.

5.2 Experimental Results and Analysis

5.2.1 Candidate Set Difference Ratio

In the location perturbation mechanism, the privacy budget ε affects the magnitude of perturbation noise, which further influences the candidate worker set constructed according to perturbed distances. Since the secure skill matching and worker selection procedures in DPMTA are performed over the perturbed candidate set C, the quality of the candidate set affects the reliability of the task allocation result. When

the perturbed candidate set differs greatly from the ideal candidate set constructed from real distances, workers who truly satisfy the distance threshold may be mistakenly excluded, reducing the probability that all required skills can be fully covered. Meanwhile, workers who actually exceed the distance threshold may be mistakenly included, which may reduce the practical validity of the final allocation result under real distance constraints. In addition, changes in the candidate set size may also affect the number of secure skill matching operations, thereby indirectly influencing online computation and communication costs.

To evaluate the impact of location perturbation on candidate screening quality under different privacy budgets, we fix the number of workers to 400. Then, we vary the privacy budget ε and the distance threshold Dis, and calculate the average candidate set difference ratio $CSDR$ between the perturbed candidate set C and the ideal candidate set C_{true}.

Definition 4 (Ideal Candidate Set): *For experimental evaluation, the ideal candidate worker set constructed from real locations is defined as*

$$C_{\text{true}} = \{w_i \mid dist(L_i, L_r) \leq Dis\}, \tag{31}$$

where L_i and L_r denote the real locations of worker w_i and the task requester, respectively. It should be noted that C_{true} is only used as an offline reference in the experiment and is not available to the CSP during the online task allocation process.

Definition 5 (Candidate Set Difference Ratio): *To measure the deviation between the perturbed candidate set and the ideal candidate set, the candidate set difference ratio is defined as*

$$CSDR = \left(1 - \frac{|C \cap C_{\text{true}}|}{|C \cup C_{\text{true}}|}\right) \times 100\%. \tag{32}$$

A smaller CSDR indicates that the perturbed candidate set is closer to the ideal candidate set, while a larger CSDR indicates a larger deviation caused by location perturbation.

As shown in Fig. 3, under the same distance threshold, the candidate set difference ratio generally decreases as the privacy budget ε increases. When $\varepsilon = 0.1$, the perturbation noise is relatively large, and the deviation between the perturbed distance and the real distance becomes more obvious, resulting in a higher candidate set difference ratio. When ε increases to 0.7, the candidate set difference ratio is close to zero under several distance thresholds, indicating that the perturbed candidate set is almost consistent with the ideal candidate set and the candidate screening result becomes more reliable. Moreover, the candidate set difference ratio fluctuates under different distance thresholds. This is because the change in the candidate set is also affected by the spatial distribution of workers and the number of workers located near the threshold boundary.

Overall, a smaller ε provides stronger location privacy protection but introduces larger distance errors and reduces the reliability of candidate screening. A larger ε improves the quality of the candidate set and the practical effectiveness of the task allocation result, but it weakens location privacy protection. Therefore, this experiment demonstrates the trade-off between location privacy protection and task allocation utility. It should be noted that CSDR is used in this work as an intermediate indicator to evaluate the influence of the location privacy budget ε on candidate screening quality. Although a larger CSDR may indirectly affect the final task allocation success rate and online overhead by changing the candidate worker set, this experiment does not directly measure the final success rate or the end-to-end online computation and communication overhead under different ε values. The direct impact of ε on final allocation performance and end-to-end system overhead remains a limitation of the current evaluation and will be further investigated in future work.

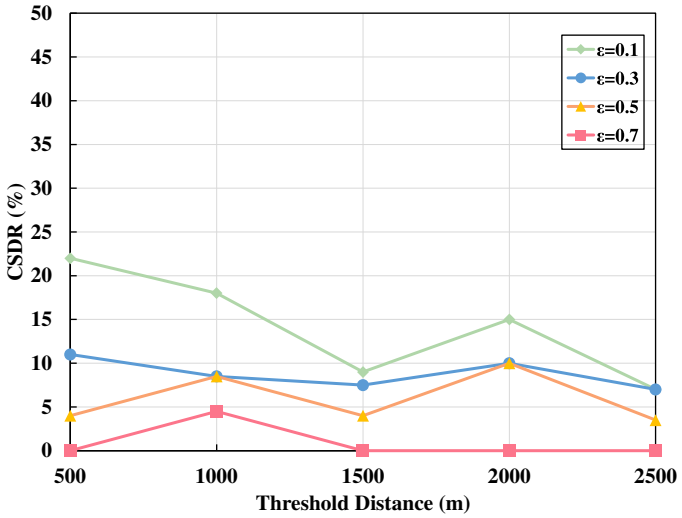


Figure 3: Candidate set difference ratio.

5.2.2 Communication Overhead

This paper evaluates the communication overhead of the DPMTA scheme from the perspective of the online process, i.e., the total amount of data exchanged among entities during one task allocation procedure. It should be noted that the generation and distribution of Beaver triples can be completed offline during the system initialization phase or idle periods, and thus do not contribute to the real-time communication burden in the online phase. To remain consistent with practical deployment scenarios, only the messages exchanged in the online phase are counted in the communication overhead statistics, while the offline preprocessing cost of Beaver triples is excluded.

- *Task publishing phase ($Comm_{init}$):* The task requester splits the task-required skill vector into two shares using XOR secret sharing and sends them to the two fog servers, respectively. Since RSA-2048 is adopted to directly encrypt the skill shares, the RSA ciphertext length is fixed at 2048 bits (256 B). Therefore, the communication overhead in the task publishing phase can be approximately regarded as a constant term:

$$\mathrm{Comm}_{init} \approx 2 \cdot |RSA| \text{ bits.} \tag{33}$$

- *Worker uploading phase ($Comm_{upload}$):* Each worker splits its skill vector into two shares using XOR secret sharing, encrypts them separately with the public keys, and uploads them. Then, the communication overhead of a single worker in the uploading phase is approximately:

$$\mathrm{Comm}_{upload}^{(i)} \approx 2 \cdot |RSA| \text{ bits.} \tag{34}$$

  Since the RSA ciphertext length is independent of the plaintext length, the above uploading overhead is mainly determined by the RSA key length.
- *Iterative selection phase ($Comm_{iter}$):* In this phase, the two fog servers perform secure bitwise AND computations over the m-dimensional skill shares of the candidate set C. For each round, they exchange the masked values e and f, incurring about $2|C|m$ bits of communication. After the computation, FSA and FSB send their shares of $t_{\ell-1}$, x_{i_ℓ}, and y_{i_ℓ} to the CSP, which reconstructs the coverage state and sends back the updated shares of t_ℓ. Therefore, each round additionally requires about $8m$ bits. The total communication overhead of the iterative selection phase is:

$$\text{Comm}_{iter} \approx \ell \cdot (2|C|m + 8m) \text{ bits.} \tag{35}$$

Fig. 4 shows the communication overhead of workers, tasks, and the server (cloud plus fog) under different skill dimensions m. As m increases from 10 to 40, the iterative selection phase overhead grows approximately linearly, while task publishing and single-worker uploading remain constant. This is because fog servers perform secure bitwise computation on m-dimensional skill shares and exchange masked intermediate values (e, f), then send/receive coverage shares with the cloud, making the overhead proportional to m. In contrast, the task publishing and worker upload phases use RSA-2048 encryption, whose fixed ciphertext length dominates their communication. Overall, online communication overhead grows controllably with m, mainly in the iterative selection phase, consistent with theoretical analysis.

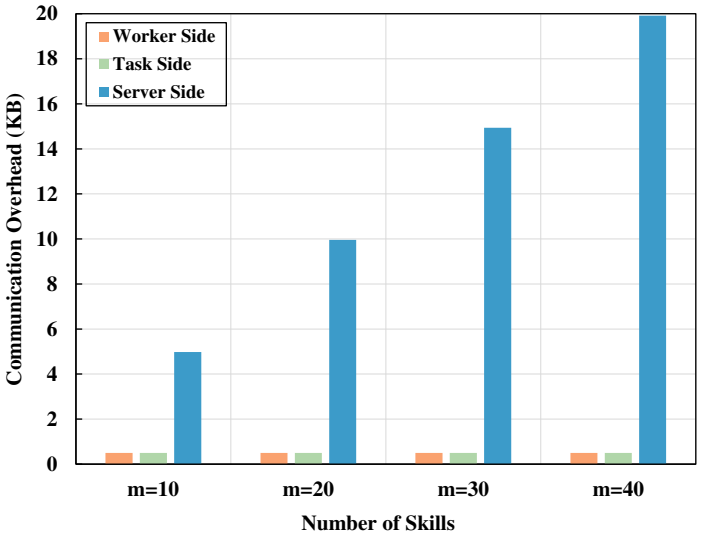


Figure 4: Communication overhead.

To further compare the online communication overhead of different task allocation schemes, we analyze Plain-Greedy, TCRS, PUGR, and DPMTA from the perspective of theoretical communication complexity. Let N denote the total number of workers, $|C|$ denote the size of the candidate worker set, m denote the skill dimension, and ℓ denote the number of greedy selection rounds in DPMTA. The main communication sources and communication overheads of different methods are summarized in Table 2.

Table 2: Communication overhead comparison of different schemes.

Method	Main communication source	Communication overhead
Plain-Greedy	Plaintext location and skill upload	$O(Nm)$
TCRS	Candidate information upload and random selection	$O(Nm)$
PUGR	Perturbed location and skill information upload	$O(Nm)$
DPMTA	Secret-share upload and secure computation interaction	$O(N + \ell\|C\|m)$

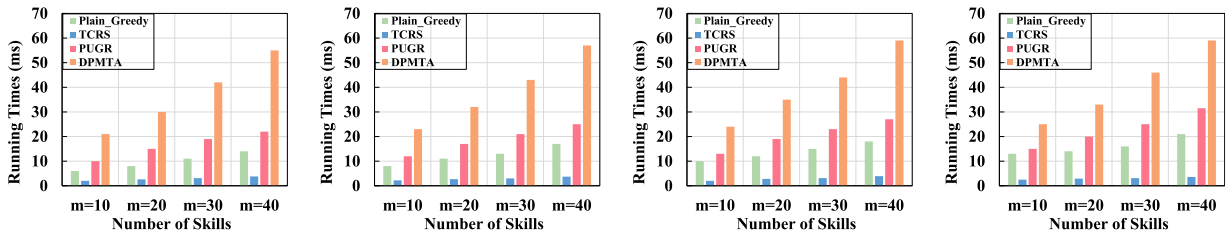
As shown in Table 2, Plain-Greedy, TCRS, and PUGR have relatively lower communication overhead because they do not involve multi-party secure computation. Plain-Greedy directly uploads plaintext locations and skill vectors, and then performs candidate filtering and greedy worker selection over plaintext data. TCRS randomly selects workers from the candidate set under the distance threshold constraint and

does not require secure skill matching. PUGR protects worker location privacy through local perturbation, and its subsequent worker selection is mainly performed by the platform according to skill contribution values and pruning strategies. Therefore, the communication overhead of these methods mainly comes from uploading worker information, such as locations and skill vectors.

In contrast, DPMTA introduces additional communication overhead to protect both location privacy and skill privacy. In the worker uploading phase, each worker uploads encrypted XOR secret shares of its skill vector to two fog servers. Since RSA ciphertexts have a fixed length, this phase contributes an $O(N)$ communication term. In the iterative selection phase, the two fog servers perform Beaver triple-based secure bitwise computations over the m-dimensional skill shares of candidate workers. This requires exchanging masked intermediate values and transmitting coverage-state shares between the fog servers and the CSP, resulting in the term $O(\ell|C|m)$. Therefore, the total online communication overhead of DPMTA can be summarized as $O(N + \ell|C|m)$. Although the communication overhead of DPMTA is higher than that of Plain-Greedy, TCRS, and PUGR, the additional communication cost is introduced for privacy protection. DPMTA protects location information through perturbation and protects skill information through XOR secret sharing and Beaver-triple-based secure computation. Therefore, DPMTA achieves stronger privacy protection at the cost of additional but controllable communication overhead.

5.2.3 Computational Overhead

Fig. 5 shows the online running time of different methods under different worker scales n and skill dimensions m, where the offline generation and distribution time of Beaver triples is not included. As shown in Fig. 5, the running time of all methods increases as the skill dimension m increases. This is because skill matching, skill contribution computation, and secure skill matching all need to process m-dimensional skill vectors. A larger skill dimension requires more computation in coverage evaluation and worker selection.



(a) $n = 100$ (b) $n = 200$ (c) $n = 300$ (d) $n = 400$

Figure 5: Computational overhead.

TCRS has the lowest running time because it only randomly selects workers from the candidate set and does not involve complex skill computation or secure computation. Plain-Greedy has a slightly higher running time than TCRS since it computes marginal skill coverage over plaintext data. PUGR further increases the running time because it needs to compute skill contribution values and perform pruning-based filtering. DPMTA executes secure skill matching and coverage-state updating based on XOR secret sharing and Beaver triples, which introduces additional secure bitwise computation overhead. Therefore, DPMTA has the highest running time.

Under the same skill dimension, increasing the worker scale n only leads to a limited increase in running time. This is because the candidate worker set is constrained by the distance threshold, and not all workers participate in the subsequent matching and selection process. Therefore, the running time is mainly affected by the skill dimension and the candidate set size. Overall, although DPMTA has higher computational

overhead than the other methods, it can simultaneously protect location privacy and skill privacy. Moreover, its online running time remains at the millisecond level, indicating that the additional overhead is acceptable.

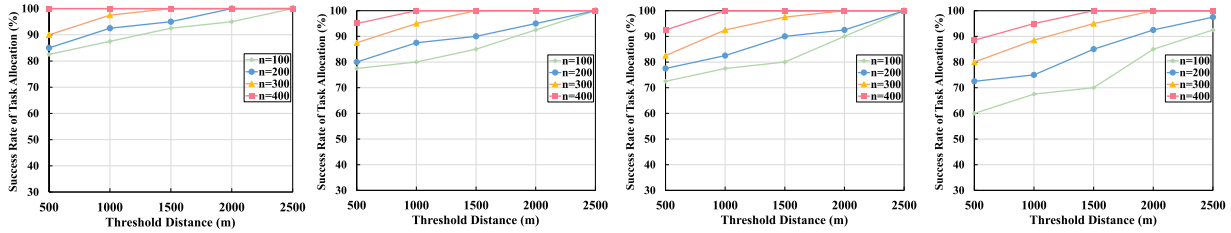
5.2.4 Success Rate of Task Allocation

This section evaluates the task allocation success rate of DPMTA under different skill distributions and task-complexity settings. The success rate measures whether the selected workers can collaboratively cover all required skills of a task. In the experiment, the threshold distance Dis is varied, the skill dimension is set to $m \in \{10, 20, 30, 40\}$, the worker scale is set to $n \in \{100, 200, 300, 400\}$, and 40 randomly generated tasks are used for evaluation.

To simulate skewed skill distributions in real MCS scenarios, we assign different occurrence probabilities p_k to different skills. For each skill dimension k, the skill value of worker w_i is generated as

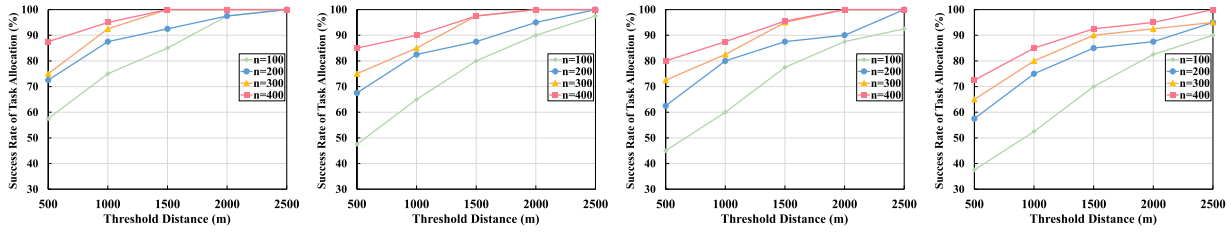
$$s_i[k] = \begin{cases} 1, & \text{with probability } p_k, \\ 0, & \text{with probability } 1 - p_k. \end{cases} \tag{36}$$

A smaller p_k indicates a rare skill, while a larger p_k indicates a common skill. In the experiment, skills are divided into rare, normal, and common categories, corresponding to $p_k \in [0, 0.3]$, $p_k \in (0.3, 0.7]$, and $p_k \in (0.7, 1]$, respectively. In addition, to evaluate the impact of task complexity, the number of required skills is set to $R = 2$, $R = 4$, and $R = 6$, where a larger R indicates a more complex task. Figs. 6–8 show the success rates under these three task-complexity settings.



(a) $m = 10$ (b) $m = 20$ (c) $m = 30$ (d) $m = 40$

Figure 6: Success Rate of Task Allocation under task complexity $R = 2$.

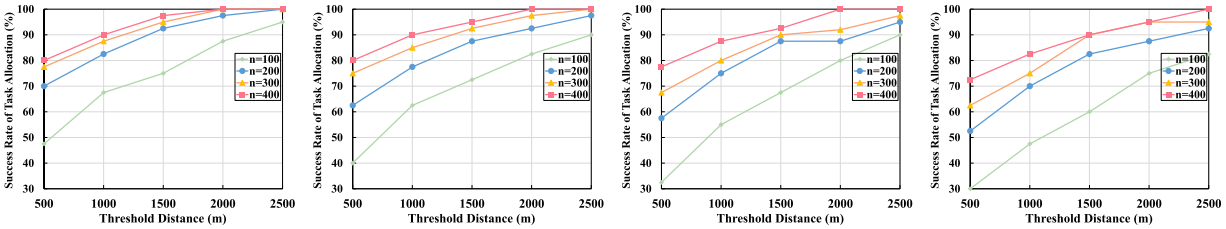


(a) $m = 10$ (b) $m = 20$ (c) $m = 30$ (d) $m = 40$

Figure 7: Success Rate of Task Allocation under task complexity $R = 4$.

As shown in Figs. 6–8, the task allocation success rate generally increases as the threshold distance Dis increases. This is because a larger distance threshold expands the candidate worker set, allowing more workers to participate in secure skill matching and collaborative selection, thereby increasing the probability of covering all required skills. When Dis is small, the number of candidate workers is limited. In particular,

under higher task complexity or larger skill dimensions, the candidate set may lack workers with some required skills, resulting in a lower success rate.



(a) $m = 10$ (b) $m = 20$ (c) $m = 30$ (d) $m = 40$

Figure 8: Success Rate of Task Allocation under task complexity $R = 6$.

The increase in worker scale n also improves the success rate. With more workers, more candidates are available for selection, and the skill diversity in the candidate set becomes richer. Therefore, it is easier to find a group of workers that can collaboratively cover the task requirements. This effect is more obvious when Dis is small, since increasing the worker scale can alleviate the shortage of candidate workers.

By comparing Figs. 6–8, we can observe that the task allocation becomes more difficult as the number of required skills R increases. Under the same Dis, m, and n, the success rate when $R = 6$ is generally lower than that when $R = 2$ or $R = 4$. This is because more complex tasks require more skills to be covered. When rare skills are involved, fewer candidate workers can satisfy the requirements, which further reduces the success rate.

Moreover, increasing the skill dimension m also raises the difficulty of task allocation. Under skewed skill distributions, a larger skill space may reduce the overlap between worker skills and task-required skills, making rare skills more likely to become the key factor affecting the success rate. Therefore, the success rate decreases more noticeably under larger m, smaller Dis, or smaller n.

Overall, the threshold distance, worker scale, skill dimension, and task complexity all affect the task allocation success rate. Larger Dis and n can improve the skill coverage capability of the candidate set and increase the success rate, while larger m and R increase the difficulty of skill coverage, especially when rare skills are involved. The experimental results show that DPMTA can maintain good task allocation performance under skewed skill distributions and different task-complexity settings.

Fig. 9 compares the task allocation success rates of Plain-Greedy, DPMTA, PUGR, and TCRS under 40 tasks, 400 workers, skill dimension $m = 20$, and distance thresholds $Dis \in \{500, 1000, 1500, 2000, 2500\}$.

As shown in Fig. 9, the task allocation success rates of Plain-Greedy, DPMTA, and PUGR generally increase as the distance threshold Dis increases. This is because a larger distance threshold expands the candidate worker set, providing more available workers and richer skill combinations. Therefore, the probability that all required skills can be collaboratively covered becomes higher. When Dis is small, the number of candidate workers is limited, and some required skills may not be sufficiently covered by the candidate workers, resulting in a relatively lower success rate.

Plain-Greedy achieves the highest or nearly the same success rate as DPMTA. This is because it directly uses real locations and plaintext skills without being affected by location perturbation or privacy-preserving computation. Therefore, it serves as a utility reference upper bound without privacy protection. The success rate of DPMTA is very close to that of Plain-Greedy, indicating that the introduced location perturbation, XOR secret sharing, and secure computation mechanisms do not significantly reduce the task allocation utility. DPMTA can maintain a high task completion capability while protecting privacy.

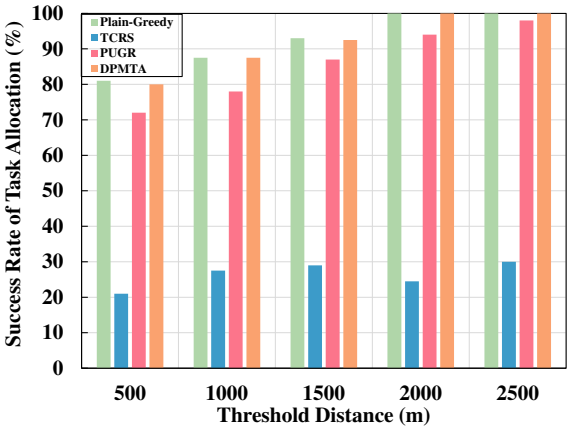


Figure 9: Success rates of different schemes.

Compared with Plain-Greedy and DPMTA, PUGR achieves a lower success rate, but it still significantly outperforms TCRS. PUGR improves worker selection through skill contribution values and pruning strategies, and its success rate also increases with the distance threshold. However, its selection process is affected by pruning strategies and local skill-contribution evaluation, which may exclude some workers that could be useful for subsequent skill complementarity. Therefore, its success rate is slightly lower than that of DPMTA, which focuses on collaborative skill coverage.

TCRS always has the lowest success rate because it randomly selects workers from the candidate set without considering their marginal contribution to the required skill coverage. As a result, it cannot reliably guarantee that the selected workers can collaboratively cover all required skills. Therefore, even when the distance threshold increases, the improvement in the success rate of TCRS remains limited.

Overall, the experimental results show that collaborative worker selection based on skill coverage gain can significantly improve the success rate of multi-skill task allocation. DPMTA achieves a success rate close to the plaintext greedy method Plain-Greedy while protecting both location privacy and skill privacy, demonstrating a good trade-off between privacy protection and task allocation utility.

6 Conclusion

In this paper, we proposed DPMTA, a dual-fog privacy-preserving task allocation scheme for multi-skill collaborative tasks in mobile crowdsensing. The proposed scheme protects task and worker locations using differential privacy, and employs XOR-based secret sharing with Beaver triple-assisted secure Boolean computation to enable skill matching, coverage updating, and collaborative worker selection without revealing plaintext skills. A random permutation mechanism is further introduced to hide the direct mapping between skill positions and semantic skill labels. Security analysis shows that, under the non-collusion assumption of the two fog servers, DPMTA can effectively protect task-required skills, worker skills, and location information. Experimental results demonstrate that DPMTA maintains a high task allocation success rate with acceptable communication and computation overhead, confirming its feasibility and effectiveness in multi-skill collaborative mobile crowdsensing scenarios. Future work will consider privacy-preserving task allocation under dynamic worker participation and more complex task constraints.

Acknowledgement: We gratefully acknowledge the use of the NYC dataset and the support from the funding sources listed in the Funding Statement.

Funding Statement: This work is supported by the National Natural Science Foundation of China (Nos. 62302230, U22B2062, U22A2030, U23A20303, 62302229, 62202051, 62372149), and the China Postdoctoral Science Foundation (No. 2024M751480).

Author Contributions: Study conception and design: Jie Li, Fuyuan Song; data collection: Jie Li; analysis and interpretation of results: Jie Li, Fuyuan Song, Qin Jiang; draft manuscript preparation: Jie Li; Review: Jie Li. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Guo B, Wang Z, Yu Z, Wang Y, Yen NY, Huang R, et al. Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm. ACM Comput Surv. 2015;48(1):1–31. doi:10.1145/2794400.
2. Hu S, Su L, Liu H, Wang H, Abdelzaher TF. Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. ACM Trans Sens Netw. 2015;11(4):1–27.
3. Cheng Y, Li X, Li Z, Jiang S, Li Y, Jia J, et al. AirCloud: A cloud-based air-quality monitoring system for everyone. In: Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems; 2014 Nov 3–6; Memphis, Tennessee. p. 251–65.
4. Wang Y, Li Y, Wang W, Duan P, Sai AMVV, Cai Z. Mobile crowdsourcing based on 5G and 6G: A survey. Neurocomputing. 2025;618:128993. doi:10.2139/ssrn.4757416.
5. Chen J, Yang J. Maximizing coverage quality with budget constrained in mobile crowd-sensing network for environmental monitoring applications. Sensors. 2019;19(10):2399. doi:10.3390/s19102399.
6. Song F, Ding S, Huang C, Jiang Q, Fu Z. A privacy-preserving and efficient spatial keyword based task matching scheme in crowdsourcing. In: Proceedings of the GLOBECOM 2025—2025 IEEE Global Communications Conference; 2025 Dec 8–12; Taipei, Taiwan. p. 2910–5.
7. Wang H, Yang Y, Wang E, Liu W, Xu Y, Wu J. Truthful user recruitment for cooperative crowdsensing task: A combinatorial multi-armed bandit approach. IEEE Trans Mob Comput. 2022;22(7):4314–31.
8. Wang J, Wang L, Wang Y, Zhang D, Kong L. Task allocation in mobile crowd sensing: state-of-the-art and future opportunities. IEEE Internet Things J. 2018;5(5):3747–57. doi:10.1109/jiot.2018.2864341.
9. Zhang C, Zhu L, Xu C, Ni J, Huang C, Shen X. Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing. IEEE Trans Mob Comput. 2021;21(12):4410–25. doi:10.1109/tmc.2021.3080714.
10. Ma Y, Gao X, Bhatti SS, Chen G. Clustering based priority queue algorithm for spatial task assignment in crowdsourcing. IEEE Trans Serv Comput. 2024;17(2):452–65. doi:10.1109/tsc.2024.3353293.
11. Song F, Liang J, Zhang C, Fu Z, Qin Z, Guo S. Achieving efficient and privacy-preserving location-based task recommendation in spatial crowdsourcing. IEEE Trans Dependable Secur Comput. 2023;21(4):4006–23. doi:10.1109/tdsc.2023.3342239.
12. Wang Y, Cai Z, Zhan ZH, Zhao B, Tong X, Qi L. Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing. IEEE Trans Comput Soc Syst. 2020;7(4):1033–46. doi:10.1109/tcss.2020.2995760.
13. Li X, Zhang L, Zhou M, Bian K. Task recommendation based on user preferences and user-task matching in mobile crowdsensing. Appl Intell. 2024;54(1):131–46. doi:10.1007/s10489-023-05208-w.
14. Chen Y, Guo D, Bhuiyan MZA, Xu M, Wang G, Lv P. Towards profit optimization during online participant selection in compressive mobile crowdsensing. ACM Trans Sens Netw. 2019;15(4):1–29. doi:10.1145/3342515.

15. Han L, Yu Z, Yu Z, Wang L, Yin H, Guo B. Online organizing large-scale heterogeneous tasks and multi-skilled participants in mobile crowdsensing. IEEE Trans Mob Comput. 2021;22(5):2892–909. doi:10.1109/tmc.2021.3132616.
16. Wei K, Qi G, Li Z, Guo S, Chen J. Group task recommendation in mobile crowdsensing: an attention-based neural collaborative approach. IEEE Trans Mob Comput. 2023;23(8):8066–76.
17. Fang D, Yao Y, Zhang P, Huang S. A task allocation algorithm for private skill sensing based on local differential privacy. Electron Inf Technol. 2025;47:1–11 (In Chinese).
18. Lian R, Zheng Y, Wang C. Privro: A privacy-preserving crowdsourcing service with robust quality awareness. IEEE Trans Serv Comput. 2024;17(4):1682–97.
19. Wang H, Wang E, Yang Y, Wu J, Dressler F. Privacy-preserving online task assignment in spatial crowdsourcing: A graph-based approach. In: Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications; 2022 May 2–5; Virtual. p. 570–9.
20. Yang D, Zhang D, Zheng VW, Yu Z. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. IEEE Trans Syst Man Cybern Syst. 2014;45(1):129–42. doi:10.1109/tsmc.2014.2327053.