Towards Privacy-Preserving Task Assignment for Fully Distributed Spatial Crowdsourcing

Mingzhe Li, *Student Member, IEEE*, Jingrou Wu, *Student Member, IEEE*, Wei Wang, *Member, IEEE*, and Jin Zhang, *Member, IEEE*

***Abstract*—With the proliferation of human-carried mobile devices, spatial crowdsourcing has emerged as a transformative system, where requesters outsource their spatio-temporal tasks to a set of workers who are willing to perform the tasks at the specified locations. However, in order to make efficient assignments, 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 paper, 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 datasets illustrate that our proposed system achieves a large number of task assignments in an efficient and privacy-preserving manner.**

***Index Terms*—Spatial crowdsourcing, task assignment, privacy preserving, distributed system.**

I. Introduction

THANKS to the proliferation of mobile devices and the advancement in sensor technologies, data collection 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], [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], [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 has 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 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, 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.

Existing studies assume a *centralized* model for privacy protection in which a server oversees a global picture and makes task assignment for requesters [25], [15], [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 one hour 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

M. Li is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and Hong Kong University of Science and Technology, Hong Kong (email: mlibn@cse.ust.hk).

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

J. 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 (email: zhangj4@sustech.edu.cn).

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

J. Zhang and W. Wang are the corresponding authors.

became a more promising framework for efficient task assignment by exploiting nearby human intelligence [23], [5], [24], [20]. 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 PriTA (PRIvacy-preserving Task Assignment), 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, 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 recieved 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 datasets. 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 paper 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 the paper in Section VII.

II. Background and System Model

A. Spatial Crowdsourcing Model

In traditional spatial crowdsourcing systems, there are usually three main entities: the *requesters*, the *workers*, and the *platform (server)*. The requesters want to outsource some spatio-temporal 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 [23], [5], [24], [20]. 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 paper, we will focus on the task assignment on such *fully-distributed* spatial crowdsourcing systems.

Figure 1 gives an architecture overview of our system model. There are M requesters $\{r_1, r_2, ..., r_i, ..., r_M\}$, and N workers $\{w_1, w_2, ..., w_j, ..., w_N\}$. The requester has some tasks needed to be completed in its location. In this paper 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 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 platform, the task assignment between the tasks and the workers can only be

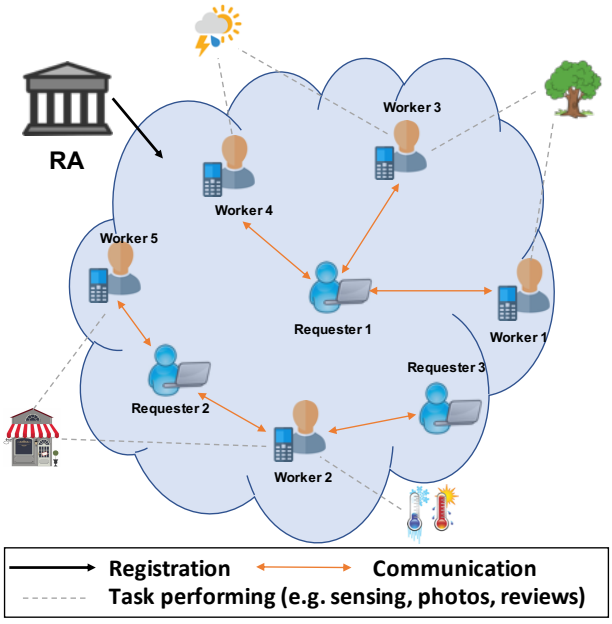


Fig. 1: System architecture.

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, a message can not 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 paper.

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 [26], [21], [4], 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 paper.

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 paper, we propose to use Fan-Vercauteren (FV) homomorphic encryption and only use the somewhat homomorphic encryption of [10]. FV homomorphic encryption is built upon the hardness of Ring-Learning-with-Errors (RLWE) 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 paper:

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

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

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 which 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 privacy-preserving task assignment 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, sends the encrypted results afterwards. 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 5 main steps along with an initialization step in our framework. Firstly, 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 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, 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 the RA upon joining the system. Each authenticated requester/worker is assigned a unique ID. As a result, no malicious external attacker holds the probability to join in the system and 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}, dsk_{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 doesn't 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_{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 subsection, we describe our privacy-preserving, fully distributed task assignment protocol. The basic flow chart of our system is illustrated in Figure 2.

Step 1: task releasing

When the requester has a task needed to be done, it broadcasts a TASK_RELEASING message to release the task to its neighboring workers. The message contains three domains: the requester's ID r_i, the homomorphic public key epk_{r_i} of requester r_i and the encrypted location of task t_i using the requester's homomorphic public key epk_{r_i}. That is,

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

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.,

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

Step 2: distance computing

Once a worker receives a TASK_RELEASING 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

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

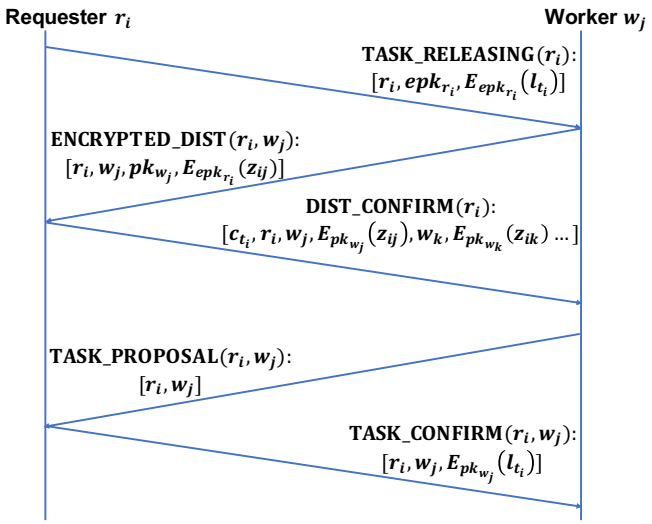


Fig. 2: PriTA protocol design.

the comparison result in the ENCRYPTED_DIST message and sends the message to the requester. Note that the distance comparison are 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 falls into its moving range R_j. We use the Euclidean distance from worker w_j to task t_i, i.e.,

$$d_{ij} = \sqrt{\Delta x_{ij}^2 + \Delta y_{ij}^2}, \tag{5}$$

where Δx_{ij} and Δy_{ij} are the difference of x- and y-coordinate 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 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: the ID of the requester r_i, the ID of the worker w_j, the worker's public key epk_{w_j}, and $E_{epk_{r_i}}(z_{ij})$, 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:

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

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, Eq. (6) can be computed by bringing Eqs. (7), (8) into Eqs. (9), (10) and bringing Eqs. (9), (10), (11) into Eq. (6):

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

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

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

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

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

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 requester r_i is as follows:

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

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 dsk_{r_i} to do decryption, i.e.,

$$z_{ij} = D_{dsk_{r_i}}(E_{epk_{r_i}}(z_{ij})). \tag{12}$$

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 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, $t_{backoff} = nT_{unit}$, where $n = d_{ij}^2$. T_{unit} is a predefined empirical value decided by the system developers based on their experience or industry requirements, *this value is known to all workers but no requester inside the system*. The worker will finally send out TASK_PROPOSAL message at time $t_a + T_w + t_{backoff}$ 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_COMFIRM 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_{busy} until all the assigned tasks are completed.

D. Remarks on Parameter Decision

There are several predefined parameters in our protocol (e.g., T_{unit}, T_W, T_{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

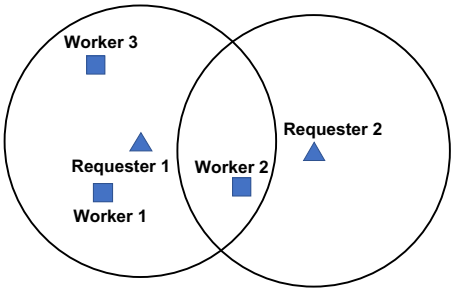
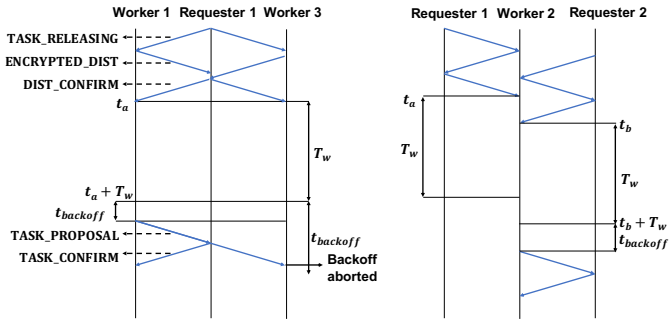


Fig. 3: A running example.



(a) Backoff mechanism. (b) Waiting mechanism.

Fig. 4: Basic procedure.

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_{busy} can be decided as smaller values (e.g., 10 minutes or 20 minutes). 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_{busy} can be preset to be larger values (e.g., 1 hour).

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 Figure 3 and the corresponding time line is depicted in Figure 4a and Figure 4b. 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 Figure 4a. 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 Figure 4b. 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 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 ENCRYPTED_DIST message. Without loss of generality, assume w_j communicates with requester r_i, thus only r_i can successfully decrypt $E_{epk_{r_i}}(z_{ij})$ in ENCRYPTED_DIST, because no party else has the corresponding homomorphic decryption key dsk_{r_i}. Even if there is a third party eavesdropper listening all the transmission messages, it can not derive the real location of a worker. As for the requester r_i, even though it can decrypt $E_{epk_{r_i}}(z_{ij})$, 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_{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 non-colluded 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, firstly, a requester and its task are in the same location. Secondly, 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_RELEASING message 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 the 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 don't know the decryption key for that worker, sk_{w_j}, and don't know the moving range of 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 the literature [25], [29], [26]. 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. (3) The computational complexity and the communication overhead of PriTA are practical in real system.

A. Experimental Setup

Datasets: Our evaluation is based on the Google Cloud NYC Taxi and Limousine Commission (TLC) dataset [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_{busy}. We also consider to use three auxiliary dataset to demonstrate the performance of our system, i.e., a T-Drive dataset [26], a NYC Citi Bike Trips dataset [1], and a Gowalla dataset [19].

Settings: In the experiments, by default, we randomly sample 300 requesters and 300 workers from the dataset. 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 = 5km. 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 10 subtasks and requires at most 10 workers to perform, and each worker can do at most 2 subtasks among them. We assume Poisson task arrivals where the arrival interval λ is by default set to 10 s. The maximum waiting time T_w and the busy time of workers T_{busy} are both set to 10 minutes.

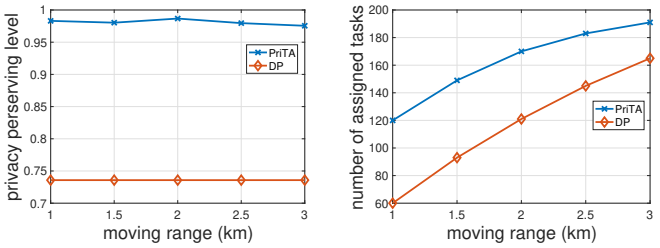
In our experiments, we assume a stable network environment where the message transmission time is negligible compared with the unit backoff time T_{unit}. We make this simplified assumption because the main focus of this work is to make effective task assignment while preserving location privacy, rather 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 differential privacy-based scheme (*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, (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. 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*, the *computation time*, and the *communication time*, that respectively evaluate the privacy preserving performance, task assignment performance, computational cost, and the communication overhead.

B. Privacy Preserving Evaluation

Methodology: We compare PriTA with DP on T-Drive dataset to evaluate the privacy preserving performance. We



(a) Privacy preserving level. (b) Assigned tasks.

Fig. 5: Privacy preserving evaluation.

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

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

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

In DP, the privacy preserving level is given by

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

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 $P_{DP}(r) = 1 - (1 + \frac{\epsilon}{R} r) \cdot e^{-\frac{\epsilon}{R} r}$.

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, 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 privacy preserving level for the two schemes keeps the same.

Results: Figure 5a 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. Figure 5b 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 (Figure 6a); (2) varying task's capacity, while keeping the capacity of all the tasks the same (Figure 6b); (3) varying both capacity, while the capacity among different workers and tasks are different (Figure 6c), the capacity of both workers and tasks is varied within the range of 1 to x; (4) varying task arrival interval (Figure 6d); (5) varying worker's moving range (Figure 6e); (6) varying number of requesters/workers (Figure 6f). Except for the TLC dataset, we also repeat all the above experiment on Gowalla and NYC Citi Bike dataset. Due to page limit, we only show the number of assigned tasks against task/requester's capacity under heterogeneous scenario (Figure 7).

Results: (1) All the figures in Figure 6 show that PriTA achieves relatively good assignment results compared with OPT, 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 switched to a suitable task via backoff overhear and abortion. (2) The number of the assigned tasks increase 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) Figure 7 shows that in Gowalla and NYC Citi Bike datasets, the performance trend under varying parameters is similar as in TLC dataset, which means that our scheme is not sensitive to parameters or dataset, thus is applicable for a wide range of scenarios.

The number of assigned tasks in Figure 6 and Figure 7 are 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.

TABLE II: Average computation time of a requester.

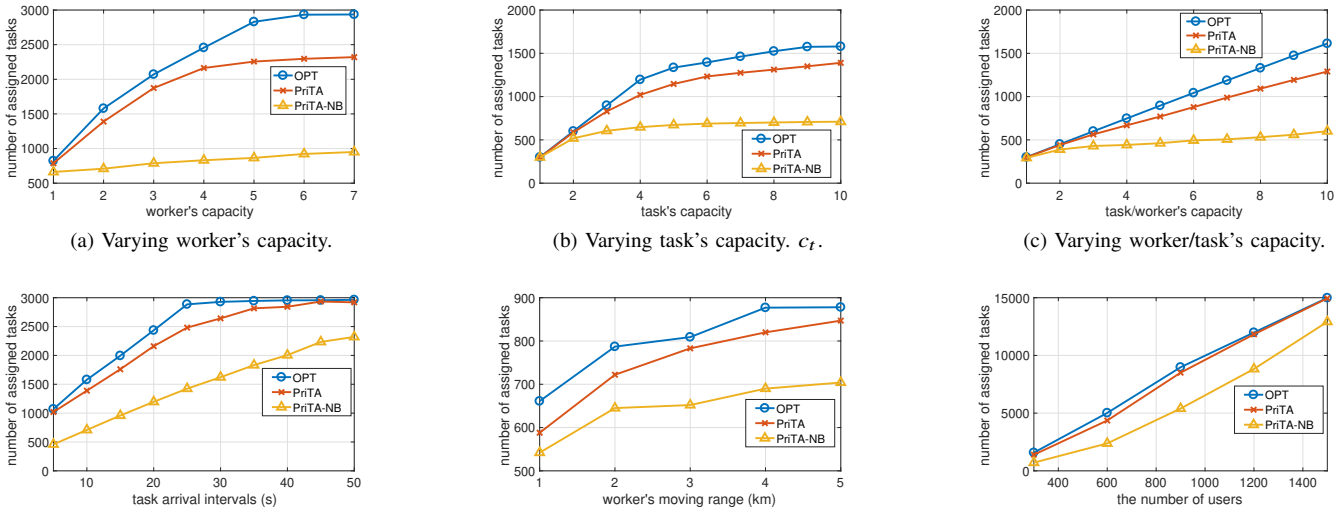
	$E_{epk_{r_i}}(l_{t_i})$	$D_{dsk_{r_i}}(E_{epk_{r_i}}(z_{ij}))$	$E_{pk_{w_j}}(z_{ij})$
unit time (ms)	4.0588	0.4233	2.0574
total time (ms)	4.0588	26.4563	19.0310

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

D. Computation Cost Evaluation

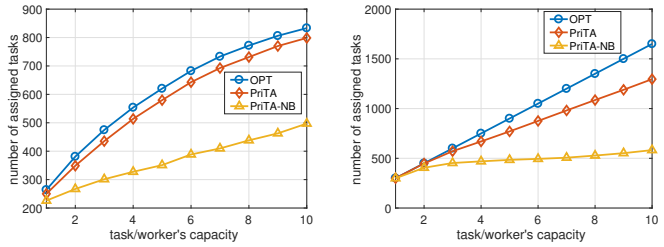
Methodology: We evaluate the computation cost of the workers and requesters in our proposed PriTA framework. Specifically, we record the time cost for encryption, decryption, and computation on the ciphertext during the whole



(a) Varying worker's capacity. (b) Varying task's capacity. c_t. (c) Varying worker/task's capacity.

(d) Varying task arrival interval. (e) Varying worker's moving range. (f) Varying the number of users.

Fig. 6: Task assignment evaluation.



(a) Gowalla. (b) NYC Citi Bike.

Fig. 7: Dataset evaluation.

procedure. The experiment is run on a desktop with an Intel Core CPU 2.7GHz processor and an 8GB RAM running Ubuntu 14.04.

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 $E_{epk_{r_i}}(l_{t_i})$ 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 the difference between the total time and unit time. Generally speaking, homomorphic encryption cost 10x 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 one second, 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 each worker and each requester needs to send and receive in every step.

Results: The results of our communication cost evaluation are shown in Table IV and Table 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_RELEASING and DIST_CONFIRM messages that a requester sends have much smaller size. The reason is that these two messages are sent by broadcast thus don't need to be sent multiple times for different workers. While for a worker, it will receive multiple TASK_RELEASING and DIST_CONFIRM messages from different requesters, leading to a bigger received data size for TASK_RELEASING and DIST_CONFIRM in Table V.

VI. Related Works

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

Task Assignment in Spatial Crowdsourcing: Many studies have focused on how to assign tasks in spatial crowdsourcing. Kazemi, et al. [16] classifies spatial crowdsourcing into two modes: Worker Selected Tasks (WST) mode and Server Assigned Tasks (SAT) mode, where workers positively select tasks in WST and negatively wait for tasks to be assigned by server in SAT. Studies on SAT [16], [30] generally aim to maximize the number of assigned tasks by the server, while works on WST [8], [6] 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 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.

TABLE IV: Average communication overhead of a requester.

	TASK_RELEASING (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_RELEASING (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

Location Privacy Preserving in Spatial Crowdsourcing: To protect the location privacy in spatial crowdsourcing, existing works mainly leverage k-anonymity or differential privacy techniques. M. Gruteser et. al. [12] use 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. In [25], the authors propose 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 [29], [26] 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. In [18], the authors use homomorphic encryption to encrypt the locations of both workers and tasks, thus achieve 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 requires 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. W. Chang, et al. [5] proposes a distributed and self-organized crowdsourcing scheme within mobile social networks, in which the requester positively sends the task to the potential workers via multi-hop 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. Conclusions

In this paper, 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. Evaluations driven by real-world dataset demonstrated that despite of being fully distributed, PriTA achieves more efficient task assignment at the same level of location privacy compared with existing works.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61701216, Shenzhen Science, Technology and Innovation Commission Basic Research Project under Grant No. JCYJ20180507181527806, Guangdong Provincial Key Laboratory (Grant No. 2020B121201001) and "Guangdong Innovative and Entrepreneurial Research Team Program" (2016ZT06G587) and the "Shenzhen Sci-Tech Fund" (KYTDPT20181011104007).

References

[1] Google cloud. https://cloud.google.com/bigquery/public-data/.
[2] Lora. 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. *arXiv preprint arXiv:1212.1984*, 2012.
[4] C. Cai, Y. Zheng, and C. Wang. Leveraging crowdsensed data streams to discover and sell knowledge: A secure and efficient realization. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 589–599. IEEE, 2018.
[5] W. Chang and J. Wu. Progressive or conservative: Rationally allocate cooperative work in mobile social networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):2020–2035, 2014.
[6] M. H. Cheung, R. Southwell, F. Hou, and J. Huang. Distributed time-sensitive task selection in mobile crowdsensing. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 157–166. ACM, 2015.
[7] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*, pages 20–31. IEEE, 2012.
[8] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems*, pages 324–333. ACM, 2013.
[9] D. Deng, C. Shahabi, and L. Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 21. ACM, 2015.
[10] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
[11] C. Gentry and D. Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford University Stanford, 2009.
[12] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.
[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 Transactions on Sensor Networks (TOSN)*, 11(4):55, 2015.
[14] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students, Forthcoming*, 2010.

[15] L. Kazemi and C. Shahabi. A privacy-aware framework for participatory sensing. *ACM SIGKDD Explorations Newsletter*, 13(1):43–51, 2011.

[16] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th international conference on advances in geographic information systems*, pages 189–198. ACM, 2012.

[17] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[18] B. Liu, L. Chen, X. Zhu, Y. Zhang, C. Zhang, and W. Qiu. Protecting location privacy in spatial crowdsourcing using encrypted data. *Advances in Database Technology-EDBT*, 2017.

[19] Y. Liu, W. Wei, A. Sun, and C. Miao. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 739–748. ACM, 2014.

[20] I. Lykourentzou, S. Wang, R. E. Kraut, and S. P. Dow. Team dating: A self-organized team formation strategy for collaborative crowdsourcing. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1243–1249. ACM, 2016.

[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 *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[22] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter. Mobile crowd sensing in space weather monitoring: the mahali project. *IEEE Communications Magazine*, 52(8):22–28, 2014.

[23] L. Pu, X. Chen, J. Xu, and X. Fu. Crowdlet: Optimal worker recruitment for self-organized mobile crowdsourcing. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[24] L. Pu, X. Chen, J. Xu, and X. Fu. Crowd foraging: A qos-oriented self-organized mobile crowdsourcing framework over opportunistic networks. *IEEE Journal on Selected Areas in Communications*, 35(4):848–862, 2017.

[25] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment*, 7(10):919–930, 2014.

[26] H. To, C. Shahabi, and L. Xiong. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 833–844. IEEE, 2018.

[27] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *2016 IEEE 32Nd international conference on data engineering (ICDE)*, pages 49–60. IEEE, 2016.

[28] C. Wang, H. Liu, K.-L. Wright, B. Krishnamachari, and M. Annavaram. A privacy mechanism for mobile-based urban traffic monitoring. *Pervasive and Mobile Computing*, 20:1–12, 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 *Proceedings of the 26th International Conference on World Wide Web*, pages 627–636. International World Wide Web Conferences Steering Committee, 2017.

[30] X. Wang, R. Jia, X. Tian, and X. Gan. Dynamic task assignment in crowdsensing with location awareness and location diversity. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2420–2428. IEEE, 2018.