Computer Networks 224 (2023) 109600

Contents lists available at ScienceDirect

Computer Networks

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

PPTA: A location privacy-preserving and flexible task assignment service for spatial crowdsourcing

Menglun Zhou [a], Yifeng Zheng [a,*], Songlei Wang [a], Zhongyun Hua [a], Hejiao Huang [a], Yansong Gao [b], Xiaohua Jia [a,c]

[a] *School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China*
[b] *School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China*
[c] *Department of Computer Science, City University of Hong Kong, Hong Kong, China*

ARTICLE INFO

Keywords:
Spatial crowdsourcing
Task assignment
Location privacy

ABSTRACT

With the rapid growth of sensor-rich mobile devices, spatial crowdsourcing (SC) has emerged as a new crowdsourcing paradigm harnessing the crowd to perform location-dependent tasks. To appropriately select workers that are near the tasks, SC systems need to perform location-based task assignment, which requires collecting worker locations and task locations. Such practice, however, may easily compromise the location privacy of workers. In light of this, in this paper, we design, implement, and evaluate PPTA, a new system framework for location privacy-preserving task assignment in SC with strong security guarantees. PPTA takes advantage of only lightweight cryptography (such as additive secret sharing, function secret sharing, and secure shuffle), and provides a suite of tailored secure components required by practical location-based task assignment processes. Specifically, aiming for practical usability, PPTA is designed to flexibly support two realistic task assignment settings: (i) the online setting where tasks arrive and get processed at the SC platform one by one, and (ii) the batch-based setting where tasks arrive and get processed in a batch. Extensive experiments over a real-world dataset demonstrate that while providing strong security guarantees, PPTA supports task assignment with efficacy comparable to plaintext baselines and with promising performance.

1. Introduction

The rapid growth of sensor-rich mobile devices enables individuals to easily collect various kinds of valuable information about their surroundings. Along with such growth is the emergence of a new crowdsourcing paradigm called spatial crowdsourcing (SC) [1,2], which harnesses the crowd to perform location-dependent tasks (e.g., taking photos at certain spots). In particular, workers in SC systems are required to move physically to designated locations to conduct the assigned tasks. Location-based task assignment is thus a fundamental demand in SC systems, where the SC platform needs to assign tasks to eligible workers based on their spatial proximity [3,4].

Such location-based task assignment requires collecting the locations of both tasks and workers. However, while being interested in contributing to the SC service, workers may not be willing to expose their private locations as this would violate their privacy [2]. Such privacy concerns, if not addressed appropriately, will seriously hinder the growth and deployment of SC systems in practice. It is noted that task locations, if not protected, can also be exploited to infer worker locations based on the task assignment result. Specifically, workers that get selected for a task are close to the location of that task (say within a certain distance). So knowing the task assignment result along with the task locations will allow an adversary to infer information about the workers' current locations. Therefore, there is an urgent demand that security must be embedded in such SC service paradigm from the very beginning, providing protection for worker locations as well as task locations.

In the literature, location privacy-preserving task assignment in SC has received increasing attentions in recent years [5–8]. Existing works can be generally divided into two categories based on the task assignment settings considered: the online setting and the batch-based setting. For the online setting, tasks arrive at the SC platform one by one. Once a task arrives, the SC platform assigns it to eligible workers. The goal in this setting is to minimize the average distance between a task and the workers assigned to this task [5]. In the batch-based setting, tasks are delivered to the SC platform batch by batch. Given a batch of tasks, the goal in this setting is to maximize the number of successfully matched worker-task pairs [7,8].

* Corresponding author.
E-mail addresses: menglun.zhou@outlook.com (M. Zhou), yifeng.zheng@hit.edu.cn (Y. Zheng), songlei.wang@outlook.com (S. Wang), huazhongyun@hit.edu.cn (Z. Hua), huanghejiao@hit.edu.cn (H. Huang), yansong.gao@njust.edu.cn (Y. Gao), csjia@cityu.edu.hk (X. Jia).

https://doi.org/10.1016/j.comnet.2023.109600
Received 26 September 2022; Received in revised form 2 January 2023; Accepted 27 January 2023
Available online 31 January 2023

Despite being valuable, existing works on location privacy-preserving task assignment are customized for either the online setting [5,6] or the batch-based setting [7,8]. Yet, in practice the locations of workers may be used in different assignment settings and the SC platform needs to flexibly switch among them. For example, in scenarios like ride hailing [9,10], a passenger publishes a single ride hailing task and wants the SC platform to provide instant task assignment services (i.e., the online task assignment setting). In other scenarios like environmental data collection [11,12], a task requester may want to monitor multiple areas for a research study, where the monitoring of each area corresponds to a task. So the task requester publishes the tasks in a batch and wants the SC platform to assign tasks in a batch to maximize the number of matched worker-task pairs. Therefore, it is practically important for a SC system to flexibly support both the online setting and the batch-based setting. Meanwhile, the SC platform should be able to flexibly switch between different task assignment settings. Consider the fact that workers who are not selected in the current round of task assignment will continue to participate in the next round. If the two rounds have different task assignment settings, the SC platform is then required to switch to another task assignment setting on the same workers left from the previous round. Therefore, a SC system with flexible support for different task assignment settings should allow the SC platform to directly do the switch.

The overarching challenge in supporting flexible location privacy-preserving task assignment is how to allow the workers to be agnostic to the specific task assignment settings when uploading their locations in protected form, while still enabling the SC platform to flexibly switch between different settings. In prior works, workers are required to protect their locations via a tailored approach to make the protected locations compatible with the designated setting of task assignment. However, in practical SC systems, workers normally have no knowledge in the subsequent setting of task assignment when uploading their locations to the SC platform. In addition, with prior works, if the SC platform wants to change the setting of task assignment on the same workers, they require the workers to re-upload their protected locations specific to the new setting. This will heavily hinder the secure task assignment process because the SC platform has to wait for all the workers to re-upload their protected locations. Hence, it remains to be explored how to allow one-off encryption of workers' locations and enable the SC platform to securely execute different task assignment strategies on demand.

In addition, it is important to offer strong security without sacrificing the utility of task assignment. Note that prior works supporting single setting are faced with notable information leakages (e.g., exposing the distances between tasks and locations [5,8]), or produce coarse-grained/inaccurate assignment results due to the use of approximated locations [7], or noise addition techniques based on differential privacy [6,8]. Hence, we need to design a framework for flexible location privacy-preserving task assignment from the ground up.

In light of the above, in this paper, we design, implement, and evaluate PPTA, the first system framework for privacy-preserving and flexible location-based task assignment in SC, which embraces both the online setting and batch-based setting with strong security guarantees. PPTA is built from a delicate synergy of insights on practical location-based task assignment strategies and advanced lightweight cryptography, allowing the SC platform to assign tasks to workers based on their encrypted locations. Specifically, we first deeply examine and adequately decompose the process of practical location-based task assignment into three phases: available workers labeling, k-nearest available workers search, and k-nearest available workers revealing. To meet the functionalities and security demands of these phases, we design a suite of tailored secure components based on lightweight cryptography (such as additive secret sharing, function secret sharing, and secure shuffle).

From a high-level point of view, PPTA provides two protocols for privacy-preserving location-based task assignment under the online setting and the batch-based setting, respectively. In both protocols, the locations of tasks and workers are protected with a lightweight cryptographic technique called additive secret sharing [13], which works by generating secret shares for the input data. For compatibility with the working paradigm of additive secret sharing, the SC platform in PPTA is jointly empowered by two SC service providers. Such two-server model has seen increasing adoption in academia [14–16] as well as in industry [17]. The adoption of such model in PPTA follows this trend. Upon receiving the secret shares of task locations and worker locations, the SC service providers perform task assignment according to customized secure protocols developed in PPTA. For both settings, PPTA ensures that the SC service providers can effectively perform the task assignment without knowing locations of tasks and workers, and only learn which workers are assigned to which tasks. Also, in contrast with prior works, PPTA provides strong security guarantees by keeping confidential the distances between tasks and workers, as well as hides the relative proximity of each matched worker to a task, i.e., the order information regarding which matched worker is closer to a task. We highlight our contributions below:

- We propose PPTA, a new system framework supporting location privacy-preserving and flexible task assignment in SC, simultaneously embracing both the online setting and batch-based setting with strong security guarantees.
- We adequately decompose the process of secure location-based task assignment and devise a suite of corresponding secure components for both settings, including secure available workers labeling, secure k-nearest available workers search, and secure k-nearest available workers revealing.
- We formally analyze the security of PPTA, and conduct extensive experiments on a real-world dataset. The results demonstrate that while providing security guarantees, PPTA can support task assignment with efficacy comparable to as plaintext baselines and with promising performance.

The rest of this paper is organized as follows. We first review the related work in Section 2 and introduce preliminaries in Section 3. Then we present the problem statement in Section 4. We elaborate on the designs of PPTA in Sections 5 and 6, where Section 5 presents the secure task assignment protocol under the online setting and Section 6 presents the secure task assignment protocol under the batch-based setting. The security analysis is presented in Section 7, followed by the experiments in Section 8. Finally, we conclude this paper in Section 9.

2. Related work

In recent years, there have been a variety of schemes for privacy-preserving location-based task assignment in SC [4–8,18,19]. According to the techniques used for location protection during task assignment, these schemes can be divided into three categories: (i) cloaking-based [4,19], (ii) perturbation-based [6,18], and (iii) encryption-based [5,7]. Cloaking-based schemes [4,19] protect the locations of tasks and workers by obfuscating a location into a cloaked region. Perturbation-based schemes [6,18] distort locations by adding artificial noises generated based on geo-indistinguishability [20], which is an extension of the conventional differential privacy notion [21]. Encryption-based schemes usually encrypt the locations of workers and tasks into ciphertexts through custom encryption schemes [5,7]. However, we note that all of them are customized for location privacy-preserving task assignment in either the online setting or the batch-based setting, lacking flexibility for meeting practical demands.

Furthermore, prior works are also confronted with issues of inaccurate task assignment and/or notable information leakages. In particular, the cloaking-based and perturbation-based schemes [4,6,18,19] have the shortcomings of inaccurate task assignment. The (state-of-the-art) encryption-based schemes [5,7] are faced with notable leakages and/or inaccurate task assignment. Specifically, the work [7] leaks the relative

Table 1
Comparison of PPTA with Existing Works on Privacy-Preserving Location-Based Task Assignment.

Property	[5]	[7]	[8]	[6]	PPTA
Supporting online setting	✓	×	×	✓	✓
Supporting batch-based setting	×	✓	✓	×	✓
Distance protection	×	✓	×	✓	✓
Relative proximity protection	×	×	×	✓	✓
Accurate assignment	✓	×	×	×	✓
Offline requesters/workers	×	✓	×	✓	✓
Workers agnostic to settings	×	×	×	×	✓

proximity between workers and tasks to the SC platform, which has been shown to be exploitable for various attacks [22,23]. In addition, the work [7] cannot support fine-grained and accurate task assignment since it uses a grid-based encryption scheme to encrypt approximate locations of workers and tasks and then assigns tasks based on the grids of tasks and workers. The work [5] has weaker security guarantees since it directly exposes the distances between tasks and workers. Besides, the task requester and workers in its design has to stay online during the task assignment since they have to undertake some processing workload.

In additional recent work [8], Li et al. propose a scheme that combines the strategies of perturbation and encryption. However, it has limited security guarantees and practicability like [5] since it also directly exposes the task-worker distances and requires the task requester and workers to stay online during the task assignment process.

Different from existing works, PPTA can flexibly support location privacy-preserving task assignment for both the online setting and the batch-based setting. The prominent benefit is that the workers are allowed to be agnostic to the subsequent task assignment settings when uploading their encrypted locations and the SC platform can flexibly switch between different settings. In prior works workers have to protect their locations via approaches tailored for the subsequent task assignment setting. Meanwhile, PPTA ensures stronger security guarantees by keeping confidential the distances between tasks and workers throughout the whole secure task assignment process, as well as hides the relative proximity of each matched worker to a task in the assignment result. Furthermore, PPTA allows the process of secure task assignment to be fully processed at the SC platform. Table 1 summarizes the prominent advantages of our PPTA over existing works on privacy-preserving location-based task assignment.

3. Preliminaries

3.1. Additive secret sharing

Additive secret sharing (ASS) [13] is a cryptographic primitive which provides protection for a private value x in two-party setting by splitting it into two secret shares, where each of the share alone reveals nothing about x. For convenience, we represent the ASS of x by $[\![x]\!]$. There are two types of ASS: (1) *Arithmetic sharing*: $x = [\![x]\!]_0^A + [\![x]\!]_1^A$, where $x, [\![x]\!]_0^A, [\![x]\!]_1^A \in \mathbb{Z}_{2^l}$, $l > 1$ and the shares $[\![x]\!]_0^A, [\![x]\!]_1^A$ are held by two parties respectively. (2) *Binary sharing*: $x = [\![x]\!]_0^B \oplus [\![x]\!]_1^B$, where $x, [\![x]\!]_0^B, [\![x]\!]_1^B \in \mathbb{Z}_2$ and the shares $[\![x]\!]_0^B, [\![x]\!]_1^B$ are held by two parties respectively. Given two secret-shared values $[\![x]\!]^A$ and $[\![y]\!]^A$, and a constant a, the following operations can be supported securely:

- *Linear operation*: Linear operation includes addition/subtraction $[\![x \pm y]\!]^A = [\![x]\!]^A \pm [\![y]\!]^A$ and scalar multiplication $[\![a \cdot x]\!]^A = a \cdot [\![x]\!]^A$. The linear operation can be performed locally and does not require communication.
- *Multiplication*: The multiplication $[\![x \cdot y]\!]^A = [\![x]\!]^A \cdot [\![y]\!]^A$ can be securely performed with the use of Beaver's triples which can be prepared offline in advance. The multiplication between two arithmetic secret-shared values requires one round of online communication.

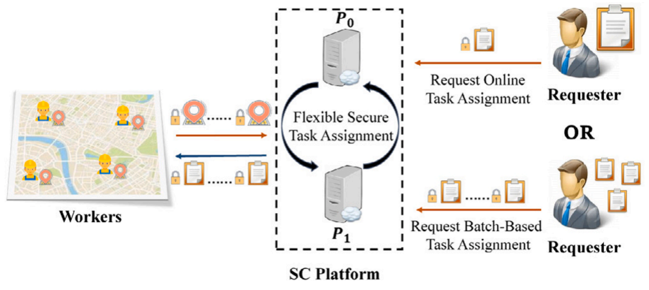

P_0
Workers
Flexible Secure Task Assignment
P_1
SC Platform
Request Online Task Assignment
Requester
OR
Request Batch-Based Task Assignment
Requester

Fig. 1. The system architecture of PPTA.

Note that for binary sharing, the addition can be replaced with XOR ($\oplus$) and the multiplication can be replaced with AND ($\otimes$). To reconstruct a value from its shares, the two parties need to send their own shares $[\![x]\!]_i^{A/B}, i \in \{0, 1\}$ to each other, then locally computes $x = [\![x]\!]_0 + / \oplus [\![x]\!]_1$ to obtain x in plaintext. We denote the reconstruction operation above as $\mathbf{Rec}(\cdot, \cdot)$.

3.2. Function secret sharing

Function secret sharing (FSS) [24,25] allows secure two-party evaluation of non-linear functions (e.g., comparison) with low interactions in the secret sharing domain. Specifically, in a two-party FSS scheme, a target function f is split into two succinct keys. Each key alone does not reveal private information about the target function f. A formal definition is given below:

Definition 1. A two-party FSS scheme consists of two probabilistic polynomial-time (PPT) algorithms (KeyGen, Eval), which are formally defined as follows.

- $(k_0, k_1) \leftarrow$KeyGen$(1^\lambda, f)$: Given a security parameter λ and the function description f, output a pair of succinct FSS keys (k_0, k_1).
- Eval(k_i, x): Given a FSS key k_i and an evaluation point x, output the secret share of the evaluation result $[\![f(x)]\!]_i$.

The security assurance of FSS is that an adversary with access to one of (k_0, k_1) cannot infer any private information about the target function f and the output $f(x)$.

4. Problem statement

4.1. System model

PPTA is aimed at allowing location privacy-preserving and flexible task assignment in SC. As illustrated in Fig. 1, there are three kinds of entities in PPTA: the task requester (abbr. requester), the SC platform, and workers. The requester holds a set of location-dependent tasks, each of which is tagged with a location, a search range r, and a task description. It wants to have the tasks assigned to available workers who can travel to the designated locations to perform the tasks. Each task can only be assigned to at most k workers whose distances to the task are within its search range r. Such task assignment is facilitated by the SC platform, which serves as a bridge between the requester and the workers. To this end, the SC platform needs to collect the locations of tasks and workers and then perform task assignment according to designated mechanisms.

However, for privacy concerns, the workers may not want to expose their locations in the service, which thus poses a demand for protecting the locations of both the tasks and workers. That is, the functionality of task assignment should be supported without the SC platform seeing the actual locations of the tasks and workers. Therefore, in PPTA the locations of tasks and workers are provided to the SC platform in

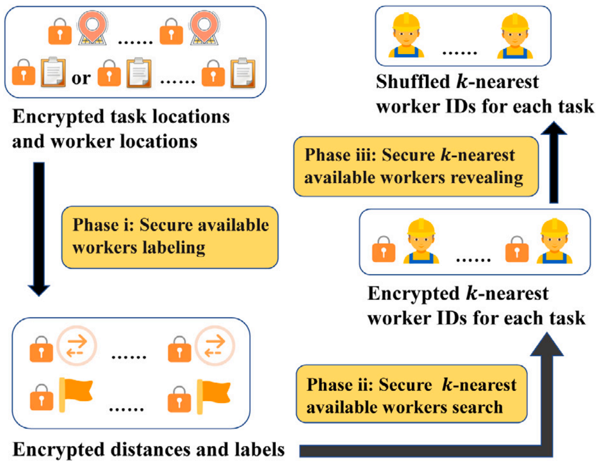

Encrypted task locations and worker locations
or
Phase i: Secure available workers labeling
Encrypted distances and labels
Phase ii: Secure k-nearest available workers search
Encrypted k-nearest worker IDs for each task
Phase iii: Secure k-nearest available workers revealing
Shuffled k-nearest worker IDs for each task

Fig. 2. Illustration of the secure task assignment workflow in PPTA for both the online setting and the batch-based setting.

encrypted form for protection. For high efficiency, PPTA resorts to the lightweight cryptographic technique named additive secret sharing for location encryption and for supporting subsequent secure processing for task assignment at the SC platform. For compatibility with the working paradigm of additive secret sharing, the SC platform in PPTA is jointly run by two SC service providers (denoted by $\mathcal{P}_0$ and $\mathcal{P}_1$), which collaboratively empower the location privacy-preserving SC service. Such two-server model has seen increasing adoption in both academia [14–16,26,27] as well as in industry [17]. PPTA follows such trend and contributes a new design point to location privacy-preserving and flexible task assignment in SC.

To cater for realistic demands, PPTA is designed to flexibly support two realistic settings for location privacy-preserving task assignment: the *online setting* and the *batch-based setting*. In the online setting, the requester delivers the tasks one by one to the SC platform. Once a task arrives, the SC platform will assign it to available workers whose distance is less than its search range r. The aim is to minimize the average distance required by workers to travel to the task location. In this paper we follow the online assignment strategy in [5], which assigns each task to k-nearest available workers. Note that in the online setting, once a worker is assigned with a task, it will be unavailable for the upcoming tasks.

In the batch-based setting, the requester delivers tasks batch by batch to the SC platform. Given a batch of tasks $\mathcal{T}$, a set of workers $\mathcal{W}$, task assignment in the batch-based setting [7,8] aims to find an optimal match $\mathcal{O} = \left\{(w_i, t_j) \mid w_i \in \mathcal{W}, t_j \in \mathcal{T}, dist\left(w_i, t_j\right) \le r_j\right\}$ such that $|\mathcal{O}(t_j)| \le k$, $|\mathcal{O}(w_i)| \le b$ and $|\mathcal{O}|$ is maximized, where $dist(\cdot)$ is the Euclidean distance function, b is the maximum number of tasks a worker can handle, $\mathcal{O}(t_j)$ is the set of workers assigned with task t_j, $\mathcal{O}(w_i)$ is the set of tasks assigned to worker w_i, and $|\mathcal{O}|$ is the number of matched worker-task pairs.

PPTA provides customized secure protocols (see Section 5 and Section 6 respectively) to embrace both settings for location-privacy task assignment in SC. It is noted that our protocols will focus on introducing how to support task assignment given that the locations of tasks and workers are protected. For each task's description, the requester can simply secret-share it and distribute the shares among $\mathcal{P}_0$ and $\mathcal{P}_1$. Once the (secure) task assignment process is completed, the shares of the task descriptions and task locations can be delivered to the corresponding workers.

4.2. Threat model

Along the service flow in PPTA, we consider that the threats to worker location privacy primarily come from the SC service providers, which are assumed to be honest-but-curious and non-colluding adversaries. In particular, each SC service provider in PPTA will honestly follow our protocols but may *individually* attempt to infer worker locations and task locations (which, if known, can also reveal information regarding the locations of workers assigned to the corresponding tasks). PPTA is thus aimed at providing protections for the worker locations as well as task locations throughout the task assignment process. The SC service providers can only learn the task assignment results, i.e., which workers are assigned to a specified task. Following prior works in location privacy-preserving SC [8,18], we assume no collusion among the parties in PPTA.

5. Location privacy-preserving task assignment in the online setting

5.1. Overview

For the online setting, each registered worker w_i with identifier id_i first distributes the secret shares of its location $(\llbracket x_i \rrbracket^A, \llbracket y_i \rrbracket^A)$ to the SC service providers $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively, where $i \in [1, N]$ and N is the number of workers. Note that the locations here are assumed to in Cartesian coordinates, which can be converted from the raw GIS coordinates (i.e., longitude and latitude) using a common technique—Miller cylindrical projection [28]. Hereafter, for ease of presentation, we will refer to $\mathcal{P}_0$ and $\mathcal{P}_1$ simply as $\mathcal{P}_{\{0,1\}}$. The IDs of all workers (denoted as $\mathcal{W}$) are represented by the set $\mathcal{I} = \{id_i\}_{i\in[1,N]}$ and the encrypted locations of all workers are represented by the set $\llbracket \mathcal{H} \rrbracket^A = \{(\llbracket x_i \rrbracket^A, \llbracket y_i \rrbracket^A)\}_{i\in[1,N]}$. It is noted that hereafter we will omit the subscript $i \in [1, N]$ for sets when it does not affect the presentation. For a task t, the requester distributes the secret shares of its location $\llbracket \tau \rrbracket^A = (\llbracket a \rrbracket^A, \llbracket b \rrbracket^A)$ and its encrypted search range $\llbracket r \rrbracket^A$ to $\mathcal{P}_{\{0,1\}}$, respectively. Upon receiving the encrypted location of task t, $\mathcal{P}_{\{0,1\}}$ securely assign the task to appropriate workers as per the design of PPTA. Recall that task assignment in the online setting aims to find k workers from the available workers to minimize the average distance for workers to travel to the task location.

As illustrated in Fig. 2, our construction of secure task assignment in the online setting is comprised of three components: (i) secure available workers labeling (denoted as secLabel); (ii) secure k-nearest available workers search (denoted as secSearch); (iii) secure k-nearest available workers revealing (denoted as secReveal). At a high level, secure task assignment in the online setting proceeds as follows in PPTA. Given the encrypted location $(\llbracket a \rrbracket^A, \llbracket b \rrbracket^A)$ of task t and the encrypted location $(\llbracket x_i \rrbracket^A, \llbracket y_i \rrbracket^A)$ of each worker w_i, PPTA provides secLabel to have the SC platform securely calculate the encrypted distance between the task t and w_i, and then obliviously label available workers whose distance from the task is within its search range r. Then, based on the encrypted distances and labels, PPTA provides secSearch to allow the SC platform to securely fetch k-nearest available workers for the task, which produces the encrypted IDs of the k-nearest available workers. Afterwards, to protect the order information about which workers in the k-nearest available workers are closer to the task, PPTA provides secReveal to allow the SC platform to securely reveal the k-nearest available workers' IDs. In Algorithm 1, we give an overview of PPTA's complete construction for secure task assignment in the online setting, which relies on the coordination of the three components: secLabel, secSearch, and secReveal. In what follows, we elaborate on the design of each component.

5.2. Secure available workers labeling

Given the secret sharings of the task location and the worker locations, the secure available workers labeling component secLabel is to compute a secret-shared label for each worker, the value of which indicates whether the corresponding worker is an available one (i.e., its distance to the task location is within its search range). Algorithm 2

Algorithm 1 Secure Task Assignment in the Online Setting

```
Input: Task t's encrypted location $[\![\tau]\!]^A$ and its encrypted search range $[\![r]\!]^A$, registered workers' ID set $\mathcal{I}$ and encrypted location set $[\![\mathcal{H}]\!]^A$.
Output: $k$-nearest available workers' IDs $\hat{\mathbf{I}}^\star$ for task $t$.
    # Phase (i): secure available workers labeling:
1:  $([\![\mathcal{L}]\!]^B, [\![\mathcal{D}]\!]^A) = \mathsf{secLabel}([\![\tau]\!]^A, [\![\mathcal{H}]\!]^A, [\![r]\!]^A)$.
    # Phase (ii): secure $k$-nearest workers search:
2:  $[\![\mathbf{I}^\star]\!]^A = \mathsf{secSearch}([\![\mathcal{L}]\!]^B, [\![\mathcal{D}]\!]^A, \mathcal{I})$.
    # Phase (iii): secure $k$-nearest workers revealing:
3:  $\hat{\mathbf{I}}^\star = \mathsf{secReveal}([\![\mathbf{I}^\star]\!]^A)$.
4:  Outputting $k$-nearest available workers' ID $\hat{\mathbf{I}}^\star$ for task $t$.
```

Algorithm 2 Secure Available Workers Labeling secLabel

```
Input: Task t's encrypted location $[\![\tau]\!]^A$ and its encrypted search range $[\![r]\!]^A$, registered workers' encrypted location set $[\![\mathcal{H}]\!]^A$.
Output: Encrypted label set $[\![\mathcal{L}]\!]^B$ and encrypted distance set $[\![\mathcal{D}]\!]^A$.
1:  Initializing two empty sets $[\![\mathcal{L}]\!]^B = \emptyset$ and $[\![\mathcal{D}]\!]^A = \emptyset$.
2:  for all $i \in [1, N]$ do
3:      $[\![d_i]\!]^A = ([\![x_i]\!]^A - [\![a]\!]^A)^2 + ([\![y_i]\!]^A - [\![b]\!]^A)^2$.
4:      $[\![r^2]\!]^A = [\![r]\!]^A \cdot [\![r]\!]^A$.
5:      $[\![l_i]\!]^B = [\![(d_i - r^2 \le 0)]\!]^B$.
6:      $[\![\mathcal{D}]\!]^A.add([\![d_i]\!]^A)$.
7:      $[\![\mathcal{L}]\!]^B.add([\![l_i]\!]^B)$.
8:  end for
9:  Outputting the encrypted label set $[\![\mathcal{L}]\!]^B$ and encrypted distance set $[\![\mathcal{D}]\!]^A$.
```

provides the detailed construction of secLabel. Specifically, $\mathcal{P}_{\{0,1\}}$ first securely compute the encrypted distance (i.e., $[\![d_i]\!]^A$) between the task t and each worker w_i, and then securely compute the encrypted label $[\![l_i]\!]^B$ for each worker w_i. Here, $l_i = 1$ if the distance between w_i and t is within r, i.e., w_i is within the search range r and so w_i is an available worker. Otherwise, $l_i = 0$. All workers' labels are represented by $[\![\mathcal{L}]\!]^B = \{[\![l_i]\!]^B\}$ and distances from the task t are represented by $[\![\mathcal{D}]\!]^A = \{[\![d_i]\!]^A\}$. Therefore, the procedure is comprised of two steps: (i) secure distance calculation and (ii) secure worker labels calculation, which are introduced below.

Secure distance calculation. In the plaintext domain, given the locations $((a,b), (x_i, y_i))$ of the task and worker w_i, the commonly used distance metric is the Euclidean distance:

$$\hat{d}_i = \sqrt{(x_i - a)^2 + (y_i - b)^2}. \tag{1}$$

However, the square root operation is not naturally supported in secret sharing domain. We note that in our context, only the *comparison results* between distances are required instead of the distances. Therefore, PPTA utilizes a roundabout strategy to omit the square root operation in Eq. (1). Specifically, PPTA uses the *squared Euclidean distance* between the task t and each worker w_i, and then compares their squared Euclidean distance (i.e., d_i) with r^2 to evaluate whether w_i is an available worker. Correctness holds since $\hat{d}_i \le r$ is equivalent to $d_i \le r^2$. Therefore, PPTA lets $\mathcal{P}_{\{0,1\}}$ securely calculate the squared Euclidean distance between the task t and each worker w_i by

$$[\![d_i]\!]^A = ([\![x_i]\!]^A - [\![a]\!]^A)^2 + ([\![y_i]\!]^A - [\![b]\!]^A)^2. \tag{2}$$

It is noted that due to the correctness of calculation in the secret sharing domain, the magnitude relationship is consistent in the plaintext domain and the secret sharing domain (i.e., Eq. (2)). Since addition/subtraction and squaring (i.e., multiplication) operations are naturally supported in secret sharing domain, $\mathcal{P}_{\{0,1\}}$ can securely calculate the squared Euclidean distance between the task and each worker, as well as the squared search range $[\![r^2]\!]^A$.

Secure worker labels calculation. After securely calculating the encrypted squared Euclidean distance $[\![d_i]\!]^A$ between the task t and each worker w_i, $\mathcal{P}_{\{0,1\}}$ should obliviously evaluate the availability of each worker. Our insight is to compute an encrypted label (i.e., a bit) $[\![l_i]\!]^B$ for each worker w_i, where $[\![l_i]\!]^B = [\![d_i - r^2 \le 0]\!]^B$. Then we should consider how to allow $\mathcal{P}_{\{0,1\}}$ to securely evaluate $[\![d_i - r^2 \le 0]\!]^B$ given the encrypted squared Euclidean distance $[\![d_i]\!]^A$ and the squared search range $[\![r^2]\!]^A$.

We identify the state-of-the-art construction of FSS-based secure comparison [24] as a promising choice, which is named as distributed comparison function (DCF) and allows low-interaction secure comparison in the secret sharing domain. FSS-based DCF can securely compute the function $g^{\le}_{\alpha,\beta}(x)$, which outputs β if the input $x \le \alpha$ and outputs 0 if the input $x > \alpha$. To apply DCF for secure comparison in a two-party scenario, each party should hold a DCF key $k_i, i \in \{0,1\}$ for the function $g^{\le}_{\alpha,\beta}(\cdot)$ and then evaluate the DCF key over the public input $x \in \mathbb{Z}_{2^l}$ by invoking $\mathsf{Eval}(k_i, x)$. This allows each party to produce a share of the output $g^{\le}_{\alpha,\beta}(x)$.

However, since the FSS-based DCF evaluation requires the two parties to work on identical inputs, the evaluation of DCF over a private (secret-shared) value cannot be achieved by the above process and requires a delicate treatment. To deal with the challenge, PPTA keeps the private value x secret by letting the two parties open the masked version of x, inspired by [24]. More specifically, given the target function $g^{\le}_{\alpha,\beta}$, a pair of DCF keys for $g^{\le}_{\alpha,\beta}$'s offset function $g^{\le}_{\alpha,\beta,\gamma}$ is generated, where $g^{\le}_{\alpha,\beta}(x) = g^{\le}_{\alpha,\beta,\gamma}(x+\gamma)$, and γ is a random value and is split into additive secret shares $\gamma = [\![\gamma]\!]^A_1 + [\![\gamma]\!]^A_2$, each for one party. After that, each party sends the masked share $[\![x]\!]^A_i + [\![\gamma]\!]^A_i$ to each other to open $x+\gamma$ without leaking x. Finally, the two parties securely evaluate the FSS keys about the offset function $g^{\le}_{\alpha,\beta,\gamma}$ on the public input $x+\gamma$ to output the secret sharing of $g^{\le}_{\alpha,\beta,\gamma}(x+\gamma)$, where $g^{\le}_{\alpha,\beta,\gamma}(x+\gamma) = g^{\le}_{\alpha,\beta}(x)$.

We now show how PPTA builds on the DCF to allow $\mathcal{P}_{\{0,1\}}$ to securely evaluate $[\![d_i - r^2 \le 0]\!]^B$ given the encrypted squared Euclidean distance $[\![d_i]\!]^A$ and the squared search range $[\![r^2]\!]^A$. To use the FSS-based DCF, PPTA sets the input domain as $\mathbb{Z}_{2^l}$, $\alpha = 0$, and the output domain as $\mathbb{Z}_2$ and $\beta = 1$. With these parameters, the DCF keys k_0 and k_1 and the secret shares $[\![\gamma]\!]^A_1, [\![\gamma]\!]^A_2$ of the random offset γ can be prepared offline and distributed to $\mathcal{P}_{\{0,1\}}$, respectively. Note that such offline preparatory work can be done by a third-party in practice [24]. Upon receiving the DCF key k_c and the secret share $[\![\gamma]\!]^A_c, c \in \{0,1\}$, to securely evaluate $[\![d_i - r^2 \le 0]\!]^B$, $\mathcal{P}_c$ first locally computes $[\![d_i - r^2]\!]^A$ and exchanges $[\![d_i - r^2]\!]^A_c + [\![\gamma]\!]^A_c$ to reveal $d_i - r^2 + \gamma$, and then evaluates $\mathsf{Eval}(k_c, d_i - r^2 + \gamma)$, which will output $[\![1]\!]^B_c$ if $d_i \le r^2$. Otherwise, $[\![0]\!]^B_c$ is outputted.

5.3. Secure k-nearest available workers search

We now introduce how $\mathcal{P}_{\{0,1\}}$ obliviously search for k-nearest available workers for task t. Our solution is to securely instantiate the bubble sort [29] in the secret sharing domain to allow $\mathcal{P}_{\{0,1\}}$ to obliviously fetch k-nearest available workers' encrypted IDs (denoted as an encrypted array $[\![\mathbf{I}^\star]\!]^A = [[\![id^\star_1]\!]^A, \ldots, [\![id^\star_k]\!]^A]$), without knowing the relative proximity of each worker to task t. It is noted that since the number of available workers is likely to be less than k, $\mathbf{I}^\star$ may contain some dummy IDs. We use -1 to represent dummy ID, which is distinguished from all other IDs of workers in the system.

A plausible approach is to let $\mathcal{P}_{\{0,1\}}$ first sort workers' encrypted IDs based on their encrypted distances (i.e., $[\![\mathcal{D}]\!]^A$) to task t and then fetch the first k IDs from the sorted encrypted IDs as $[\![\mathbf{I}^\star]\!]^A$. However, if the number of available workers is less than k, $[\![\mathbf{I}^\star]\!]^A$ generated by the method may contain the IDs of unavailable workers, which will result in incorrect assignment. Therefore, PPTA lets $\mathcal{P}_{\{0,1\}}$ obliviously fetch $[\![\mathbf{I}^\star]\!]^A$ via a different strategy.

At a high level, PPTA lets $\mathcal{P}_{\{0,1\}}$ set each ID $id^\star_i$ in $[\![\mathbf{I}^\star]\!]^A$ to $[\![-1]\!]^A$ at the beginning. Meanwhile, PPTA lets $\mathcal{P}_{\{0,1\}}$ define an encrypted

Algorithm 3 Secure k-Nearest Available Workers Search secSearch

```
Input: Encrypted label set $[\![\mathcal{L}]\!]^B$; encrypted distance set $[\![\mathcal{D}]\!]^A$; public ID set $\mathcal{I}$.
Output: $k$-nearest workers' encrypted IDs $[\![\mathbf{I}^\star]\!]^A$.
1: Initializing $[\![\mathbf{I}^\star]\!]^A = [[\![id_1^\star]\!]^A, \cdots, [\![id_k^\star]\!]^A]$ and $[\![\mathbf{D}^\star]\!]^A = [[\![d_1^\star]\!]^A, \cdots, [\![d_k^\star]\!]^A]$, where $[\![id_i^\star]\!]^A = [\![-1]\!]^A$ and $[\![d_i^\star]\!]^A = [\![MAX]\!]^A, i \in [1,k]$.
2: Parsing $\mathcal{I}$ into secret sharing $[\![\mathcal{I}]\!]^A$.
3: for all $i \in [1,k]$ do
4:     for all $j \in [1,N]$ do
5:         $[\![\delta]\!]^B = [\![d_j < d_i^\star]\!]^B$. # by FSS-based DCF
6:         $[\![\delta']\!]^B = [\![\delta]\!]^B \otimes [\![l_j]\!]^B$ .
           # Obliviously swapping $[\![id_i^\star]\!]^A$ and $[\![id_j]\!]^A$:
7:         $[\![\rho]\!]^A = [\![id_i^\star]\!]^A + [\![\delta']\!]^B \cdot ([\![id_j]\!]^A - [\![id_i^\star]\!]^A)$.
8:         $[\![\varphi]\!]^A = [\![id_j]\!]^A + [\![\delta']\!]^B \cdot ([\![id_i^\star]\!]^A - [\![id_j]\!]^A)$.
9:         $[\![id_i^\star]\!]^A = [\![\rho]\!]^A$;   $[\![id_j]\!]^A = [\![\varphi]\!]^A$.
10:        Obliviously swapping $[\![d_i^\star]\!]^A$ and $[\![d_j]\!]^A$ based on $[\![\delta']\!]^B$ by the same way.
11:    end for
12: end for
13: Outputting $k$-nearest workers' encrypted IDs $[\![\mathbf{I}^\star]\!]^A$.
```

array $[\![\mathbf{D}^\star]\!]^A = [[\![d_1^\star]\!]^A, \ldots, [\![d_k^\star]\!]^A]$ to store k-nearest available workers' encrypted distances to task t, where $[\![d_i^\star]\!]^A$ is set to $[\![MAX]\!]^A$ at the beginning and MAX is a pre-set system-wide maximum value. After that, given each $[\![id_i^\star]\!]^A, [\![d_i^\star]\!]^A, i \in [1,k]$, PPTA provides techniques to allow $\mathcal{P}_{\{0,1\}}$ to obliviously compare $[\![d_i^\star]\!]^A$ and each $[\![d_j]\!]^A \in [\![\mathcal{D}]\!]^A$, and then obliviously swap $([\![id_i^\star]\!]^A, [\![d_i^\star]\!]^A)$ and $([\![id_j]\!]^A, [\![d_j]\!]^A)$ when the underlying $d_j < d_i^\star$ **and** $l_j = 1$ (i.e., worker w_j is an available worker). Obviously, if the number of available workers is less than k, there will be some $[\![-1]\!]^A$ in $[\![\mathbf{I}^\star]\!]^A$. Therefore, after *securely* revealing $[\![\mathbf{I}^\star]\!]^A$ (as later introduced in Section 5.4), $\mathcal{P}_{\{0,1\}}$ can filter out invalid IDs to obtain the IDs of matched available workers.

We next introduce how to securely realize the above idea. Firstly, the secure comparison between $[\![d_i^\star]\!]^A$ and $[\![d_j]\!]^A$ can be realized by the FSS-based DCF introduced in Section 5.2. We represent the comparison as $[\![\delta]\!]^B = [\![d_j < d_i^\star]\!]^B$. It is noted that the FSS-based DCF $g^{<}_{\alpha,\beta}(x)$ is analogous to $g^{\leq}_{\alpha,\beta}(x)$. However, $\mathcal{P}_{\{0,1\}}$ cannot simply swap $([\![id_i^\star]\!]^A, [\![d_i^\star]\!]^A)$ and $([\![id_j]\!]^A, [\![d_j]\!]^A)$ based on $[\![\delta]\!]^B$ because the worker w_j may not be an available worker, i.e., w_j is out of the search range r. Therefore, PPTA further lets $\mathcal{P}_{\{0,1\}}$ AND $[\![\delta]\!]^B$ by w_j's label $[\![l_j]\!]^B$. Formally, $\mathcal{P}_{\{0,1\}}$ perform the following

$$[\![\delta']\!]^B = [\![\delta]\!]^B \otimes [\![l_j]\!]^B,$$

where $\delta' = 1$ indicates that the worker w_j is an available worker as well as closer to the task t than the worker with ID $id_i^\star$. Correctness holds since $\delta' = 1$ if and only if $\delta = 1$ and $l_j = 1$, where $\delta = 1$ indicates that w_j is closer to the task and $l_j = 1$ indicates that w_j is an available worker. We then should consider how to allow $\mathcal{P}_{\{0,1\}}$ to obliviously swap $([\![id_i^\star]\!]^A, [\![d_i^\star]\!]^A)$ and $([\![id_j]\!]^A, [\![d_j]\!]^A)$ based on $[\![\delta']\!]^B$.

Our main idea is to first transform the secure swapping operation into:

$$\begin{cases} [\![\rho]\!]^A = [\![id_i^\star]\!]^A + [\![\delta']\!]^B \cdot ([\![id_j]\!]^A - [\![id_i^\star]\!]^A); \\ [\![\varphi]\!]^A = [\![id_j]\!]^A + [\![\delta']\!]^B \cdot ([\![id_i^\star]\!]^A - [\![id_j]\!]^A); \\ [\![id_i^\star]\!]^A = [\![\rho]\!]^A; \quad [\![id_j]\!]^A = [\![\varphi]\!]^A, \end{cases}$$

where ρ and φ are two temporary variables to store the swapped values. The secure swapping of $[\![d_i^\star]\!]^A$ and $[\![d_j]\!]^A$ is similar to the above process. It is noted that the addition and subtraction are naturally supported in the secret sharing domain. The multiplication between a binary secret-shared value and an arithmetic secret-shared value (e.g., $[\![\delta']\!]^B \cdot ([\![id_j]\!]^A - [\![id_i^\star]\!]^A)$), however, is not naturally supported in the secret sharing domain and tailored protocol is required.

Algorithm 4 Secure k-Nearest Available Workers Revealing secReveal

```
Input: $k$-nearest workers' encrypted IDs $[\![\mathbf{I}^\star]\!]^A$.
Output: $k$-nearest workers' shuffled IDs $\hat{\mathbf{I}}^\star$ in plaintext.
1: $[\![\hat{\mathbf{I}}^\star]\!]^A$ = secShuffle($[\![\mathbf{I}^\star]\!]^A$).
2: $\mathcal{P}_{\{0,1\}}$ perform $\mathbf{Rec}([\![\hat{\mathbf{I}}^\star]\!]^A_0, [\![\hat{\mathbf{I}}^\star]\!]^A_1)$ to securely reveal the shuffled $\hat{\mathbf{I}}^\star$.
3: Outputting $k$-nearest workers' IDs $\hat{\mathbf{I}}^\star$.
```

Given that $\mathcal{P}_{\{0,1\}}$ hold a binary secret-shared value $[\![\sigma]\!]^B$ and an arithmetic secret-shared value $[\![\mu]\!]^A$, $\mathcal{P}_{\{0,1\}}$ can proceed as follows to securely calculate $[\![\sigma \cdot \mu]\!]^A$ (inspired by [30]). Firstly, $\mathcal{P}_0$ generates a random value $r \in \mathbb{Z}_{2^l}$ and then sends two messages $m_i = (i \oplus [\![\sigma]\!]^B_0) \cdot [\![\mu]\!]^A_0 - r, i \in \{0,1\}$ to $\mathcal{P}_1$. Secondly, $\mathcal{P}_1$ keeps m_0 if $[\![\sigma]\!]^B_1 = 0$, and otherwise $\mathcal{P}_1$ keeps m_1. Therefore, $\mathcal{P}_1$ holds the share $m_{[\![\sigma]\!]^B_1} = \sigma \cdot [\![\mu]\!]^A_0 - r$ and $\mathcal{P}_0$ holds the share r. For the other share $[\![\mu]\!]^A_1$, $\mathcal{P}_1$ in turn acts as the sender and $\mathcal{P}_0$ acts as receiver to perform the above two steps again. At the end of the evaluation, $\mathcal{P}_{\{0,1\}}$ hold the secret sharings of $[\![\sigma \cdot \mu]\!]^A$. With the above tailored design, $\mathcal{P}_{\{0,1\}}$ can obliviously fetch k-nearest available workers' encrypted IDs $[\![\mathbf{I}^\star]\!]^A$, as detailed in Algorithm 3.

5.4. Secure k-nearest available workers revealing

Upon producing the encrypted k-nearest available workers' IDs $[\![\mathbf{I}^\star]\!]^A$, $\mathcal{P}_{\{0,1\}}$ then need to decrypt them to obtain the plaintext IDs $\mathbf{I}^\star$, so as to assign the task t to the corresponding workers. To decrypt the encrypted IDs, a naive method is to let $\mathcal{P}_{\{0,1\}}$ simply exchange the secret shares of $[\![\mathbf{I}^\star]\!]^A$ they hold. Such naive method, however, will leak the orders regarding the matched workers' distances from the task because the bubble sort outputs ordered results. This leakage has been shown to be exploitable for various attacks [22,23]. Therefore, a custom approach is needed to allow $\mathcal{P}_{\{0,1\}}$ to securely reveal the k-nearest workers' IDs without knowing their original orders.

Our key idea is to first have $\mathcal{P}_{\{0,1\}}$ obliviously shuffle the encrypted k-nearest available workers' IDs $[\![\mathbf{I}^\star]\!]^A$, i.e., a shuffle is performed without $\mathcal{P}_{\{0,1\}}$ knowing the permutation, so as to break the orders of IDs in $\mathbf{I}^\star$. Since $[\![\mathbf{I}^\star]\!]^A$ is shuffled, we can let $\mathcal{P}_{\{0,1\}}$ securely reveal the IDs in the shuffled $[\![\mathbf{I}^\star]\!]^A$ (denoted as $[\![\hat{\mathbf{I}}^\star]\!]^A$) to identify which workers are the matched ones for task t. Here what we need is a technique to allow $\mathcal{P}_{\{0,1\}}$ to securely perform the shuffle in the secret sharing domain. In particular, given a secret-shared array with ordered elements $[\![\mathbf{X}]\!]^A$, we need a secret-shared shuffle protocol that allows the SC service providers to collaboratively *shuffle* the elements in $[\![\mathbf{X}]\!]^A$ and produce the secret shares of the shuffled array $[\![\pi(\mathbf{X})]\!]^A$, while no party can learn the permutation $\pi(\cdot)$. We identify that the state-of-the-art construction of secret-shared shuffle from [31] is well suited for our purpose, since it allows secret-shared shuffling on additive secret shares. We encapsulate the secret-shared shuffle protocol as secShuffle, which inputs the ordered array $[\![\mathbf{X}]\!]^A$, and outputs the shuffled array $[\![\pi(\mathbf{X})]\!]^A$.

PPTA adapts secShuffle to instantiate the secure k-nearest available workers revealing process, where the encrypted k-nearest available workers' ID array $[\![\mathbf{I}^\star]\!]^A$ is set as the input array and each encrypted ID in $[\![\mathbf{I}^\star]\!]^A$ is an element in the input array. Algorithm 4 describes our protocol for secure k-nearest workers revealing, which inputs the encrypted IDs $[\![\mathbf{I}^\star]\!]^A$, and outputs the shuffled IDs $\hat{\mathbf{I}}^\star$ in plaintext. It is noted that since the number of available workers is probably less than k, $\hat{\mathbf{I}}^\star$ may contain dummy IDs, each of which is equal to -1 (as introduced above in secSearch). Therefore, $\mathcal{P}_{\{0,1\}}$ can filter IDs from $\hat{\mathbf{I}}^\star$ if they are equal to -1.

Finally, based on the valid IDs in $\hat{\mathbf{I}}^\star$, $\mathcal{P}_{\{0,1\}}$ send the secret shares of the task t's locations and description to the corresponding workers for decryption. Each matched worker then obtains the task description and location, and can move to the specified task location to perform the task according to the task description.

Algorithm 5 Secure Task Assignment in the Batch-based Setting

```
Input: The encrypted location $[\![\tau_j]\!]^A$ and encrypted search range $[\![r_j]\!]^A$ of each $t_j, j \in [1, S]$, workers' ID set $\mathcal{I}$ and encrypted location set $[\![\mathcal{H}]\!]^A$.
Output: $k$-nearest workers' IDs $\hat{\mathbf{I}}_j^\star$ for each task $t_j$.
1: Initializing counter set $\mathcal{C} = \{c_i = 0\}_{i\in[1,N]}$.
2: for all $j \in [1, S]$ do
3:     # Phase i: secure available workers labeling:
4:     $([\![\mathcal{L}_j]\!]^B, [\![\mathcal{D}_j]\!]^A) = \mathsf{secLabel}^\diamond([\![\tau_j]\!]^A, [\![\mathcal{H}]\!]^A, [\![r_j]\!]^A, \mathcal{C})$.
       # Phase ii: secure $k$-nearest available workers search:
5:     $[\![\mathbf{I}_j^\star]\!]^A = \mathsf{secSearch}([\![\mathcal{L}_j]\!]^B, [\![\mathcal{D}_j]\!]^A, \mathcal{I})$.
       # Phase iii: secure $k$-nearest available workers revealing:
6:     $\hat{\mathbf{I}}_j^\star = \mathsf{secReveal}([\![\mathbf{I}_j^\star]\!]^A)$.
7:     for all $id^\star \in \hat{\mathbf{I}}_j^\star$ do
8:         Updating the counter $c_i$ corresponding to the worker with ID $id^\star$: $c_i = c_i + 1$.
9:     end for
10: end for
11: Outputting $k$-nearest workers' IDs $\hat{\mathbf{I}}_j^\star$ for each task $t_j$.
```

Algorithm 6 Secure Available Workers Labeling $\mathsf{secLabel}^\diamond$

```
Input: Task $t$'s encrypted location $[\![\tau]\!]^A$ and its encrypted search range $[\![r]\!]^A$, registered workers' encrypted location set $[\![\mathcal{H}]\!]^A$, and counter set $\mathcal{C}$.
Output: Workers' encrypted label set $[\![\mathcal{L}]\!]^B$ and encrypted distance set $[\![\mathcal{D}]\!]^A$.
1: Initializing two empty sets $[\![\mathcal{L}]\!]^B = \emptyset$ and $[\![\mathcal{D}]\!]^A = \emptyset$.
2: for all $i \in [1, N]$ do
3:     $[\![d_i]\!]^A = ([\![x_i]\!]^A - [\![a]\!]^A)^2 + ([\![y_i]\!]^A - [\![b]\!]^A)^2$.
4:     $[\![r^2]\!]^A = [\![r]\!]^A \cdot [\![r]\!]^A$.
5:     $\underline{[\![l_i]\!]^B = [\![(d_i - r^2 \le 0)]\!]^B \otimes (c_i < b)}$.
6:     $[\![\mathcal{D}]\!]^A.add([\![d_i]\!]^A)$.
7:     $[\![\mathcal{L}]\!]^B.add([\![l_i]\!]^B)$.
8: end for
9: Outputting the encrypted label set $[\![\mathcal{L}]\!]^B$ and encrypted distance set $[\![\mathcal{D}]\!]^A$.
```

6. Location privacy-preserving task assignment in the batch-based setting

In this section, we present our secure task assignment protocol for the batch-based setting. As aforementioned, in such setting, the requester will distribute a batch of tasks to the SC platform. Once receiving a task batch $\mathcal{T} = \{t_j, j \in [1, S]\}$, the SC platform will perform task assignment to output an optimal match $\mathcal{O} = \left\{\left(w_i, t_j\right) | w_i \in \mathcal{W}, t_j \in \mathcal{T}, d\left(w_i, t_j\right) \le r_j\right\}$ such that $|\mathcal{O}(t_j)| \le k$, $|\mathcal{O}(w_i)| \le b$ and $|\mathcal{O}|$ is maximized, where $\mathcal{O}(t_j)$ is the set of workers assigned with task t_j, $\mathcal{O}(w_i)$ is the set of tasks assigned to worker w_i, and $|\mathcal{O}|$ is the number of matched worker-task pairs.

The problem of optimal task assignment in the batch-based setting can be transformed into the maximum flow (Max-Flow) problem [32]. However, although the Max-Flow based approach can produce the optimal result, directly applying it may lead to high performance overheads [7,32]. Therefore, we turn to leverage a greedy method to design our secure task assignment protocol in the batch-based setting, inspired by [7]. The basic idea of the greedy method for task assignment in the batch-based setting is as follows. Given each $t_j \in \mathcal{T}$ in turn, the SC platform selects k-nearest workers for t_j from the workers who are available workers for t_j **and** have been assigned less than b tasks until now. As will be later shown in the experiments, the greedy method based secure task assignment in the batch-based setting can generate results with quality similar to the optimal results generated by the Max-Flow based approach.

Algorithm 5 shows our design for secure task assignment in the batch-based setting. Given the encrypted location $[\![\tau_j]\!]^A = ([\![a_j]\!]^A, [\![b_j]\!]^A)$ of task $t_j \in \mathcal{T}$ and the encrypted location $([\![x_i]\!]^A, [\![y_i]\!]^A)$ of each worker $w_i \in \mathcal{W}$, similar to the secure task assignment in the online setting, $\mathcal{P}_{\{0,1\}}$ need to first obliviously label the available workers who are within the search range r_j. However, since in the batch-based setting, each worker is limited to a maximum of b tasks, the secure available workers labeling component secLabel in online setting (i.e., Algorithm 2) cannot be directly used in batch-based setting and a delicate treatment is required.

Our solution is to let $\mathcal{P}_{\{0,1\}}$ maintain a public counter c_i for each worker w_i. At the beginning of the secure task assignment, $c_i = 0$. Afterwards, at the end of assignment for each task $t_j \in \mathcal{T}$, if a worker w_i is assigned to task t_j, the counter c_i of worker w_i will be set as $c_i + 1$. During the process of secure available workers labeling, instead of letting $\mathcal{P}_{\{0,1\}}$ obliviously label a worker w_i by $[\![l_i]\!]^B = [\![(d_i - r^2 \le 0)]\!]^B$ as in the online setting, PPTA lets $\mathcal{P}_{\{0,1\}}$ obliviously label w_i by

$$[\![l_i]\!]^B = [\![(d_i - r^2 \le 0)]\!]^B \otimes (c_i < b),$$

where $(c_i < b) = 1 \in \mathbb{Z}_2$ if $c_i < b$. With this way, if and only if the worker w_i is within the search range and its counter $c_i < b$ (i.e., the number of tasks assigned to w_i is still smaller than b), w_i's encrypted label $[\![l_i]\!]^B$ will be set to $[\![1]\!]^B$. We present our new component for secure available workers labeling in Algorithm 6 (denoted as $\mathsf{secLabel}^\diamond$), which inputs the encrypted location $[\![\tau]\!]^A = ([\![a]\!]^A, [\![b]\!]^A)$ and the search range r of a task $t \in \mathcal{T}$, workers' encrypted location set $[\![\mathcal{H}]\!]^A$, and counter set $\mathcal{C} = \{c_i\}$, and outputs workers' encrypted label set $[\![\mathcal{L}]\!]^B$ and encrypted distance set $[\![\mathcal{D}]\!]^A$ for the task t. Note that Algorithm 6 is same as Algorithm 2 except for line 5.

After obliviously labeling available workers, $\mathcal{P}_{\{0,1\}}$ securely search k-nearest available workers for the task t_j by the component secSearch (i.e., Algorithm 3), which inputs the label set $[\![\mathcal{L}_j]\!]^B$, the distance set $[\![\mathcal{D}_j]\!]^A$ and public ID set $\mathcal{I}$, and outputs the k-nearest available workers' encrypted IDs $[\![\mathbf{I}_j^\star]\!]^A$ for task t_j. After that, $\mathcal{P}_{\{0,1\}}$ securely reveal the encrypted IDs $[\![\mathbf{I}_j^\star]\!]^A$ by the secure component secReveal (i.e., Algorithm 4), which inputs the encrypted IDs $[\![\mathbf{I}_j^\star]\!]^A$, and outputs the shuffled IDs $\hat{\mathbf{I}}_j^\star$ in plaintext.

After $\mathcal{P}_{\{0,1\}}$ obtain k-nearest available worker IDs $\hat{\mathbf{I}}_j^\star$ for each task $t_j \in \mathcal{T}$, $\mathcal{P}_{\{0,1\}}$ send the secret shares of each task t_j's locations and description to the corresponding workers for decryption. After the k-nearest available workers obtain the task description and location, they move to the specified location to perform the task based on the description.

Remark. It is noted that although the secure task assignment algorithms for the online setting and the batch-based setting are not the same, the workers only need to upload their secret-shared locations once, and the remaining processing of secure task assignment under different settings is fully performed at the SC platform, which will decide which algorithms to use based on the specific setting. Therefore, PPTA does ensure that task assignment can be securely performed with workers being agnostic to the specific settings.

7. Security analysis

Our secure task assignment protocols guarantee that throughout the process of task assignment, $\mathcal{P}_0$ and $\mathcal{P}_1$ only know the assignment result for tasks, i.e., which workers are the eligible ones for a specific task. Next, we follow the standard simulation-based paradigm [33] to prove the confidentiality that PPTA provides for the locations used for task assignment in SC. We first analyze the online setting. We start with defining the corresponding ideal functionality $\mathcal{F}$, which comprises the following parts:

- **Input.** The requester submits its task t to $\mathcal{F}$, and each worker submits its ID and location to $\mathcal{F}$.

- **Task assignment.** Upon receiving task t from the requester, and the ID and location from each worker, $\mathcal{F}$ performs task assignment to search k-nearest available workers for task t. After that, $\mathcal{F}$ sends the task description to the assigned workers.
- **Output.** The assigned workers perform the task based on its requirement at the specified location.

Let $\prod$ denote the protocol for securely realizing the ideal functionality $\mathcal{F}$. $\prod$'s security is formally defined as follows.

Definition 2. Let $\text{view}_{\Pi}^{\mathcal{P}_i}$ denote each $\mathcal{P}_i$'s view during the execution of Π. We say that Π is secure in the semi-honest and non-colluding setting, if for each corrupted $\mathcal{P}_i$ there exists a PPT simulator who can generate a simulated view $\text{Sim}^{\mathcal{P}_i}$ such that $\text{Sim}^{\mathcal{P}_i}$ is indistinguishable from $\text{view}_{\Pi}^{\mathcal{P}_i}$.

Theorem 1. *In the semi-honest and non-colluding threat model and given the security of ASS and FSS, PPTA securely realizes the ideal functionality $\mathcal{F}$ according to Definition 2.*

Proof. Recall that our protocol for secure task assignment in the online setting (i.e., Algorithm 1) consists of three subroutines: i) secure available workers labeling secLabel (i.e., Algorithm 2); ii) secure k-nearest available workers search secSearch (i.e., Algorithm 3); iii) secure k-nearest available workers revealing secReveal (i.e., Algorithm 4). Each subroutine in Algorithm 1 is invoked in order as per the processing pipeline and their inputs are secret shares that are indistinguishable form random values. If the simulator for each subroutine exists, PPTA is secure [34–36]. We next analyze the existence of simulators for the three subroutines in turn. It is noted that since the roles of $\mathcal{P}_{\{0,1\}}$ in PPTA are symmetric, it suffices to prove the existence of simulators for $\mathcal{P}_0$.

- **Simulator for $\mathcal{P}_0$ in** secLabel. Since the operations apart from line 5 in Algorithm 2 are basic additions and multiplications over additive secret shares, we only prove that the simulator for $\mathcal{P}_0$ exists in the execution of line 5, i.e., $[\![(d_i - r^2 \le 0)]\!]^B$. At the beginning, $\mathcal{P}_0$ holds the FSS-based DCF key k_0 and the secret share $[\![d_i - r^2]\!]_0^A$, and later receives the masked secret share $[\![d_i - r^2]\!]_1^A + [\![\gamma]\!]_1^A$ from $\mathcal{P}_1$, followed by outputting the evaluation result $[\![(d_i - r^2 \le 0)]\!]_0^B$. Since all the information $\mathcal{P}_0$ receives is all legitimate in FSS-based DCF, the simulator for $\mathcal{P}_0$ can be trivially constructed by invoking the simulator of FSS-based DCF. From the security of ASS [13] and FSS [24], the simulator for $\mathcal{P}_0$ in secLabel exists.
- **Simulator for $\mathcal{P}_0$ in** secSearch. Note that the operations apart from line 5, line 7, and line 8 in Algorithm 3 are basic additions and multiplications over additive secret shares, we only prove that the simulator for $\mathcal{P}_0$ exists in the execution of line 5, i.e., $[\![d_j < d_i^\star]\!]^B$, and line 7, and line 8, i.e., the multiplication between a binary secret-shared value and an arithmetic secret-shared value, denoted as $[\![\sigma \cdot \mu]\!]^A$ given that $\mathcal{P}_{\{0,1\}}$ hold $[\![\sigma]\!]^B$ and $[\![\mu]\!]^A$. Since $[\![d_j < d_i^\star]\!]^B$ is the standard use of FSS-based DCF, the simulator for $\mathcal{P}_0$ can be trivially constructed by invoking the simulator of FSS-based DCF. We then analyze the operation $[\![\sigma \cdot \mu]\!]^A$. At the beginning, $\mathcal{P}_0$ holds $[\![\sigma]\!]_0^B$ and $[\![\mu]\!]_0^A$, and later receives two messages $m_i = (i \oplus [\![\sigma]\!]_0^B) \cdot [\![\mu]\!]_0^A - r, i \in \{0,1\}$ to $\mathcal{P}_1$. We need to prove that $m_i, i \in \{0,1\}$ are uniformly random in $\mathcal{P}_0$'s view. Since r is uniformly random in $\mathcal{P}_0$'s view, which implies that $m_i, i \in \{0,1\}$ are also uniformly random in $\mathcal{P}_0$'s view since r is independent of other values used in the generation of $m_i, i \in \{0,1\}$ [37]. Therefore, from the security of ASS [13] and FSS [24], the simulator for $\mathcal{P}_0$ in secSearch exists.
- **Simulator for $\mathcal{P}_0$ in** secReveal. At the beginning of secReveal, $\mathcal{P}_0$ has the secret share $[\![\mathbf{I}^\star]\!]_0^A$, and later receives secret shares in secure shuffle (i.e., secShuffle) and secret shares $[\![\hat{\mathbf{I}}^\star]\!]_1^A$ from $\mathcal{P}_1$ to recover $\hat{\mathbf{I}}^\star$. Therefore, from the security of secret shuffle [31] and ASS [13], the simulator for $\mathcal{P}_0$ in secReveal exists. □

We next analyze the security of task assignment in the batch-based setting. We start with defining the corresponding ideal functionality $\mathcal{F}'$, which comprises the following parts:

- **Input.** The requester submits its task batch $\mathcal{T}$ to $\mathcal{F}'$; each worker submits its ID and location to $\mathcal{F}'$.
- **Task assignment.** Upon receiving the task batch $\mathcal{T}$ from the requester, and the ID and location from each worker, $\mathcal{F}'$ performs task assignment as per the greedy method. After that, $\mathcal{F}'$ sends each task content t to its assigned workers.
- **Output.** The assigned workers perform the task $t \in \mathcal{T}$ based on its requirement at the specified location.

Let $\prod'$ denote the protocol for security protocol realizing the ideal functionality $\mathcal{F}'$. The security of $\prod$ can be formally defined as in Definition 2.

Theorem 2. *In the semi-honest and non-colluding threat model and given the security of ASS and FSS, PPTA securely realizes the ideal functionality $\mathcal{F}'$ according to Definition 2.*

Proof. Recall that our protocol for secure task assignment in the batch-based setting (i.e., Algorithm 5) consists of three subroutines: i) secure available workers labeling (i.e., Algorithm 6 secLabel$^\diamond$); ii) secure k-nearest available workers search (i.e., Algorithm 3 secSearch); iii) secure k-nearest workers available revealing (i.e., Algorithm 4 secReveal). Each subroutine in Algorithm 5 is invoked in order as per the processing pipeline and their inputs are random secret shares. Therefore, if the simulator for each subroutine exists, PPTA is secure [34–36]. It is noted that the simulators for $\mathcal{P}_{\{0,1\}}$ in secSearch and secReveal exist based on Proof 1. In addition, the only difference between secLabel$^\diamond$ and secLabel is line 5. Specifically, line 5 in secLabel$^\diamond$ is $[\![(d_i - r^2 \le 0)]\!]^B \otimes (c_i < b)$ and line 5 in secLabel is $[\![(d_i - r^2 \le 0)]\!]^B$, namely, line 5 in secLabel$^\diamond$ has one more basic multiplication than that in secLabel. Since the simulator for $\mathcal{P}_{\{0,1\}}$ in secLabel exists based on Proof 1, the simulator for $\mathcal{P}_{\{0,1\}}$ in secLabel$^\diamond$ also exists. □

8. Experiments

8.1. Setup

Implementation. We implement the protocols in PPTA with C++. Our implementation is comprised of about 1800 lines of C++ code (excluding the code of libraries). We also implement a test module with another 400 lines of C++ code. All experiments are conducted on a workstation with 16 cores and 64 GB RAM running on a 3.8 GHz Intel i7-10700k CPU. The server-side communication on the workstation is emulated by the loopback filesystem, where the delay is fixed at 2 ms. For the primitive of ASS, we set the ring size to 2^{64}, which can be implemented directly with unsigned long data types in C++. This setting allows to substitute modular addition (multiplication) with regular addition (multiplication), which greatly improves the computation efficiency [38]. We use the method proposed in [38] to handle real numbers in secret sharing domain. Specifically, for a private real number x, we encode it by fixed-point representation with t bits of precision: $\lfloor x \cdot 2^t \rceil$. We use $t = 8$ in our experiments. The security parameter λ of DCF is set to 128.

Dataset. We conduct experiments on a real-world dataset (named as Yelp) from Kaggle[1], which is also used in prior work [5]. Yelp contains 150346 samples across 8 metropolitan areas in the USA and Canada. Each sample in Yelp contains a location in GIS coordinates (latitude and longitude) and a user ID. We randomly choose some locations as task locations and worker locations respectively. Additionally, we convert these sampled locations to Cartesian coordinates through a

[1] https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset

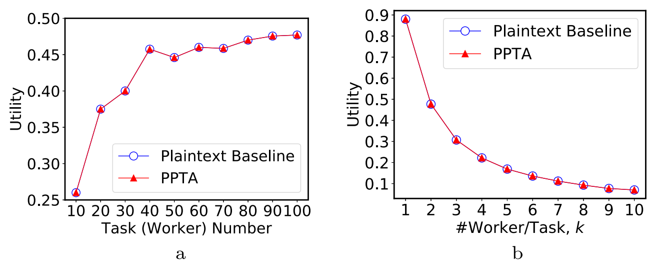


Fig. 3. Utility in the online setting for varying (a): $|\mathcal{T}|$ ($|\mathcal{W}|$); (b): k.

commonly used technique: Miller cylindrical projection [28] and shift the lower-left boundary point to $(0, 0)$.

Plaintext baselines. We compare the effectiveness of PPTA with different plaintext baselines. Specifically, in the online setting, we use the plaintext task assignment method as the baseline method, which performs the task assignment without considering privacy protection and can directly access the plaintext locations of tasks and workers. In the batch-based setting, we use the plaintext Max-Flow algorithm as the baseline method, which first constructs the flow network based on the plaintext locations of tasks and workers, and then performs fold-fulkerson algorithm [39] to obtain the optimal matches.

Metrics. For the online setting, we compare the utility, average distance of task assignment in our PPTA with the plaintext baseline and evaluate the running time of PPTA. The utility is measured by the percentage of tasks successfully assigned to k workers. Average distance is measured by the average travel distance between all assigned task-worker pairs. Running time is measured by the average assignment time for a single task through the secure task assignment protocol, i.e., Algorithm 1. In all experiments for the online setting, the numbers of workers and tasks are both set to 100, the search range r is fixed to 2000 m and k is set to 2 by default unless otherwise specified. For the batch-based setting, we compare the number of matched pairs, average distance of task assignment in our PPTA with the plaintext baseline and evaluate the running time of PPTA. In all experiments for the batch-based setting, the numbers of workers and tasks are both set to 50, the search range r is fixed to 2000 m, k is set to 2, b (i.e., the maximum number of tasks a worker can handle) is set to 2 by default unless otherwise specified. It is worth nothing that different from [7], there are no wrong assignments in PPTA due to the correctness of the adopted secure computation techniques. Therefore, we do not evaluate the error rate in our experiments. The reported results are the average over 10 runs.

8.2. Evaluation on ppta in the online setting

8.2.1. Utility evaluation

We first evaluate the utility of secure task assignment protocol in Algorithm 1 by comparing the percentage of tasks successfully assigned to k workers in PPTA and the plaintext baseline, by varying the number of tasks (workers), i.e., $|\mathcal{T}|$ ($|\mathcal{W}|$), and the required number of workers assigned to each task (i.e., k). The results are summarized in Figs. 3(a) and 3(b), where Fig. 3(a) shows the utility with the increase of data size and Fig. 3(b) shows the utility with the increase of k. It is observed that PPTA can achieve exactly the same utility as the plaintext baseline. In addition, the utility in both methods increases as the data size grows, because more workers mean higher density of workers, and thus higher success rate of task assignment.

8.2.2. Average distance evaluation

We now evaluate the average travel distance between all task-worker pairs generated by PPTA. Figs. 4(a) and 4(b) compare the average travel distance between all task-worker pairs generated by PPTA and plaintext baseline, for varying $|\mathcal{T}|$ ($|\mathcal{W}|$) and k, respectively.

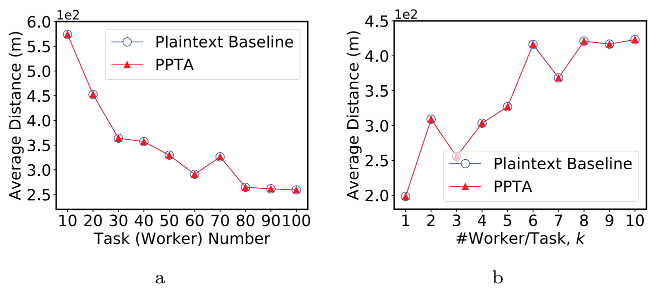


Fig. 4. Average distance in the online setting for varying (a): $|\mathcal{T}|$ ($|\mathcal{W}|$); (b): k.

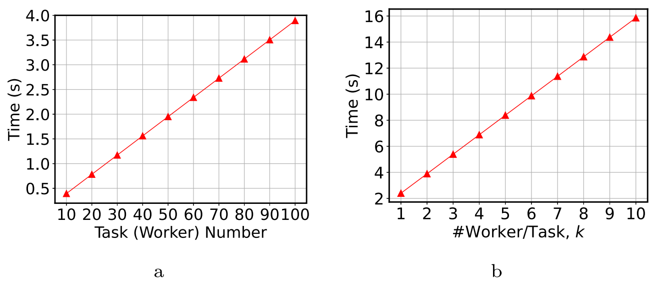


Fig. 5. Running time in the online setting for varying (a): $|\mathcal{T}|$ ($|\mathcal{W}|$); (b): k.

It is observed that the average travel distance in PPTA is similar to that in the plaintext baseline and they exhibit consistent behavior. In addition, in both methods, the average travel distance decreases as the data size increases since the more workers, the higher the probability of finding workers closer to each task. However, the average travel distance will increase as k increases. The reason is that the more workers are required for each task, the higher the probability that the task will be assigned to workers farther away, which will lead to longer average travel distance.

8.2.3. Running time evaluation

We then evaluate the running time of the secure task assignment protocol in Algorithm 1 under the online setting. Figs. 5(a) and 5(b) summarize the running time of PPTA, for varying the task (worker) number $|\mathcal{T}|$ ($|\mathcal{W}|$) and k, respectively. The reported running time includes both the computation time and communication time. It is observed that the running time is linear to $|\mathcal{T}|$ ($|\mathcal{W}|$) and k. Specifically, as $|\mathcal{T}|$ ($|\mathcal{W}|$) increases from 10 to 100, the running time of PPTA increases from 0.39 s to 3.68 s. As k increases from 1 to 10, the running time of PPTA increases from 2.14 s to 15.86 s. It is noted that the number of DCF keys at each SC service provider consumed in the evaluation for the online setting is 300–110000.

8.3. Evaluation on ppta in the batch-based setting

8.3.1. Evaluation on number of matched pairs

We first compare the secure task assignment protocol in the batch-based setting (i.e., Algorithm 5) with the plaintext baseline (i.e., the Max-Flow method) on the number of matched pairs. Fig. 6(a), Figs. 6(b) and 6(c) summarize the number of matched pairs in PPTA and the Max-Flow method, for varying $|\mathcal{T}|$ ($|\mathcal{W}|$), k, and b. The results show that no matter how $|\mathcal{T}|$ ($|\mathcal{W}|$), k and b change, the difference in the number of matched pairs between the two methods does not exceed 0.5, which means that the task assignment generated by PPTA is similar to the optimal task assignment generated by the maximum flow method. In addition, it is observed that as k and b increase, the number of matched pairs of both methods increases. The reason is that the increase of k means that each task requires more workers, and thus more matched pairs can be found. On the other hand, the increase in b is equivalent

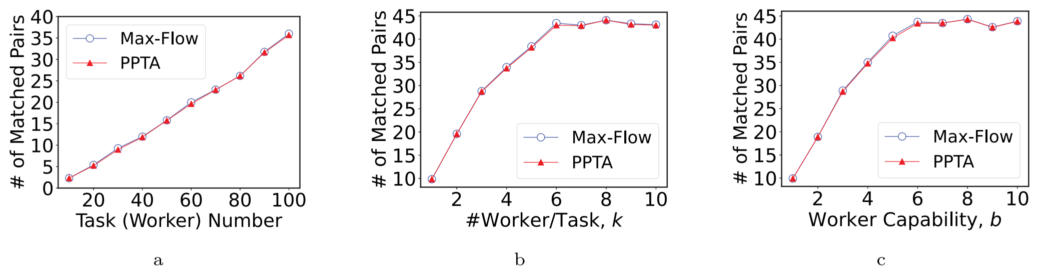


Fig. 6. The number of matched pairs in the batch-based setting for varying (a): $|\mathcal{T}|$ ($|\mathcal{W}|$); (b): k; (c): b.

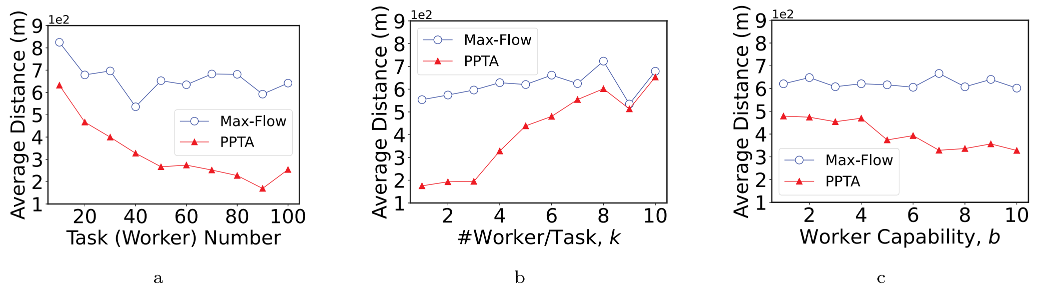


Fig. 7. Average distance in the batch-based setting for varying (a): $|\mathcal{T}|$ ($|\mathcal{W}|$); (b): k; (c): b.

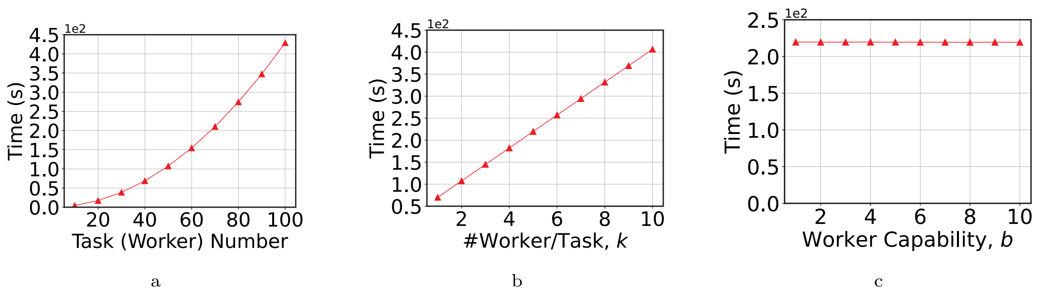


Fig. 8. Running time in the batch-based setting for varying (a): $|\mathcal{T}|$ ($|\mathcal{W}|$); (b): k; (c): b.

to the increase in the number of candidate workers for each task, and thus the number of matched pairs also increases.

8.3.2. Average distance evaluation

The average distance is also an important metric in task assignment under the batch-based setting. Fig. 7(a), Figs. 7(b) and 7(c) show the average travel distance of task assignment generated by PPTA and the Max-Flow method, for varying $|\mathcal{T}|$ ($|\mathcal{W}|$), k and b. It is shown that although two methods can achieve almost the same # of matched pairs, PPTA outperforms Max-Flow in the average travel distance. The reason is that the strategy of PPTA ensures that the closer workers are given much priority during task assignment, while the Max-Flow method gives the equal considerations to all workers in the search range. We analyze the result in each figure in more detail as follows.

Fig. 7(a) shows the average distance for varying $|\mathcal{T}|$ ($|\mathcal{W}|$). As the data size increases from 10 to 100, the average distance between all task-worker pairs generated by PPTA decreases from 632.4 m to 254.3 m, while that of the Max-Flow method only decreases from 824.9 m to 641.8 m. The average distance in both methods decreases as $|\mathcal{T}|$ ($|\mathcal{W}|$) increases, due to the worker density increases. However, the average distance between all task-worker pairs generated by Max-Flow is more stable than PPTA as the increase of data size. The reason is that PPTA will always choose the possible nearest worker. Therefore, when the worker density increases, the average distance decreases sharply. Since the Max-Flow method considers all workers with equal probability, the average distance between all task-worker pairs generated by it decreases gently. Similarly, Fig. 7(b) illustrates the average distance for varying k, where the average distance in both methods increases as k increases. The reason is that the increase of k will also increase the number of matched pairs, and thus more far apart worker-task pairs will be produced. In addition, as k increases, the average distance between all task-worker pairs generated by PPTA will approach that of Max-Flow. The reason is that under a small k, PPTA can choose workers who are close to tasks. However, as k increases, the number of tasks assigned to workers closer to tasks will reach the maximum (i.e., b), and thus PPTA has to choose workers from the ones who are farther away from tasks. Finally, Fig. 7(c) illustrates the average distance for varying b. It is observed that the average distance in Max-Flow remains in the range 600 m–670 m as b increases, while the average distance of PPTA decreases as b increases. The reason is that although there are more candidate workers to choose, the Max-Flow method has to choose the workers with an equal probability. In contrast, PPTA can choose the closer worker, and thus achieves the smaller average distance.

8.3.3. Running time evaluation

We now evaluate the running time of secure task assignment protocol under the batch-based setting. Specifically, Fig. 8(a), Fig. 8(b), and Fig. 8(c) show the running time, for varying $|\mathcal{T}|$ ($|\mathcal{W}|$), k, and b, respectively. It is observed that the running time is correlated with the $|\mathcal{T}|$ ($|\mathcal{W}|$) and k, but independent with b. The reason is that b is the maximum number of tasks a worker can handle, which is independent with the time complexity of secure task assignment protocol under the batch-based setting, i.e., Algorithm 5. It is noted that the number of DCF keys at each SC service provider consumed in the evaluation for the batch-based setting is 300–30000.

8.4. Comparison with the state-of-the-art prior works

A fair comparison between PPTA and the state-of-the-art prior works [5,7] does not exist due to their limitations analyzed in Section 2. Specifically, both prior works [5,7] do not offer strong protection for the distances between tasks and workers. In addition, while providing strong security guarantees, PPTA provides more accurate task assignment than [7], and does not require the task requester to get involved in the online phase in contrast with [5]. Specifically, PPTA assigns tasks based on the accurate distances between tasks and workers, while the scheme in [7] assigns tasks based on the approximate distances. Besides, since the scheme in [5] produces a candidate worker set containing dummy workers (due to the use of anonymization), the requester needs to further locally filter the set to obtain the final assignment result. In contrast, PPTA does not require the requester to be involved in the whole process of secure task assignment.

9. Conclusion

In this paper, we design, implement, and evaluate PPTA, a new system framework supporting location privacy-preserving and flexible task assignment in SC, with stronger security and richer functionalities over prior art. PPTA flexibly supports the online task assignment setting as well as the batch-based task assignment setting, and only makes use of lightweight cryptographic techniques (like FSS, ASS, and secure shuffle). We provide formal security analysis for PPTA following the standard simulation-based paradigm. Extensive experiments over a real-world dataset demonstrate that while providing strong security guarantees for SC services, PPTA can achieve nearly the same effectiveness of task assignment as the plaintext baselines with promising performance.

CRediT authorship contribution statement

Menglun Zhou: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. **Yifeng Zheng:** Conceptualization, Methodology, Supervision, Funding acquisition, Writing – review and editing. **Songlei Wang:** Conceptualization, Methodology, Writing – review and editing. **Zhongyun Hua:** Conceptualization, Validation, Writing – review and editing. **Hejiao Huang:** Validation, Writing – review and editing. **Yansong Gao:** Validation, Writing – review and editing. **Xiaohua Jia:** Validation, Writing – review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grants No. 2021A1515110027 and No. 2021A1515011406, and in part by the Shenzhen Science and Technology Program under Grants No. RCBS20210609103056041, No JCYJ20220531095416037, No. GXWD20220817124827001, and No. JCYJ20210324132406016, in part by the National Natural Science Foundation of China under Grant No. 62002167, and in part by the Natural Science Foundation of JiangSu, China under Grant No. BK20200461.

References

[1] Y. Tong, L. Chen, C. Shahabi, Spatial crowdsourcing: Challenges, techniques, and applications, Proc. VLDB Endow. 10 (12) (2017) 1988–1991.
[2] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, C. Shahabi, Spatial crowdsourcing: a survey, VLDB J. 29 (1) (2020) 217–250.
[3] A. Artikis, M. Weidlich, F. Schnitzler, I. Boutsis, T. Liebig, N. Piatkowski, C. Bockermann, K. Morik, V. Kalogeraki, J. Marecek, A. Gal, S. Mannor, D. Gunopulos, D. Kinane, Heterogeneous stream processing and crowdsourcing for urban traffic management, in: Proc. of EDBT, 2014.
[4] L. Pournajaf, L. Xiong, V.S. Sunderam, X. Xu, STAC: spatial task assignment for crowd sensing with cloaked participant locations, in: Proc. of ACM SIGSPATIAL, 2015.
[5] H. Li, Q. Song, G. Li, Q. Li, R. Wang, GPSC: A grid-based privacy-reserving framework for online spatial crowdsourcing, IEEE Trans. Knowl. Data Eng. (2021) 1.
[6] Q. Tao, Y. Tong, Z. Zhou, Y. Shi, L. Chen, K. Xu, Differentially private online task assignment in spatial crowdsourcing: A tree-based approach, in: Proc. of IEEE ICDE, 2020.
[7] D. Yuan, Q. Li, G. Li, Q. Wang, K. Ren, PriRadar: A privacy-preserving framework for spatial crowdsourcing, IEEE Trans. Inf. For. Secur. 15 (2020) 299–314.
[8] M. Li, J. Wang, L. Zheng, H. Wu, P. Cheng, L. Chen, X. Lin, Privacy-preserving batch-based task assignment in spatial crowdsourcing with untrusted server, in: Proc. of ACM CIKM, 2021.
[9] D. Shi, Y. Tong, Z. Zhou, B. Song, W. Lv, Q. Yang, Learning to assign: Towards fair task assignment in large-scale ride hailing, in: Proc. of ACM SIGKDD, 2021.
[10] Y. Tong, Y. Chen, Z. Zhou, L. Chen, J. Wang, Q. Yang, J. Ye, W. Lv, The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms, in: Proc. of ACM SIGKDD, 2017.
[11] X. Li, D.W. Goldberg, Toward a mobile crowdsensing system for road surface assessment, Comput. Environ. Urban Syst. 69 (2018) 51–62.
[12] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, B. Nath, Real-time air quality monitoring through mobile sensing in metropolitan areas, in: Proc. of the ACM SIGKDD International Workshop on Urban Computing, 2013.
[13] D. Demmler, T. Schneider, M. Zohner, ABY - A framework for efficient mixed-protocol secure two-party computation, in: Proc. of NDSS, 2015.
[14] N. Agrawal, A.S. Shamsabadi, M.J. Kusner, A. Gascón, QUOTIENT: two-party secure neural network training and prediction, in: Proc. of ACM CCS, 2019.
[15] W. Chen, R.A. Popa, Metal: A metadata-hiding file-sharing system, in: Proc. of NDSS, 2020.
[16] X. Ding, Z. Wang, P. Zhou, K.-K.R. Choo, H. Jin, Efficient and privacy-preserving multi-party skyline queries over encrypted data, IEEE Trans. Inf. For. Secur. 16 (2021) 4589–4604.
[17] M.S. Blog, Next steps in privacy-preserving Telemetry with Prio, 2022, online at https://blog.mozilla.org/security/2019/06/06/next-steps-in-privacy-preserving-telemetry-with-prio/. (Accessed 1 Jun 2022).
[18] H. To, C. Shahabi, L. Xiong, Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server, in: Proc. of IEEE ICDE, 2018.
[19] L. Pournajaf, L. Xiong, V.S. Sunderam, S. Goryczka, Spatial task assignment for crowd sensing with cloaked locations, in: Proc. of IEEE MDM, 2014.
[20] M.E. Andrés, N.E. Bordenabe, K. Chatzikokolakis, C. Palamidessi, Geo-indistinguishability: Differential privacy for location-based systems, in: Proc. of ACM CCS, 2013.
[21] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Proc. of ICALP, 2006.
[22] E.M. Kornaropoulos, C. Papamanthou, R. Tamassia, Data recovery on encrypted databases with k-nearest neighbor query leakage, in: Proc. of IEEE S&P, 2019.
[23] E.A. Markatou, F. Falzon, R. Tamassia, W. Schor, Reconstructing with less: Leakage abuse attacks in two dimensions, in: Proc. of ACM CCS, 2021.
[24] E. Boyle, N. Chandran, N. Gilboa, D. Gupta, Y. Ishai, N. Kumar, M. Rathee, Function secret sharing for mixed-mode and fixed-point secure computation, in: Proc. of EUROCRYPT, 2021.
[25] E. Boyle, N. Gilboa, Y. Ishai, Function secret sharing, in: Proc. of EUROCRYPT, 2015.
[26] S. Wang, Y. Zheng, X. Jia, X. Yi, PeGraph: A system for privacy-preserving and efficient search over encrypted social graphs, IEEE Trans. Inf. For. Secur. 17 (2022) 3179–3194.
[27] Y. Zheng, H. Duan, C. Wang, Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing, IEEE Trans. Inf. For. Secur. 13 (10) (2018) 2475–2489.
[28] J.P. Snyder, Flattening the Earth: Two Thousand Years of Map Projections, University of Chicago Press, 1997.
[29] O. Astrachan, Bubble sort: an archaeological algorithmic analysis, ACM Sigcse Bull. 35 (1) (2003) 1–5.
[30] P. Mohassel, P. Rindal, Aby^3: A mixed protocol framework for machine learning, in: Proc. of ACM CCS, 2018.
[31] S. Eskandarian, D. Boneh, Clarion: Anonymous communication from multiparty shuffling protocols, in: Proc. of NDSS, 2022.
[32] D.B. West, et al., Introduction to Graph Theory, Vol. 2, Prentice hall Upper Saddle River, 2001.
[33] Y. Lindell, How to simulate it - a tutorial on the simulation proof technique, in: Electronic Colloquium on Computational Complexity, 2017, p. 112.

[34] R. Canetti, Security and composition of multiparty cryptographic protocols, J. Cryptol. 13 (1) (2000) 143–202.
[35] J. Katz, Y. Lindell, Handling expected polynomial-time strategies in simulation-based security proofs, J. Cryptol. 21 (3) (2008) 303–349.
[36] M. Curran, X. Liang, H. Gupta, O. Pandey, S.R. Das, Procsa: Protecting privacy in crowdsourced spectrum allocation, in: Proc. of ESORICS, 2019.
[37] T. Araki, J. Furukawa, Y. Lindell, A. Nof, K. Ohara, High-throughput semi-honest secure three-party computation with an honest majority, in: Proc. of ACM CCS, 2016.
[38] P. Mohassel, Y. Zhang, SecureML: A system for scalable privacy-preserving machine learning, in: Proc. of IEEE S&P, 2017.
[39] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2022.

Menglun Zhou received the BE degree in computer science and technology from Harbin Institute of Technology, China, in 2021. He is currently working toward the ME degree in the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include cloud computing security and secure multi-party computation.

Yifeng Zheng is an Assistant Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He received the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2019. He worked as a postdoc with the Commonwealth Scientific and Industrial Research Organization, Australia, and with the City University of Hong Kong. His work has appeared in prestigious conferences such as ESORICS, DSN, ACM AsiaCCS, IEEE PERCOM, IEEE INFOCOM, IEEE ICDCS, as well as journals such as IEEE TDSC and IEEE TIFS. His current research interests are focused on security and privacy related to cloud computing, IoT, machine learning, and multimedia.

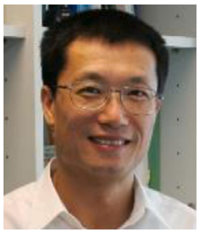
Songlei Wang received the BE degree in internet of things from China University of Petroleum (East China), Qingdao, China, in 2018, the ME degree in computer technology from Harbin Institute of Technology, Shenzhen, China, in 2021. He is currently working toward the PhD degree in the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include cloud computing security and secure machine learning.

Zhongyun Hua received the B.S. degree in software engineering from Chongqing University, Chongqing, China, in 2011, and the M.S. and Ph.D. degrees in software engineering from the University of Macau, Macau, China, in 2013 and 2016, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen, China. He has published more than 50 papers on the subject and receiving more than 2700 citations. His research interests include chaotic systems, image processing, and information security. He serves as an Associate Editor for International Journal of Bifurcation and Chaos.

Hejiao Huang received the PhD degree in computer science from City University of Hong Kong in 2004. She is currently a professor with Harbin Institute of Technology, Shenzhen, China, and previously was an invited professor with INRIA, France. Her research interests include cloud computing, trustworthy computing, formal methods for system design and wireless networks.

Yansong Gao received his Ph.D. degree from the University of Adelaide, Australia, in 2017. He was then with Data61, CSIRO, Sydney, Australia as a Postdoc Research Fellow. He is currently with Nanjing University of Science and Technology, China, as an associate professor.

His current research interests are AI security and privacy, hardware security, and system security.


Xiaohua Jia received his BSc (1984) and MEng (1987) from University of Science and Technology of China, and DSc (1991) in Information Science from University of Tokyo. He is currently Chair Professor with Dept of Computer Science at City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks and mobile computing. Prof. Jia is an editor of IEEE Internet of Things, IEEE Transactions on Parallel and Distributed Systems (2006–2009), Wireless Networks, Journal of World Wide Web, Journal of Combinatorial Optimization, etc. He is the General Chair of ACM MobiHoc 2008, TPC Co Chair of IEEE GlobeCom 2010 Ad Hoc and Sensor Networking Symposium, Area-Chair of IEEE INFOCOM 2010 and 2015. He is an IEEE Fellow.