

# Toward Privacy-Preserving Task Assignment for Fully Distributed Spatial Crowdsourcing

Mingzhe Li<sup>1</sup>, *Student Member, IEEE*, Jingrou Wu, *Student Member, IEEE*,  
Wei Wang<sup>2</sup>, *Member, IEEE*, and Jin Zhang<sup>3</sup>, *Member, IEEE*

**Abstract**—With the proliferation of human-carried mobile devices, spatial crowdsourcing has emerged as a transformative system, where requesters outsource their spatiotemporal tasks to a set of workers who are willing to perform the tasks at the specified locations. However, in order to make efficient assignments, the existing spatial crowdsourcing system usually requires workers and/or tasks to expose their locations, which raises a significant concern of compromising location privacy. In addition, traditional spatial crowdsourcing systems employ a centralized server to manage the information of workers and tasks. Such a centralized design does not scale to a large number of workers/tasks, making the server easily a bottleneck. In this article, we present an online framework for assigning tasks to workers without compromising the location privacy in a *fully distributed manner*. Our system protects the location privacy of both workers and tasks through *homomorphic encryption*. We further propose a novel *wait-and-decide* mechanism and a *proportional-backoff* mechanism to increase the number of assigned tasks. Extensive experiments on real-world data sets illustrate that our proposed system achieves a large number of task assignments in an efficient and privacy-preserving manner.

**Index Terms**—Distributed system, privacy preserving, spatial crowdsourcing, task assignment.

## I. INTRODUCTION

**T**HANKS to the proliferation of mobile devices and the advancement in sensor technologies, data collection

Manuscript received September 30, 2020; revised February 24, 2021; accepted March 17, 2021. Date of publication March 30, 2021; date of current version September 6, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61701216; in part by the Shenzhen Science, Technology and Innovation Commission Basic Research Project under Grant JCYJ20180507181527806; in part by the Guangdong Provincial Key Laboratory under Grant 2020B121201001; in part by the Guangdong Innovative and Entrepreneurial Research Team Program under Grant 2016ZT06G587; and in part by the Shenzhen Sci-Tech Fund under Grant KYTDPT20181011104007. (*Corresponding authors: Jin Zhang; Wei Wang.*)

Mingzhe Li is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China, and also with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: mlibn@cse.ust.hk).

Jingrou Wu is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: 11510281@mail.sustech.edu.cn).

Wei Wang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: weiwa@cse.ust.hk).

Jin Zhang is with the Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: zhangj4@sustech.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3069462

and sharing using smartphones have become commonplace for mobile users. Exploiting the wisdom and mobility of a large number of mobile users, spatial crowdsourcing [16] has emerged as a new mechanism for efficient and scalable data collection. In traditional spatial crowdsourcing frameworks, requesters register at a centralized server to publish tasks with spatial or temporal information. The server then acts as a broker to manage the tasks and assigns them to different workers. A worker, upon accepting a task, needs to physically travel to the specified location and perform that task. Spatial crowdsourcing has been employed in a wide spectrum of applications, such as traffic management, weather monitoring, environmental sensing, crises response, smart cities, and journalism (e.g., [13], [22], and [28]).

In spatial crowdsourcing, location information of tasks and workers critically affects the system performance. In order to make optimal task assignment (e.g., maximizing the number of assigned tasks, or minimizing the travel cost), existing spatial crowdsourcing systems (e.g., [16] and [27]) usually require workers (requesters) to reveal their exact locations (task locations) to the server. However, the server in spatial crowdsourcing may not be fully trustable in practice, not to mention the workers and requesters. Thus, the revelation of the locations of either tasks or workers raises serious privacy concerns. In light of this problem, there is a pressing need to build a secure spatial crowdsourcing system that protects the individual *location privacy* for both requesters and workers.

Many recent research have focused on making efficient task assignment while protecting location privacy. Most of the existing works achieve location privacy preservation based on location obfuscation or perturbation techniques, such as *k*-anonymity [12] and differential privacy [3], where *k*-anonymity allows a user to hide its location among  $k - 1$  other users, while differential privacy preserves a user's location privacy by empowering the user to generate and report a perturbed location according to a certain noise function. However, these techniques have their own drawbacks. The drawback of *k*-anonymity is that it needs to group entities (e.g., workers and requesters) together, making it a poor fit in spatial crowdsourcing systems where no trust relationship between requesters and workers can be assumed in practice. A well-known problem for differential privacy is the unnecessary revelation of task locations to unassigned workers [26] due to the location perturbation. Moreover, the perturbed locations caused by differential privacy will also compromise the performance of task assignment.

The existing studies assume a *centralized* model for privacy protection in which a server oversees a global picture and makes task assignment for requesters [15], [25], [29]. A main issue for such a centralized model is that many centralized crowdsourcing systems are not flexible and cannot provide a rapid task assignment [23]. For example, only less than 15% of tasks can be finished in 1 h in Amazon Mechanical Turk [14]. Another economic concern is that the server who acts as a broker is expected to charge commissions to the requesters, which, in turn, reduces the monetary income of workers [17].

Recent years, *fully distributed* spatial crowdsourcing has become a more promising framework for efficient task assignment by exploiting nearby human intelligence [5], [20], [23], [24]. The term fully distributed is in the sense that task assignment is obtained through bidirectional selection between nearby mobile workers and requesters without any involvement of a centralized server. The fully distributed architecture can overcome the drawbacks existed in the centralized spatial crowdsourcing systems, yet none of the previous distributed systems considers the location privacy protection problem during the task assignment procedure.

Motivated by these problems, we propose a framework named privacy-preserving task assignment (PriTA), which enables a group of requesters and workers to achieve *online* task assignment in an effective and *fully distributed* manner, without compromising their location privacy. Despite the great potential, developing a realistic PriTA system remains challenging. One key issue is how to get precise distances between workers and requesters to guarantee the task assignment performance while preserving the location privacy of workers/tasks. Another challenge is how to increase the number of assigned tasks under such a fully distributed scenario.

To address the first challenge, we propose to use encryption to protect location privacy as the precise location information is concealed behind the ciphertexts after encryption and will not be revealed to any party. Specifically, *homomorphic encryption* [11] is proposed to be exploited in which ciphertexts can be directly used in various meaningful operations (e.g., addition and multiplication). Hence, a participant (i.e., worker and requester) can calculate distances and other distance-related information based on the encrypted location information. As a result, the homomorphic encryption scheme has the potential to yield not only robust location privacy protection but also high-quality task assignment since no obfuscation or perturbation of the location is needed.

For the second issue, we propose a *wait-and-decide* mechanism and a *proportional-backoff* mechanism in order to make efficient task assignment based on encrypted messages in a fully distributed setting, where the efficiency of the task assignment in our system is measured by the number of assigned tasks. Specifically, none of the requesters or workers could hold a global view like a server to manage the task assignment process in the distributed spatial crowdsourcing scenario. Therefore, we first design a wait-and-decide mechanism to help workers make better decisions in selecting tasks. This regard is achieved by allowing workers to wait for a period of time. A proportional-backoff mechanism is

then proposed to support requesters to choose closer workers according to the time delay of the received messages, where the backoff time is proportional to the distances between workers and tasks.

We summarize our contributions as follows.

- 1) We propose an online spatial crowdsourcing framework PriTA that, for the first time, performs task assignment between workers and requesters in a fully distributed manner, without disclosing of their accurate location information.
- 2) We propose to use homomorphic encryption to protect the location privacy for both tasks and workers. We design a distributed mechanism to achieve efficient task assignment based on exchanges of encrypted messages between requesters and workers.
- 3) We conduct extensive experiments driven by real-world data sets. The experimental results show that our proposed framework achieves *near-optimal* task assignment, with low computation and communication overhead on both workers and requesters.

The remainder of this article is organized as follows. Section II introduces the background, system model, and design objectives. Section III presents our system procedure in detail. In Section IV, we make a thorough privacy preservation analysis. Performance evaluation is given in Section V, followed by a survey of related work in Section VI. We conclude this article in Section VII.

## II. BACKGROUND AND SYSTEM MODEL

### A. Spatial Crowdsourcing Model

In traditional spatial crowdsourcing systems, there are usually three main entities: 1) the *requesters*; 2) the *workers*; and 3) the *platform (server)*. The requesters want to outsource some spatiotemporal tasks to workers. The workers are willing to accept several tasks and are able to finish them within their capability. The platform usually collects the task information and assigns them to appropriate workers. The workers who receive and accept the task assignment will move to the task locations to perform the tasks. In return, the workers can earn a reward for finished tasks.

With the development of device-to-device communications and self-organizing network, the spatial crowdsourcing system without platform is emerging and gaining more and more attention [5], [20], [23], [24]. In such systems, there is no centralized platform. Instead, all the requesters and workers communicate with each other directly, and all the decisions are made in a distributed way. In this article, we will focus on the task assignment on such *fully distributed* spatial crowdsourcing systems.

Fig. 1 gives an architecture overview of our system model. There are  $M$  requesters  $\{r_1, r_2, \dots, r_i, \dots, r_M\}$ , and  $N$  workers  $\{w_1, w_2, \dots, w_j, \dots, w_N\}$ . The requester has some tasks needed to be completed in its location. In this article, we make the assumption that requester  $r_i$  and his task  $t_i$  are in the same place and tasks are generated asynchronously. For task  $t_i$ , it has a maximum number of workers the requester requires, which is defined as the capacity of the task  $c_{t_i}$ . The farthest distance a

TABLE I  
MAJOR NOTATIONS

Notation	Description
$M, r_i, t_i$	Set of all requesters, one requester, one task
$N, w_j$	Set of all workers, one worker
$c_{t_i}, c_{w_j}$	Task capacity, worker capacity
$R_j$	Worker's moving range
$T_w, T_{busy}$	Worker's waiting time, worker's busy time
$R_t$	Transmission range of a message
$epk_{r_i}, dsk_{r_i}$	Homomorphic public key of a requester, homomorphic private key of a requester
$pk_{w_j}, sk_{w_j}$	Public encryption key of a worker, private decryption key of a worker
$E_{pk_{w_j}}(\cdot), E_{epk_{r_i}}(\cdot)$	Encryption function using a worker's public key, encryption function using a requester's public key
$D_{sk_{w_j}}(\cdot), D_{dsk_{r_i}}(\cdot)$	Decryption function using a worker's private key, decryption function using a requester's private key
$l_{t_i}$	Location of a task
$d_{ij}, z_{ij}$	Distance between $t_i$ and $w_j$ , distance comparison result between $t_i$ and $w_j$
$t_{backoff}, T_{unit}$	Backoff time, unit backoff time

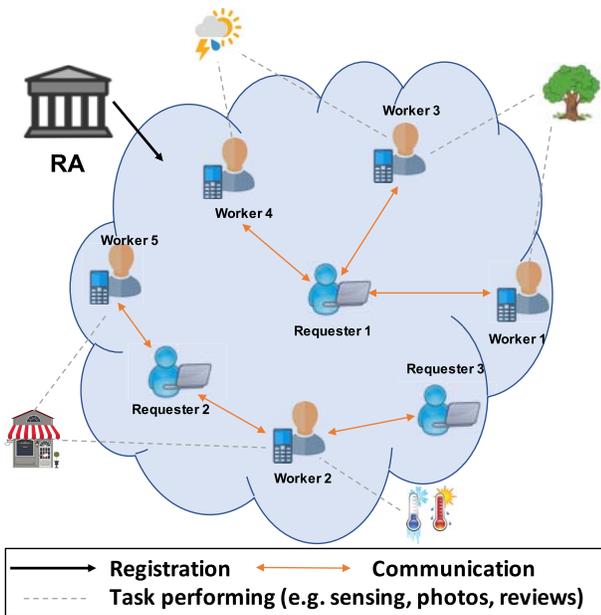


Fig. 1. System architecture.

worker  $w_j$  can go is its moving range  $R_j$ . The worker can only accept and finish the tasks whose distance from the worker's location is smaller than  $R_j$ . There is also a predefined capacity  $c_{w_j}$  for the worker  $w_j$ , which indicates the maximum number of tasks the worker can finish during a specific busy time  $T_{busy}$ . To prevent malicious node from entering the system, we assume all the requesters and workers have been authenticated by a registration authority (RA) upon joining the system. The RA is only responsible for authentication but plays no role in online task assignment.

As the system is fully distributed without a platform, the task assignment between the tasks and the workers can only be achieved by the negotiation between requesters and workers via direct device-to-device communication (e.g., WiFi, LoRa, etc.). Assume that the transmission range for the users (workers and requesters) is  $R_t$ , where a message cannot be heard by a user who is larger than  $R_t$  far away from the sender of the message. Meanwhile, the location privacy of the requesters and workers must be protected during the negotiation procedure.

The requester does not want its task location to be disclosed before the task has been successfully assigned. The worker does not want to expose his exact location to others either. Table I summarizes the major notations used in our article.

### B. Threat Model

In our model, we mainly focus on the privacy threats from *internal* requesters or workers, while security attacks from external malicious attackers are not considered. Specifically, the internal requesters and workers are those who pass the RA's authentication. They perform the fully distributed spatial crowdsourcing together in our system. The other entities outside the system are called external entities. Following the prior work [4], [21], [26], we make the following assumptions that are widely accepted in the literature. First, we assume that requesters and workers are *honest but curious*. Meaning, they all follow the specified protocols strictly, but are eager to dig out the private information of the others based on the available information. Specifically, a requester is eager to learn the location information of workers and/or the other requesters. However, it will always obey the protocol rules to participate the task assignment properly. For example, it cannot create multiple dummy tasks at different locations at the same time as this deviates the protocol requirements. For a worker, it may also want to know the locations of published tasks and the other workers without breaking the protocol requirements. For instance, it cannot create multiple fake identities, as the protocol only allows each requester/worker to have one legal identity. In addition, we assume there is *no collusion* between requesters and workers, which is also a widely accepted assumption. This means each requester and worker cannot cooperate with each other to learn location information about others. For example, the worker will not send the unit backoff time  $T_{unit}$  to any requester. We also assume trustable RA that cannot be breached by any adversary.

### C. Design Objectives

We aim to achieve four main objectives in this article.

- 1) Our system should protect the location privacy of each task safely before the task is assigned. Only those

workers who agree to do the task can receive the accurate location of that task.

- 2) Our system should protect each worker's location in every single step. The precise location cannot be learned by any other parties throughout its participation.
- 3) Our system should maximize the number of the assigned tasks in a fully distributed manner.
- 4) Low computation and communication cost are required for workers and requesters for fast task assignment.

#### D. Homomorphic Encryption

To protect the location privacy for both tasks and workers in a distributed setting, we propose to use homomorphic encryption, a public-key encryption scheme supporting various operations over the encrypted data, to hide the sensitive location information behind the ciphertexts.

Due to the reason that fully homomorphic encryption is inefficient in computation, somewhat homomorphic encryption (SHE) was proposed as an alternative, in which only a limited number of addition and multiplication operations are supported. In this article, we propose to use Fan-Vercauteren (FV) homomorphic encryption and only use the SHE of [10]. FV homomorphic encryption is built upon the hardness of the ring-learning-with-errors (RLWEs) problem. Given a homomorphic public key  $epk$ , the message  $m$  after encryption is denoted by  $E_{epk}(m)$ , where  $E_{epk}(\cdot)$  denotes the encryption function. In order to decrypt the ciphertext  $E_{epk}(m)$ , we need a decryption function  $D_{dsk}$  with a homomorphic private key  $dsk$ . The message  $m$ , thus, can be decrypted by  $D_{dsk}(E_{epk}(m))$ . We summarize some properties of the FV homomorphic encryption that will be used in the article.

*Homomorphic Addition and Subtraction:* The homomorphic addition/subtraction operation in the ciphertext achieves the same result as the encrypted data of the addition/subtraction of two plaintext. Formally, we have

$$E_{epk}(m_1) \oplus E_{epk}(m_2) = E_{epk}(m_1 + m_2) \quad (1)$$

$$E_{epk}(m_1) \ominus E_{epk}(m_2) = E_{epk}(m_1 - m_2) \quad (2)$$

where  $\oplus$  and  $\ominus$  are the homomorphic addition and subtraction, respectively.

*Homomorphic Multiplication:* Homomorphic multiplication includes two steps: 1) ciphertext multiplication and 2) relinearization. In ciphertext multiplication, we simply multiply two ciphertexts regardless of the influence of growing errors, which are introduced by multiplication. The result for ciphertext multiplication is denoted as  $\mathbf{c}$ . Then, in relinearization, we use evaluate keys  $\mathbf{evk}$  to eliminate the interference of errors to obtain the actual ciphertext. Let  $E_{epk}^{evk}(\cdot)$  be the relinearization function to correct the ciphertext multiplication result. The homomorphic multiplication can be denoted as

$$\begin{aligned} E_{epk}(m_1) \otimes E_{epk}(m_2) &= E_{epk}^{evk}(\mathbf{c}) \\ &= E_{epk}(m_1 \times m_2) \end{aligned} \quad (3)$$

where  $\otimes$  represents the homomorphic multiplication.

In summary, homomorphic encryption allows computations to be done on encrypted data, without requiring access to a decryption key. Leveraging this property, we are able to

design protocols that allow a user to calculate the distance between two entities on the encrypted location messages, without disclosing the exact original location to neighboring users.

### III. SYSTEM DESIGN

In this section, we present our framework, called PriTA. We start with a brief overview of PriTA, followed by a deep dive into its design details.

#### A. Overview

PriTA aims at achieving location PriTA in a fully distributed spatial crowdsourcing scenario while maintaining the efficiency of task assignment.

Our design essentially considers two aspects. The first is how to acquire necessary information based on the encrypted messages. The second is how to make task assignment in an efficient way according to the obtained information. To address the former aspect, each requester will broadcast its encrypted location to nearby workers. Each worker then computes based on homomorphic encryption that whether it is less than its moving range or not for the distances between itself and the tasks, and sends the encrypted results afterward. A requester can then obtain the information of whether a worker can perform its task. As for the second respect, a novel *wait-and-decide* mechanism is employed where workers need to wait to gain higher chances to select more tasks with closer distances. Moreover, a *proportional-backoff* mechanism is adopted to enable the nearer worker to back off for shorter time to assist requesters to choose more suitable workers.

There are five main steps along with an initialization step in our framework. First, each worker/requester needs to register at an RA to get a certificate associated with its unique ID in the *initialization step*. The initialization step is necessary as it reduces the chance that an external malicious attacker joins in the system, meanwhile, it also resists the Sybil attack produced by any worker or requester. During the PriTA protocol, each requester proactively releases its task along with the encrypted location to the neighboring workers when it has a task to be performed (step 1). The requesters and the workers then interact with each other by exchanging encrypted messages (e.g., locations and intermediate results) to obtain location and distance information for task assignment (steps 2 and 3). Then, both requesters and workers wait for a certain period of time according to the wait-and-decide mechanism. In addition, based on the proportional-backoff mechanism, workers with closer distances back off to requesters for shorter time so as to have higher chances to be selected (step 4). At the end of the protocol, requesters confirm the task assignment results to the workers who are willing to perform the tasks (step 5). In addition, to avoid message collision, each worker and requester will do carrier sense before any data transmission.

#### B. System Initialization

We assume that all the requesters and workers have already been authenticated by RA upon joining the system. Each authenticated requester/worker is assigned a unique ID. As

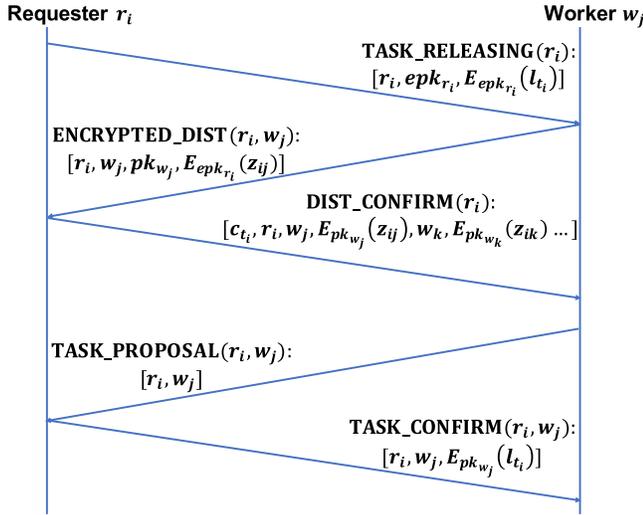


Fig. 2. PriTA protocol design.

a result, no malicious external attacker holds the probability to join in the system, and the Sybil attack is prevented as each requester/worker is only granted to have one identity. To encrypt the messages, each requester and worker should generate a public–private key pair. For requester  $r_i$ , denote its homomorphic public–private key pair by  $(epk_{r_i}, ds_{r_i})$ . For worker  $w_j$ , it also generates a key pair  $(pk_{w_j}, sk_{w_j})$  for encryption and decryption. The reason a worker does not need a homomorphic key pair is that we will not do homomorphic operations on the ciphertexts encrypted by a worker’s public key. In addition, it costs less time when encrypting or decrypting messages using general key pair. All public keys are available to everybody, yet the private keys are kept by the owners only. Moreover, the RA announces a system decided value  $T_{\text{unit}}$  only to each worker. This value will be used later to determine the backoff time in the proportional-backoff mechanism.

### C. Protocol Design

In this section, we describe our privacy-preserving, fully distributed task assignment protocol. The basic flowchart of our system is illustrated in Fig. 2.

*Step 1 (Task Releasing)*: When the requester has a task needed to be done, it broadcasts a TASK\_RELEASEING message to release the task to its neighboring workers. The message contains three domains: 1) the requester’s ID  $r_i$ ; 2) the homomorphic public key  $epk_{r_i}$  of requester  $r_i$ ; and 3) the encrypted location of task  $t_i$  using the requester’s homomorphic public key  $epk_{r_i}$ . That is

$$\text{TASK\_RELEASEING}(r_i) = [r_i, epk_{r_i}, E_{epk_{r_i}}(l_{t_i})].$$

Here,  $l_{t_i} = (x_{t_i}, y_{t_i})$  is the location of task  $t_i$ , where  $x_{t_i}$  and  $y_{t_i}$  are the  $x$ - and  $y$ -coordinates of the task location, respectively. Therefore,  $E_{epk_{r_i}}(l_{t_i})$  is the encrypted location of task  $t_i$ , i.e.,

$$E_{epk_{r_i}}(l_{t_i}) = (E_{epk_{r_i}}(x_{t_i}), E_{epk_{r_i}}(y_{t_i})). \quad (4)$$

*Step 2 (Distance Computing)*: Once a worker receives a TASK\_RELEASEING message from its neighboring requester,

the worker computes the distance between itself and the task in the encryption domain and compares the distance with its moving range. It then packages the comparison result in the ENCRYPTED\_DIST message and sends the message to the requester. Note that the distance comparison is calculated in the encryption domain, so the result is also in the encryption domain.

Each worker  $w_j$  has a predefined moving range  $R_j$ , which is the farthest distance the worker is willing to go. Worker  $w_j$  is only interested in accepting and finishing the tasks that fall into its moving range  $R_j$ . We use the Euclidean distance from worker  $w_j$  to task  $t_i$ , i.e.,

$$d_{ij} = \sqrt{\Delta x_{ij}^2 + \Delta y_{ij}^2} \quad (5)$$

where  $\Delta x_{ij}$  and  $\Delta y_{ij}$  are the difference of  $x$ - and  $y$ -coordinates between task  $t_i$  and worker  $w_j$ , respectively. Let  $z_{ij} \triangleq d_{ij}^2 - R_j^2$  be the distance comparison result between the worker  $w_j$  and task  $t_i$ , where  $z_{ij} > 0$  means the worker cannot reach the task and hence cannot accept it, and  $z_{ij} \leq 0$  means the task falls into the worker’s moving range and can be accepted.

The worker should send the comparison result back to the requester. However, as the task’s location is encrypted, it cannot obtain the plaintext of comparison result  $z_{ij}$ . Instead, we calculate it in the encryption domain and send out the encrypted result. Therefore, in our protocol, the worker sends out the ENCRYPTED\_DIST message, which contains the following four domains: 1) the ID of the requester  $r_i$ ; 2) the ID of the worker  $w_j$ ; 3) the worker’s public key  $epk_{w_j}$ ; and 4)  $E_{epk_{r_i}}(z_{ij})$ , which is the distance comparison result  $z_{ij}$  encrypted by the homomorphic encryption key  $epk_{r_i}$

$$\text{ENCRYPTED\_DIST}(r_i, w_j) = [r_i, w_j, epk_{w_j}, E_{epk_{r_i}}(z_{ij})].$$

The public key of worker  $w_j$  should be sent to the requester because it will be used for data encryption in the subsequent steps. The encrypted distance comparison result is calculated as follows:

$$\begin{aligned} E_{epk_{r_i}}(z_{ij}) &= E_{epk_{r_i}}(\Delta x_{ij}^2 + \Delta y_{ij}^2 - R_j^2) \\ &= E_{epk_{r_i}}(\Delta x_{ij}^2) \oplus E_{epk_{r_i}}(\Delta y_{ij}^2) \ominus E_{epk_{r_i}}(R_j^2). \end{aligned} \quad (6)$$

According to the principles of homomorphic encryption (Section II-D), homomorphic calculation can maintain the addition, subtraction, and multiplication operations in the encryption domain. Therefore, (6) can be computed by bringing (7), (8) into (9), (10) and bringing (9)–(11) into (6)

$$E_{epk_{r_i}}(\Delta x_{ij}) = E_{epk_{r_i}}(x_{w_j}) \ominus E_{epk_{r_i}}(x_{t_i}) \quad (7)$$

$$E_{epk_{r_i}}(\Delta y_{ij}) = E_{epk_{r_i}}(y_{w_j}) \ominus E_{epk_{r_i}}(y_{t_i}) \quad (8)$$

$$E_{epk_{r_i}}(\Delta x_{ij}^2) = E_{epk_{r_i}}(\Delta x_{ij}) \otimes E_{epk_{r_i}}(\Delta x_{ij}) \quad (9)$$

$$E_{epk_{r_i}}(\Delta y_{ij}^2) = E_{epk_{r_i}}(\Delta y_{ij}) \otimes E_{epk_{r_i}}(\Delta y_{ij}) \quad (10)$$

$$E_{epk_{r_i}}(R_j^2) = E_{epk_{r_i}}(R_j) \otimes E_{epk_{r_i}}(R_j). \quad (11)$$

Therefore, the encrypted distance comparison result can be calculated and put into the ENCRYPTED\_DIST message to send to the requester.

*Step 3 (Distance Confirmation):* When a requester sends out a TASK\_RELEASING message, it may receive multiple ENCRYPTED\_DIST messages from different workers. The requester then decrypts the distance comparison results, selects the reachable workers, and broadcasts a DIST\_CONFIRM message, incorporating the workers who can reach its task and their distance comparison results in the message. The DIST\_CONFIRM message should contain the ID and capacity of the requester, the ID of all the workers who can accept the task (e.g.,  $z_{ij} \leq 0$ ) and their distance comparison results to help the workers make decisions among multiple tasks. However, the distance comparison result should not be transmitted in plaintext. Instead, it should be encrypted using the receiving worker's public key to avoid packet sniffing from other nodes. Therefore, the DIST\_CONFIRM message sent from the requester  $r_i$  is as follows:

$$\text{DIST\_CONFIRM}(r_i) = \left[ c_{t_i}, r_i, w_j, E_{pk_{w_j}}(z_{ij}), w_k, E_{pk_{w_k}}(z_{ik}), \dots \right]$$

where  $E_{pk_{w_j}}(z_{ij})$  is the comparison result between  $t_i$  and  $w_j$ , encrypted by worker  $w_j$ 's public key, while  $w_j$  and  $w_k$  are the reachable workers selected.

To get the real value of the distance comparison result, the requester  $r_i$  uses its homomorphic private key  $dskr_i$  to do decryption, i.e.,

$$z_{ij} = D_{dskr_i}(E_{pk_{r_i}}(z_{ij})). \quad (12)$$

For worker selection, the requester will select all the workers with  $z_{ij} \leq 0$ , which means that task  $t_i$  is within the moving range of worker  $w_j$ .

*Step 4 (Task Proposal):* As a worker may receive several DIST\_CONFIRM messages during a waiting time, it needs to compare the distances of those tasks and select the most suitable ones. The worker will then send TASK\_PROPOSAL messages to tasks it can do after a short backoff.

In this step, we need to design the format of the TASK\_PROPOSAL messages. More importantly, we need to carefully design the mechanism on how the workers collect and send out the message. In brief, we design a waiting mechanism to guarantee the workers to collect enough task information. We also adopt a *proportional backoff* mechanism to enable closer workers to backoff a smaller duration compared with the further ones, in order to increase the possibility of assigning tasks to nearby workers, which are more suitable to complete the task.

*Message Format:* Once receiving a DIST\_CONFIRM message, the worker  $w_j$  uses its private key  $sk_{w_j}$  to decrypt the message and extract the distance comparison result  $z_{ij}$ . The distance between the task  $t_i$  and worker  $w_j$  can hence be derived by  $d_{ij}^2 = z_{ij} + R_j^2$ . The worker  $w_j$  waits for a duration  $T_w$  and ranks all the distances from short to long. The ranked tasks are selected in order until the number of selected tasks exceeds the worker's capacity  $c_{w_j}$ . The worker then sends TASK\_PROPOSAL messages to each of the selected tasks. The TASK\_PROPOSAL message contains the following

domains: the ID of the worker and the ID of the task, i.e.,

$$\text{TASK\_PROPOSAL}(r_i, w_j) = [r_i, w_j].$$

*Wait-and-Decide:* The time of sending the TASK\_PROPOSAL message is also carefully designed. Suppose that at time  $t_a$ , the worker receives the DIST\_CONFIRM message from the requester  $r_i$ . During the next  $T_w$  duration, it receives several more DIST\_CONFIRM messages, but it still decides to propose to requester  $r_i$ . It will then start to conduct backoff to  $r_i$  at time  $t_a + T_w$ . If during the waiting time, another task  $t_k$  comes at time  $t_b$ , and the worker decides to propose to requester  $r_k$  too, it should then start from time  $t_b + T_w$  to start backoff to  $r_k$ . In short, the requester should start backoff at  $T_w$  duration after sending that task's DIST\_CONFIRM message. This is to guarantee that the TASK\_PROPOSAL messages to the same requester are started to conduct backoff *at the same time*. Therefore, in such a fully distributed system, local synchronization can be achieved, and the message with the shortest distance will finish backoff and be sent out first. Note that if a better task keeps on coming during the waiting duration  $T_w$ , the worker has the risk of waiting forever. To avoid this problem, we impose a maximum waiting time  $T_{\max}$  to prevent the worker from waiting forever.

*Proportional Backoff:* The backoff time is proportional to the distance,  $t_{\text{backoff}} = nT_{\text{unit}}$ , where  $n = d_{ij}^2$ .  $T_{\text{unit}}$  is a predefined empirical value decided by the system developers based on their experience or industry requirements, where *this value is known to all workers but no requester inside the system*. The worker will finally send out TASK\_PROPOSAL message at time  $t_a + T_w + t_{\text{backoff}}$  if it selects  $r_i$ . During the backoff time, if the worker hears other workers already sent out the TASK\_PROPOSAL messages to requester  $r_i$ , and the number of proposing workers already exceeds the capacity of the task  $c_{t_i}$ , then it will abort his proposal to  $r_i$ , instead, it will propose to the following tasks according to the ranking.

*Step 5 (Task Assignment):* After receiving several TASK\_PROPOSAL messages from the neighboring workers, the requester notifies those workers by sending them TASK\_CONFIRM messages to confirm their participation to the task. The requester also encapsulates the encrypted location of the task in the TASK\_CONFIRM message to tell the workers where to perform the task.

From time  $t_a + T_w$ , the requester  $r_i$  will continuously receive TASK\_PROPOSAL messages from different workers. Each time the requester receives a TASK\_PROPOSAL( $t_i, w_j$ ), he puts the worker  $w_j$  into a final chosen worker set  $W_{r_i}^*$  and sends the TASK\_CONFIRM message to the worker  $w_j$ . There are three domains in the TASK\_CONFIRM message: the ID of the task  $t_i$ , the ID of the worker  $w_j$ , and the location of the task encrypted with the public key of the worker  $E_{pk_{w_j}}(l_{t_i})$

$$\text{TASK\_CONFIRM}(r_i, w_j) = [r_i, w_j, E_{pk_{w_j}}(l_{t_i})].$$

When the size of  $W_{r_i}^*$  reaches the capacity of the task  $c_{t_i}$ , the requester will no longer confirm any further task proposals.

Upon receiving the TASK\_CONFIRM message, the worker decrypts the message using his private decryption key  $sk_{w_j}$  and

gets the location of the task  $l_{t_i}$ . The location of the task must be known by the selected worker because the final chosen worker has to do that task right on the task's location. Then, the worker moves to the task's location to conduct those assigned tasks. The worker cannot receive new tasks during the busy time  $T_{\text{busy}}$  until all the assigned tasks are completed.

#### D. Remarks on Parameter Decision

There are several predefined parameters in our protocol (e.g.,  $T_{\text{unit}}$ ,  $T_W$ ,  $T_{\text{busy}}$ ). To make the task assignment results better, it is important to select those parameters properly. However, since the actual situation in real world is complex and changeable, it is difficult to deduce optimal parameter values theoretically. Therefore, it is more practical and common to determine these parameters based on past experience, and various scenarios and requirements. For example, if our protocol is applied to a take-out delivery scenario, since workers may move quickly through electric vehicles and travel a short distance,  $T_W$ ,  $T_{\text{busy}}$  can be decided as smaller values (e.g., 10 min or 20 min). If our protocol is used in some spatial crowdsourcing scenarios that take a long time and require loose time limit (e.g., the requester requires multiple workers to come to a specific location to take pictures),  $T_W$ ,  $T_{\text{busy}}$  can be preset to be larger values (e.g., 1 h).

#### E. Running Example

We give a running example to explain how PriTA works, especially the waiting mechanism and the backoff mechanism. The example is shown in Fig. 3, and the corresponding time line is depicted in Fig. 4(a) and (b). The capacity of the task  $c_{t_i}$  and the capacity of the worker  $c_{w_j}$  are both set to 1. We assume workers have infinite moving range  $R_{w_j}$ . We also ignore the message transmitting time and computation time.

We first illustrate the backoff mechanism by analyzing worker 1, requester 1, and worker 3 in Fig. 4(a). Suppose at time  $t_a$ , worker 1 and worker 3 receive the DIST\_CONFIRM message, they will start to wait until  $t_a + T_W$ . Since neither of them receives other tasks, they will start to backoff to requester 1 at  $t_a + T_W$ . As worker 1 is closer to requester 1, task 1 will be assigned to it, and worker 3, once hearing the TASK\_PROPOSAL message from worker 1, will abort the backoff to requester 1.

Second, we explain how our waiting mechanism works in Fig. 4(b). Suppose task 1 is generated before task 2. Therefore, worker 2 will first receive the DIST\_CONFIRM message from requester 1 at time point  $t_a$ . It then starts to wait. During the waiting time  $t_a + T_W$ , worker 2 receives task 2, and the DIST\_CONFIRM message for task 2 is received at time  $t_b$ . When the time reaches  $t_a + T_W$ , worker 2 cannot receive any new arrival tasks. Since worker 2 is closer to task 2, it will select task 2 to perform. Thus, worker 2 will wait until  $t_b + T_W$  and send TASK\_PROPOSAL message to requester 2 at that time.

## IV. ANALYSIS OF PRIVACY PRESERVATION

We show in the following two theorems that PriTA is *privacy preserving* for both workers and requesters, disclosing no

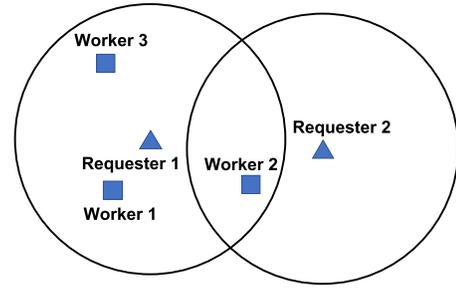


Fig. 3. Running example.

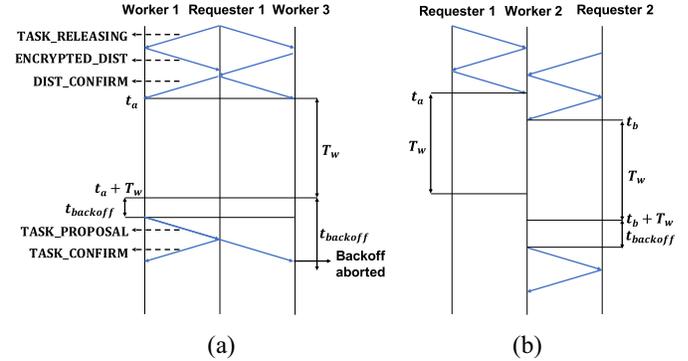


Fig. 4. Basic procedure. (a) Backoff mechanism. (b) Waiting mechanism.

entity's exact location information to any other parties inside the system.

**Theorem 1:** The real location of each worker will not be disclosed to other workers and requesters under the PriTA framework.

*Proof:* The message that contains the location information for a worker  $w_j$  is in the ENCRYPTED\_DIST message. Without loss of generality, assume  $w_j$  communicates with requester  $r_i$ ; thus, only  $r_i$  can successfully decrypt  $E_{\text{epk}_{r_i}}(z_{ij})$  in ENCRYPTED\_DIST, because no party else has the corresponding homomorphic decryption key  $\text{dsk}_{r_i}$ . Even if there is a third party eavesdropper listening all the transmission messages, it cannot derive the real location of a worker. As for the requester  $r_i$ , even though it can decrypt  $E_{\text{epk}_{r_i}}(z_{ij})$ , it can only learn whether  $w_j$  is capable to do his task and not be able to derive the worker's location based on the decrypted data.

In step 4, the requesters will receive TASK\_PROPOSAL messages from different workers based on various backoff time. However, the unit backoff time  $T_{\text{unit}}$  is only known by workers. Furthermore, influenced by the real network environment (e.g., network congestion degree), the real backoff time for each worker might be different from the theoretical value. Due to the above two reasons, a noncolluded requester, hence, has nearly no chance to deduce the distance between itself and the workers who send TASK\_PROPOSAL messages to it. Even though a requester can infer the distance between workers and itself, it cannot further deduce the exact location of any worker. This is because, first, a requester and its task are in the same location. Second, since the RA only authorizes each requester/worker with one identity, and requesters are honest but curious, a requester cannot break the protocol

and create multiple fake identities or tasks in different locations at the same time in order to learn the exact location of a specific worker.

In summary, the location privacy for every worker can be well protected in the PriTA framework. ■

*Theorem 2:* In the PriTA framework, the real location of each task will not be compromised to other workers and requesters before the task assignment and will only be revealed to the workers who will perform that task.

*Proof:* The location for a task  $t_i$  is encrypted as  $E_{epk_{r_i}}(l_{t_i})$ , and is packaged in TASK\_RELEASE message and then sent to surrounding workers. Other parties who have overheard this message cannot decrypt it as the homomorphic decryption key  $dsk_{r_i}$  is only held in requester  $r_i$ . Due to the existence of RA and the assumption that workers are *honest but curious*, they have to obey the protocol and cannot fake their locations or create multiple identities. Thus, by receiving DIST\_CONFIRM from a requester  $r_i$ , a worker  $w_j$  can only derive the distance between itself and the requester  $d_{ij}$  but no more information, hence cannot know the exact location of the task. Other workers, requesters, and even eavesdroppers even cannot derive the distance because they do not know the decryption key for that worker,  $sk_{w_j}$ , and do not know the moving range of the worker  $w_j$ . Finally, the location of a task  $t_i$  in  $E_{pk_{w_j}}(l_{t_i})$  in the TASK\_CONFIRM message sent from  $r_i$  can only be obtained by worker  $w_j$ , who is willing to perform  $t_i$ , since only  $w_j$  has the decryption key  $sk_{w_j}$ .

According to the above proof, the PriTA framework can protect the location privacy for any task before the task assignment, and the real location for a task will only be released to a worker who is willing to do it. ■

It is worth mentioning that the revelation of the real location *after* task assignment is inevitable, as workers need to physically move to the task locations. This, however, is widely accepted in [25], [26], and [29]. Also, as mentioned before, we mainly consider the privacy threats posed by internal entities, i.e., workers and requesters, and security attacks from external malicious attackers are not our concern. Having said that even though there are some attackers who are eavesdropping our system, they still can learn nothing about the exact locations of tasks or workers. This is because messages containing location information are encrypted in every single step, and the decryption key is held only by the corresponding worker or requester. Moreover, even though a few eavesdroppers have the ability to learn the backoff time of each worker, a non-colluded attacker cannot further infer the distances between tasks and workers. While a colluded external attacker can only deduce the distances between tasks and workers but nothing more beyond that.

## V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of PriTA. The main results are summarized as follows: 1) PriTA achieves better privacy-preserving performance compared with differential privacy-based scheme (DP) under the same setting; 2) despite being fully distributed, PriTA makes *near-optimal* task assignments close to the centralized global optimum under various practical scenarios. The design of wait-and-decide

and proportional-backoff schemes help to improve the task assignment results dramatically; and 3) the computational complexity and the communication overhead of PriTA are practical in real system.

### A. Experimental Setup

*Data Sets:* Our evaluation is based on the Google Cloud NYC taxi and limousine commission (TLC) data set [1], which contains hundreds of thousands of trips completed by taxis. In this case, the TLC drivers are workers located at the drop-off sites; the TLC passengers are requesters located at their pick-up sites. Tasks are generated in an order of their pick-up times. Workers are always online during the experiments except they are performing tasks during the busy time  $T_{\text{busy}}$ . We also consider to use three auxiliary data sets to demonstrate the performance of our system, i.e., a T-Drive data set [26], a NYC Citi Bike Trips data set [1], and a Gowalla data set [19].

*Settings:* In the experiments, by default, we randomly sample 300 requesters and 300 workers from the data set. We set the moving range of the worker  $R_j$  to be a random integer within 5 km. We assume a uniform message transmission range across all requesters and workers, i.e.,  $R_t = 5$  km. This is a practical setting due to the advanced long-range wireless data communication technology [2]. The capacity of the task  $c_t$  and the capacity of each worker  $c_w$  can be different (heterogeneous) or uniform (homogeneous). The default values for  $c_t$  and  $c_w$  is 10 and 2, respectively, which could be considered as each task has ten subtasks and requires at most ten workers to perform, and each worker can do at most two subtasks among them. We assume Poisson task arrivals where the arrival interval  $\lambda$  is by default set to 10 s. The maximum waiting time  $T_w$  and the busy time of workers  $T_{\text{busy}}$  are both set to 10 min.

In our experiments, we assume a stable network environment where the message transmission time is negligible compared with the unit backoff time  $T_{\text{unit}}$ . We make this simplified assumption because the main focus of this work is to make effective task assignment while preserving location privacy, rather than task assignment in various network environments. That being said, in practice, the backoff time can be empirically determined based on the network delay measured in real scenarios. We shall leave the evaluation in real network environment as a future work.

*Baselines:* We evaluate the performance of our solution PriTA against three baseline schemes: 1) the state-of-the-art DP [26]; 2) the global optimal scheme (OPT), which leverages a global manager overseeing all the requesters and workers to make the optimal task assignment periodically; and 3) and PriTA with no backoff (*PriTA-NB*), in which the workers send TASK\_PROPOSAL messages without conducting backoff proportional to the distance in step 4, and the requesters receive TASK\_PROPOSAL messages and choose workers randomly. The comparison with PriTA-NB can help quantify the benefit brought by the backoff mechanism.

*Metric:* We use four metrics, the *privacy-preserving level*, the *total number of the assigned tasks*, *computation time*, and *communication time*, which evaluate the privacy-preserving

performance, task assignment performance, computational cost, and communication overhead, respectively.

### B. Privacy Preserving Evaluation

*Methodology:* We compare PriTA with DP on T-Drive data set to evaluate the privacy-preserving performance. We use precisely the same settings as those in the system DP (e.g.,  $c_t = 1$  and  $c_w = 1$ ) to make our comparison results more convincing. We use privacy preserving level as the primary metric, which is defined as the probability that the real location of the privacy-protected user cannot be correctly inferred.

For PriTA, the privacy preserving level is given by

$$L_{\text{PriTA}} = 1 - P_{\text{PriTA}}(r, d_{ij}) = 1 - \frac{r^2}{d_{ij}^2 - (d_{ij} - 1)^2}. \quad (13)$$

Because in PriTA, a worker can derive the distance  $d_{ij}$  between itself and the requester. The real location of the requester falls into a circle centered at the worker's location. Therefore, the probability that the real location can be guessed out (falls into a small circular area with radius  $r$ ) equals to the area of the small circle over the area of the already-known annulus related with  $d_{ij}$ , i.e.,  $P_{\text{PriTA}}(r, d_{ij}) = [r^2 / (d_{ij}^2 - (d_{ij} - 1)^2)]$ .

In DP, the privacy-preserving level is given by

$$L_{\text{DP}} = 1 - P_{\text{DP}}(r) = \left(1 + \frac{\epsilon}{R}r\right) \cdot e^{-\frac{\epsilon}{R}r}. \quad (14)$$

In DP, each location of the worker or task is protected by reporting a perturbed location point generated by adding a Planar Laplace noise into the original location. According to [3], the probability that the true location falls in a circle around the perturbed point with radius  $r$  is given as  $P_{\text{DP}}(r) = 1 - (1 + [\epsilon/R]r) \cdot e^{-(\epsilon/R)r}$ .

To compare PriTA with DP, we change the parameters (e.g., moving range  $R_j$ ) to guarantee that PriTA achieves the same number of assigned tasks as DP, and we then compare the privacy-preserving level. We set  $r = R$ , and  $d_{ij}$  will change with  $R_j$ . We also compare the number of assigned tasks when the privacy-preserving level for the two schemes keeps the same.

*Results:* Fig. 5(a) shows that PriTA achieves higher privacy-preserving level than DP, meaning that an attacker has a lower chance to make a successful guess that a real location falls in a certain circle in PriTA. Fig. 5(b) shows that PriTA achieves better performance on task assignment when PriTA and DP protect the same level of privacy. The reason behind is that our scheme provides accurate distance for the workers compared with DP, and effective task assignment usually highly relies on the distance information.

### C. Task Assignment Evaluation

We compare the number of assigned tasks among OPT, PriTA, and PriTA-NB, under various scenarios, and draw the conclusion that PriTA attains near-optimal performance, significantly outperforming PriTA-NB.

*Methodology:* We run the experiment under various scenarios: 1) varying worker's capacity, while keeping the capacity of all the workers the same [Fig. 6(a)]; 2) varying task's

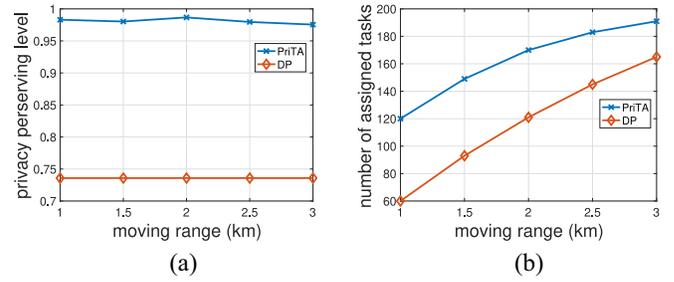


Fig. 5. Privacy preserving evaluation. (a) Privacy-preserving level. (b) Assigned tasks.

capacity, while keeping the capacity of all the tasks the same [Fig. 6(b)]; 3) varying both capacity, while the capacity among different workers and tasks are different [Fig. 6(c)], the capacity of both workers and tasks is varied within the range of 1 to  $x$ ; 4) varying task arrival interval [Fig. 6(d)]; 5) varying worker's moving range [Fig. 6(e)]; and 6) varying the number of requesters/workers [Fig. 6(f)]. Except for the TLC data set, we also repeat all the above experiment on Gowalla and NYC Citi Bike data sets. Due to page limit, we only show the number of assigned tasks against task/requester's capacity under a heterogeneous scenario (Fig. 7).

#### Results:

- 1) All the figures in Fig. 6 show that PriTA achieves relatively good assignment results compared with OPT, and the gap is mainly due to that in the fully distributed scenario no one has the global view of the system. Moreover, the naive PriTA-NB acts as the worst as the best worker has no priority and the other workers cannot smartly switch to a suitable task via backoff overhead and abortion.
- 2) The number of the assigned tasks increases with the worker's capacity, task's capacity, task arrival interval, worker's moving range, and the number of workers/requesters for all three schemes. Because by increasing any of the above parameters, the potential matching tasks will increase.
- 3) Fig. 7 shows that in Gowalla and NYC Citi Bike data sets, the performance trend under varying parameters is similar as in the TLC data set, which means that our scheme is not sensitive to parameters or data set, thus is applicable for a wide range of scenarios.

The number of assigned tasks in Figs. 6 and 7 is observed to be larger than 300. The reason is that each task's capacity  $c_t$  could be larger than 1; hence, the number of assigned tasks is actually the number of assigned subtasks, which could be far more larger than 300.

### D. Computation Cost Evaluation

*Methodology:* We evaluate the computation cost of the workers and requesters in our proposed PriTA framework. Specifically, we record the time cost for encryption, decryption, and computation on the ciphertext during the whole procedure. The experiment is run on a desktop with an Intel Core CPU 2.7-GHz processor and an 8-GB RAM running Ubuntu 14.04.

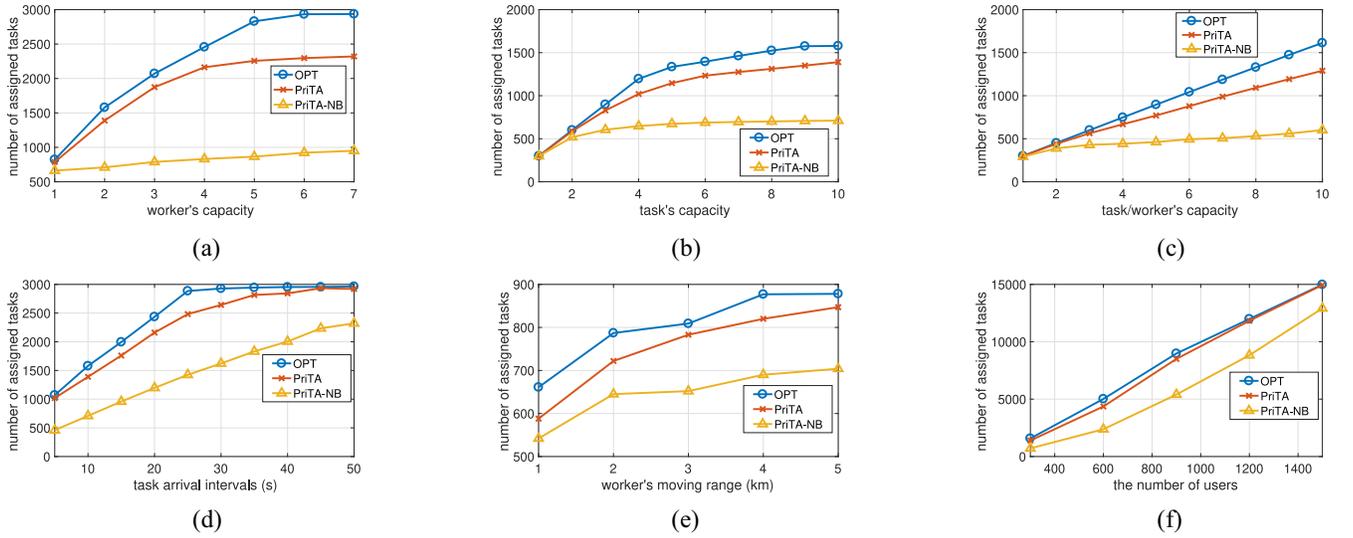


Fig. 6. Task assignment evaluation. (a) Varying worker's capacity. (b) Varying task's capacity.  $c_r$ . (c) Varying worker/task's capacity. (d) Varying task arrival interval. (e) Varying worker's moving range. (f) Varying the number of users.

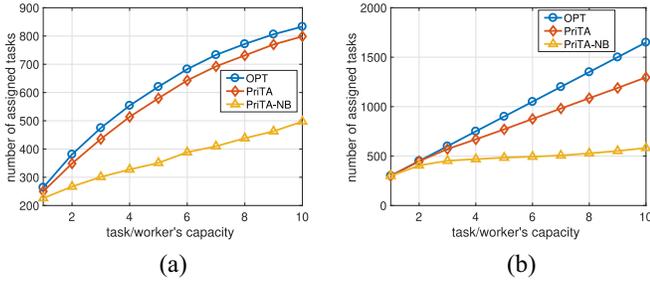


Fig. 7. Data set evaluation. (a) Gowalla. (b) NYC Citi Bike.

TABLE II  
AVERAGE COMPUTATION TIME OF A REQUESTER

	$E_{epk_{r_i}}(l_i)$	$D_{dsk_{r_i}}(E_{epk_{r_i}}(z_{ij}))$	$E_{pk_{w_j}}(z_{ij})$
unit time (ms)	4.0588	0.4233	2.0574
total time (ms)	4.0588	26.4563	19.0310

TABLE III  
AVERAGE COMPUTATION TIME OF A WORKER

	$E_{epk_{r_i}}(l_{w_j})$ $E_{epk_{r_i}}(R_j)$	computation on ciphertext	$D_{pk_{w_j}}(E_{pk_{w_j}}(z_{ij}))$
unit time (ms)	6.0561	7.7330	0.4329
total time (ms)	379.2653	485.2518	4.004

*Results:* The average computation time of a single requester is shown in Table II. The task location is encrypted only once so the time cost for  $E_{epk_{r_i}}(l_i)$  is the same between the total time and the unit time, while  $D_{dsk_{r_i}}(E_{epk_{r_i}}(z_{ij}))$  and  $E_{pk_{w_j}}(z_{ij})$  have to be done multiple times for different workers, thus resulting in the difference between the total time and unit time. Generally speaking, homomorphic encryption costs  $10\times$  longer time than homomorphic decryption for a single operation.

Table III shows that for a single worker, the computation time is mainly spent on encrypting  $l_w$  and  $R_j$  and computing on the encrypted distance result in step 2. The total computation time for both workers and requesters is less than 1 s, which is acceptable and feasible in practice.

### E. Communication Overhead

*Methodology:* We measure the communication cost for all the workers and requesters in our PriTA system. More specifically, we evaluate the total data size of each worker, and each requester needs to send and receive in every step.

*Results:* The results of our communication cost evaluation are shown in Tables IV and V. The sizes of the data that are transmitted in our experiments are all below few kilobytes, which represents that our PriTA provides a relatively low and practical communication overhead. Specifically, in Table IV, the TASK\_RELEASE and DIST\_CONFIRM messages that a requester sends have much smaller size. The reason is that these two messages are sent by broadcast, thus do not need to be sent multiple times for different workers. While for a worker, it will receive multiple TASK\_RELEASE and DIST\_CONFIRM messages from different requesters, leading to a bigger received data size for TASK\_RELEASE and DIST\_CONFIRM in Table V.

## VI. RELATED WORKS

In this section, we review the literature from the following three aspects: 1) task assignment in spatial crowdsourcing; 2) location privacy preserving in spatial crowdsourcing; and 3) fully distributed crowdsourcing.

*Task Assignment in Spatial Crowdsourcing:* Many studies have focused on how to assign tasks in spatial crowdsourcing. Kazemi and Shahabi [16] classified spatial crowdsourcing into two modes: 1) worker selected tasks (WSTs) mode and 2) server assigned tasks (SATs) mode, where workers positively select tasks in WST and negatively wait for tasks to be

TABLE IV  
AVERAGE COMMUNICATION OVERHEAD OF A REQUESTER

	TASK_RELEASEING (send)	ENCRYPTED_DIST (receive)	DIST_CONFIRM (send)	TASK_PROPOSAL (receive)	TASK_CONFIRM (send)
overall data size (KB)	0.09	5.02	0.17	0.71	0.42

TABLE V  
AVERAGE COMMUNICATION OVERHEAD OF A WORKER

	TASK_RELEASEING (receive)	ENCRYPTED_DIST (send)	DIST_CONFIRM (receive)	TASK_PROPOSAL (send)	TASK_CONFIRM (receive)
overall data size (KB)	5.63	5.02	1.58	0.71	0.42

assigned by the server in SAT. The studies on SAT [16], [30] generally aim to maximize the number of assigned tasks by the server, while works on WST [6], [8] target at maximizing the number of completed tasks selected by workers. Only a few studies [9] combine SAT and WST together, in which the server first assigns tasks to workers and then workers select the tasks he can do to maximize the number of the final completed tasks. However, all of the aforementioned works rely on a server, and neither of them considers to protect the location privacy of workers or tasks.

*Location Privacy Preserving in Spatial Crowdsourcing:* To protect the location privacy in spatial crowdsourcing, the existing works mainly leverage  $k$ -anonymity or differential privacy techniques. Gruteser and Grunwald [12] used  $k$ -anonymity to achieve location privacy for a worker, which requires a server to obfuscate an area containing the worker's location and the positions of  $k - 1$  other workers. To *et al.* [25] proposed to use a differential privacy technique called PSD [7] to protect the location privacy of workers, while it needs a trusted third party to sanitize the collected location information of workers. A local differential privacy technique named geo-indistinguishability [3] is leveraged in [26] and [29] to protect the locations of both tasks and workers; however, they still require a server to match different tasks with workers based on the obfuscated locations. Liu *et al.* [18] used homomorphic encryption to encrypt the locations of both workers and tasks, thus achieving the protection of location privacy. In their work, two servers are needed and they have to communicate with each other multiple times to exchange information in order to assign tasks to proper workers. All the above-mentioned works require a centralized server to guarantee privacy preserving, which is not applicable in fully distributed crowdsourcing systems.

*Fully Distributed Crowdsourcing:* Recently, a few works start focusing on fully distributed, self-organized crowdsourcing. Chang and Wu [5] proposed a distributed and self-organized crowdsourcing scheme within mobile social networks, in which the requester positively sends the task to the potential workers via multihop social contacts. In [23], a self-organized mobile crowdsourcing paradigm is proposed, where a mobile requester can proactively crowdsource his task by leveraging the encountered workers at real time. However, the protection of location privacy is beyond their consideration.

## VII. CONCLUSION

In this article, we proposed a novel privacy-preserving framework PriTA for spatial crowdsourcing, which performs task assignment in a fully distributed manner without disclosing location information. PriTA uses homomorphic encryption to protect the location privacy for both tasks and workers. We proposed a distributed protocol achieving efficient task assignment based on exchanges of encrypted messages between requesters and workers. The evaluations driven by real-world data set demonstrated that despite of being fully distributed, PriTA achieves more efficient task assignment at the same level of location privacy compared with existing works.

## REFERENCES

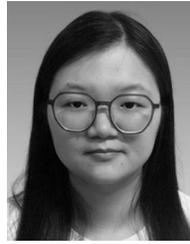
- [1] *Google Cloud*. Accessed: Sep. 1, 2020. [Online]. Available: <https://cloud.google.com/bigquery/public-data/>
- [2] *LoRa*. Accessed: Sep. 1, 2020. [Online]. Available: <https://www.semtech.com/technology/lora>
- [3] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," 2012. [Online]. Available: arXiv:1212.1984.
- [4] C. Cai, Y. Zheng, and C. Wang, "Leveraging crowdsensed data streams to discover and sell knowledge: A secure and efficient realization," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2018, pp. 589–599.
- [5] W. Chang and J. Wu, "Progressive or conservative: Rationally allocate cooperative work in mobile social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 7, pp. 2020–2035, Jul. 2015.
- [6] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 157–166.
- [7] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 20–31.
- [8] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *Proc. 21st ACM Sigspatial Int. Conf. Adv. Geogr. Inf. Syst.*, 2013, pp. 314–323.
- [9] D. Deng, C. Shahabi, and L. Zhu, "Task matching and scheduling for multiple workers in spatial crowdsourcing," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2015, p. 21.
- [10] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," in *Proc. IACR Cryptol. ePrint Archive*, 2012, p. 144.
- [11] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*, vol. 20. Stanford, CA, USA: Stanford Univ., 2009.
- [12] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. 1st Int. Conf. Mobile Syst. Appl. Services*, 2003, pp. 31–42.
- [13] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "SmartRoad: Smartphone-based crowd sensing for traffic regulator detection and identification," *ACM Trans. Sens. Netw. (TOSN)*, vol. 11, no. 4, p. 55, 2015.
- [14] P. G. Ipeirotis, "Analyzing the amazon mechanical turk marketplace," *XRDS, Crossroads ACM Mag. Students*, vol. 17, no. 2, pp. 16–21, 2010.

- [15] L. Kazemi and C. Shahabi, "A privacy-aware framework for participatory sensing," *ACM SIGKDD Explor. Newslett.*, vol. 13, no. 1, pp. 43–51, 2011.
- [16] L. Kazemi and C. Shahabi, "GeoCrowd: Enabling query answering with spatial crowdsourcing," in *Proc. 20th Int. Conf. Adv. Geogr. Inf. Syst.*, 2012, pp. 189–198.
- [17] M. Li *et al.*, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1251–1266, Jul. 2019.
- [18] B. Liu, L. Chen, X. Zhu, Y. Zhang, C. Zhang, and W. Qiu, "Protecting location privacy in spatial crowdsourcing using encrypted data," in *Proc. Adv. Database Technol. (EDBT)*, 2017, pp. 478–481.
- [19] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manag.*, 2014, pp. 739–748.
- [20] I. Lykourantzou, S. Wang, R. E. Kraut, and S. P. Dow, "Team dating: A self-organized team formation strategy for collaborative crowdsourcing," in *Proc. Conf. Extended Abstracts Human Factors Comput. Syst.*, 2016, pp. 1243–1249.
- [21] C. Miao, L. Su, W. Jiang, Y. Li, and M. Tian, "A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2017, pp. 1–9.
- [22] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter, "Mobile crowd sensing in space weather monitoring: The Mahali project," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 22–28, Aug. 2014.
- [23] L. Pu, X. Chen, J. Xu, and X. Fu, "Crowdlet: Optimal worker recruitment for self-organized mobile crowdsourcing," in *Proc. IEEE INFOCOM 35th Annu. Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
- [24] L. Pu, X. Chen, J. Xu, and X. Fu, "Crowd foraging: A QoS-oriented self-organized mobile crowdsourcing framework over opportunistic networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 4, pp. 848–862, Apr. 2017.
- [25] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *Proc. VLDB Endowment*, vol. 7, no. 10, pp. 919–930, 2014.
- [26] H. To, C. Shahabi, and L. Xiong, "Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, 2018, pp. 833–844.
- [27] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, 2016, pp. 49–60.
- [28] C. Wang, H. Liu, K.-L. Wright, B. Krishnamachari, and M. Annavam, "A privacy mechanism for mobile-based urban traffic monitoring," *Pervasive Mobile Comput.*, vol. 20, pp. 1–12, Jul. 2015.
- [29] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma, "Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 627–636.
- [30] X. Wang, R. Jia, X. Tian, and X. Gan, "Dynamic task assignment in crowdsensing with location awareness and location diversity," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 2420–2428.



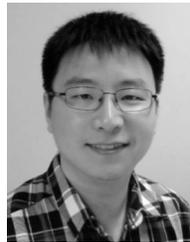
**Mingzhe Li** (Student Member, IEEE) received the B.E. degree in communication engineering from the Southern University of Science and Technology, Shenzhen, China, in 2016. He is currently pursuing the Ph.D. degrees with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, and the Southern University of Science and Technology.

His research interests are mainly in blockchain, network economics, and wireless communication and networks.



**Jingrou Wu** (Student Member, IEEE) received the B.E. degree in computer science and engineering from the Southern University of Science and Technology, Shenzhen, China, in 2019. She is currently pursuing the Ph.D. degree with the School of Computer Science, University of Technology, Sydney, NSW, Australia, and the Department of Computer Science and Engineering, Southern University of Science and Technology.

Her research interests focus on network economics and blockchain technology.



**Wei Wang** (Member, IEEE) received the B.Eng. and M.Sc. degrees from Shanghai Jiao Tong University, Shanghai, China, in 2007 and 2010, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, in 2015.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong, where he is also affiliated with HKUST Big Data Institute. His research interests cover the broad area of networking and distributed systems, with a special focus on big data and machine learning systems, cloud computing, and computer networks.



**Jin Zhang** (Member, IEEE) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2004 and 2006, respectively, and the Ph.D. degree in computer science from Hong Kong University of Science and Technology, Hong Kong, in 2009.

She is currently an Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Her research interests are mainly in mobile healthcare and wearable computing, wireless communication and networks, network economics, cognitive radio networks, and dynamic spectrum management.