Privacy-Preserving Online Task Assignment in Spatial Crowdsourcing with Untrusted Server

Hien To,1 Cyrus Shahabi,2 Li Xiong3

1,2 University of Southern California, 2 Emory University
1hto,∗shahabi@usc.edu
3lxiong@emory.edu

Abstract—With spatial crowdsourcing (SC), requesters outsource their spatiotemporal tasks (tasks associated with location and time) to a set of workers, who will perform the tasks by physically traveling to the tasks’ locations. However, current solutions require the locations of the workers and/or the tasks to be disclosed to untrusted parties (SC server) for effective assignments of tasks to workers. In this paper we propose a framework for assigning tasks to workers in an online manner without compromising the location privacy of workers and tasks. We perturb the locations of both tasks and workers based on geo-indistinguishability and then devise techniques to quantify the probability of reachability between a task and a worker, given their perturbed locations. We investigate both analytical and empirical models for quantifying the worker-task pair reachability and propose task assignment strategies that strike a balance among various metrics such as the number of completed tasks, worker travel distance and system overhead. Extensive experiments on real-world datasets show that our proposed techniques result in minimal disclosure of task locations and no disclosure of worker locations without significantly sacrificing the total number of assigned tasks.

I. INTRODUCTION

The increase in computational and communication performance of mobile devices, coupled with the advances in sensor technology, leads to an exponential growth in data collection and sharing by smartphones. Exploiting mobility of such a large volume of potential users, a new mechanism for efficient and scalable data collection has emerged, namely, spatial crowdsourcing (SC) [12]. SC has numerous applications in domains such as environmental sensing, smart cities, journalism, and crisis response. With SC, requesters and workers typically register with a crowdsourcing server that acts as a broker between parties, and often also plays a role in how tasks are assigned to workers. A requester issues one or more tasks to the server (i.e., the platform). The server then assigns the task to a worker. We refer to this phase as tasking (or task assignment).

Several studies (e.g., [12], [24]) focus on effective tasking by maximizing the number of assigned tasks while minimizing workers travel distances, for which they require workers to reveal their locations and requesters to disclose their tasks’ locations to the server. We argue that to enable effective tasking, the server does not have to know the exact locations of the workers and the tasks because a task can be matched to a nearby worker as long as their proximity is known. However, once the worker agrees to complete the task, he must travel to the task’s location, perform it, and report the result to the server. Obviously, at this phase, referred to as reporting, the disclosures of the task’s location to the assigned worker and vice versa are usually unavoidable. Thus, privacy during the reporting phase is less critical and beyond the scope of this paper; instead, we focus on privacy protection during the tasking phase.

Privacy-preserving task assignment in SC has been an active area of research in recent years. Existing studies have two major drawbacks. First, they generally focus only on protecting location privacy of workers [21], [17] and assume that task locations are public. However, task locations should be secure during tasking since they can be sensitive. For example, the task locations can indirectly reveal requesters’ location, i.e., requesters often post tasks in the proximity of their locations [22]. Second, existing studies often assume a trusted entity to sanitize the location data [11], [17], [21]. This is not always the case in all applications of SC as there is no explicit trust relationship between any two parties (e.g., requester and worker). Hence, we assume a broader privacy setting where all SC parties could be curious but not malicious, and aim to protect location privacy of both workers and tasks during the tasking phase without relying on any trusted entity.

To obscure the locations of the workers and the tasks from potentially untrustworthy entities (e.g., servers), we can apply location privacy mechanisms, such as cloaking [7], perturbation [2], [27], private information retrieval [5], and secure multi-party computation [6]. Among them, we adopt a recent perturbation technique, named geo-indistinguishability [2], for these reasons: it is a mathematically rigorous definition of location privacy, and it suits our privacy setting in that the mechanism for achieving geo-indistinguishability can be performed in real-time by smartphones of workers and requesters, without the need of any trusted anonymization entity.

The challenge is to accurately estimate the worker-task pair reachability given only perturbed (or noisy) locations of the workers and the tasks. Due to the location uncertainty, a nearby task may not be assigned to a worker because the perturbed location of the task is farther away. This may reduce the number of completed tasks. On the other hand, a task can be assigned to a worker whose location is far away because the perturbed location of the worker is close to the task. This may increase the worker travel distance.

More concretely, we first assume the online tasking strategy, which has been shown to be scalable and effective for tasking in SC [3], [24], [23]. Within online tasking, the set of workers is known up front, and each task, upon arrival, needs to be immediately matched to an available worker (i.e., one at
a time); once assigned, the worker becomes unavailable for assignment. The main objective is to assign as many tasks to the workers as possible during a given time interval. Consider the example of three online tasks in Figure 1. Each task arrives one-by-one in the order of \( t_1 \rightarrow t_2 \rightarrow t_3 \). Every worker corresponds to a specific geographical region (aka spatial region) represented by a circle where any enclosed task is considered reachable from (and thus can be assigned to) the worker. The optimal assignment in this example is to match \( t_1 \) to \( w_2 \), \( t_2 \) to \( w_3 \) and \( t_3 \) to \( w_1 \). However, if \( t_2 \) is assigned first to \( w_1 \), \( w_1 \) becomes unavailable and therefore \( t_3 \) remains unmatched, resulting in a local optimum. It is known that the optimal algorithm for this problem (in terms of maximizing the number of assigned tasks\(^1\)) is Ranking [10], which selects a worker that is reachable to a task based on a random rank. However, Ranking may no longer generate the optimal result when it only has access to the perturbed locations of workers and tasks since the reachability between workers and tasks cannot be exactly determined.

![Fig. 1: The tasking phase of spatial crowdsourcing.](image)

To address this challenge of location uncertainty, we partition the online tasking setting into a three-stage privacy-aware framework, dubbed SCGuard (see Table I). SCGuard involves different parties at each stage of the task assignment to ensure effective tasking. This is achieved through revealing users’ locations gradually, only if needed, as the tasking proceeds from one stage to the other. In the first stage, the server recommends a small set of nearby candidate workers for a given task (without knowing the exact locations of either party); the server then forwards the perturbed locations of these workers to the task’s requester. On receipt of these perturbed locations, in the second stage, the requester identifies the most likely reachable worker and sends the task location to this worker. Once receiving the exact location of the task, in the final stage, the selected worker accepts the task if it is enclosed within his spatial region; otherwise, he rejects the task. Hence, it is possible for a candidate worker to learn the exact location of the task even when the worker is not reachable to the task; we quantify this disclosure later. The last two stages may repeat until either the task is assigned or no candidate worker is left.

A key component in each stage of the above framework is to determine whether a task is reachable from a worker. We first devise a baseline solution by assuming the perturbed locations of tasks and workers as their actual locations. We call this baseline the “oblivious” technique, as it is oblivious to the fact that the locations are perturbed and not real. Obviously, the utility of this approach is very low. To improve the utility of the oblivious approach, we propose analytical and empirical models to quantify the probability of reachability between a task and a worker in each stage of SCGuard. Thereafter, we introduce a probability-based solution that improves the baseline in several metrics, including a higher number of assigned tasks, smaller worker travel cost and lower disclosure of location information.

The contributions of this paper are as follows.

1) We propose SCGuard, a privacy-aware framework that enables workers and requesters to participate in SC without compromising their location privacy. To the best of our knowledge, this is the first work designed to protect the privacy of both parties in SC without assuming any trusted entity.

2) We propose the analytical and empirical models to quantify the worker-task pair reachability in every stage of SCGuard, based on which of the probabilistic tasking algorithm is introduced.

3) We conduct an extensive set of experiments on a real-world dataset, showing three main results. First, the analytical model performs as well as the empirical model without relying on precomputation on past or synthetic data. Second, our probabilistic tasking algorithm is superior to the baseline in all key metrics, including a higher number of assigned tasks (×2), smaller worker travel cost (2/3) and lower disclosure of task location (/500), with only a slight increase in system overhead (20%). Third, SCGuard is able to protect location privacy of both workers and requesters without significantly compromising the key metrics of the SC system.

The remainder of this paper is organized as follows. Section II presents the necessary background, Section III introduces SCGuard, and Section IV details our proposed solution. Experimental results are presented in Section V, followed by a survey of related work in Section VI and discussion in Section VII. We conclude the paper in Section VIII.

### II. BACKGROUND ON GEO-INDISTINGUISHABILITY

**Geo-indistinguishability (Geo-I) [2]** is a notion of location privacy based on differential privacy (DP)—the de facto standard in data privacy, thanks to its strong protection guarantees rooted in statistical analysis. Similar to DP, Geo-I is a semantic model which provides protection against adversaries with background information. A mechanism provides \( \epsilon \)-geo-indistinguishability if any two locations at distance at most \( r \) produce observations with “similar” distributions bounded by \( \epsilon \). We refer to this privacy guarantee as \((\epsilon, r)\)-Geo-I. The parameter \( \epsilon \) is the level of privacy, which specifies the amount

<table>
<thead>
<tr>
<th>Stage</th>
<th>By</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Server</td>
<td>Uncertain task location, uncertain workers’ locations</td>
<td>A set of candidate workers for a given task</td>
</tr>
<tr>
<td>2</td>
<td>Requester</td>
<td>The uncertain candidate workers recommended by the server</td>
<td>The worker of the highest rank</td>
</tr>
<tr>
<td>3</td>
<td>Worker</td>
<td>Exact task location</td>
<td>Accept the task or not</td>
</tr>
</tbody>
</table>

**TABLE I: Three stages of the privacy-aware framework.**
of protection required, with smaller values corresponding to stricter privacy protection. The parameter $r$ is the radius of concern within which privacy is guaranteed. This means that an adversary cannot distinguish locations which are at most $r$ distance away. Geo-I is formally defined as follows.

**Definition 1 ((ε, r)-Geo-indistinguishability [2]):** Let $X$ be a set of exact locations. A mechanism $A$ satisfies $(\epsilon, r)$-geo-indistinguishability iff for all $x, x' \in X$ such that $d(x, x') \leq r$:

$$d_P(A(x), A(x')) \leq \epsilon d(x, x')$$

$d(x, x')$ is the Euclidean distance between $x$ and $x'$ while $d_P(A(x), A(x'))$ is the multiplicative distance between the two distributions produced by $x$ and $x'$, correspondingly. Note that we use the constrained version of Geo-I (i.e., $d(x, x') \leq r$), which forces the corresponding distributions to be at most $\epsilon r$ distance from each other.

To preserve privacy, random noise is injected into each location such that, by observing a perturbed location, an adversary cannot infer the true location among all locations within radius $r$. Particularly, one way to achieve Geo-I is to generate random point $z$ (from actual point $x \in X$) according to planar Laplace distribution. This function ensures that the probability of reporting a point in a certain (infinitesimal) area around $z$, when the actual locations are $x$ and $x'$, differs at most by a multiplicative factor $e^{-\epsilon d(x,x')}$. Hence, pdf of the noise-adding mechanism is called planar Laplacian centered at $x$.

$$d_w(x)(z) = \frac{e^{-\epsilon d(x,z)}}{2\pi}$$

where $\frac{e^{-\epsilon d(x,z)}}{2\pi}$ is a normalization factor.

**III. THE SCGUARD PRIVACY FRAMEWORK**

In this section we first present a framework for the tasking phase of SC without compromising the privacy of the locations of the individuals (both workers and requesters). We then identify potential privacy threats from the adversaries (server, requester and worker) and present countermeasures to prevent such threats from occurring. Finally, we introduce the performance metrics to evaluate and compare our different privacy-preserving algorithms.

**A. System Model and Assumptions**

We first define the notions of spatial task and worker. A **spatial task** $t$ is required to be answered at a particular location $l_t$. This means task $t$ can be answered by a human only if he is physically located at the task’s location $l_t$. A **worker**, denoted by $w$, is a carrier of a mobile device who volunteers for one of the spatial tasks. Each worker has a location $l_w$ and a spatial region $R_w$, wherein the worker can accept spatial tasks. $R_w$ is represented by a circular region centered at the worker’s location. Hence, $R_w$ also refers to a reachable distance of the worker—task $t$ is reachable from worker $w$ iff $d(w, t) \leq R_w$.

During tasking, we assume that each worker performs a single task, and all workers perform every task correctly so that every task needs to be assigned to one and only one worker. We also assume that tasks will not expire during the assignment period.

We focus on **online assignment** where the set of workers $W$ is given in advance while each task in the task set $T$ arrives online (i.e., one at a time). Without privacy protection, the optimal algorithm in terms of maximizing the number of assigned tasks is Ranking [10]. The algorithm associates each worker with a random number (or rank) so that each task, upon its arrival, is assigned to an unmatched reachable worker of the highest rank\(^2\). This algorithm can entirely run on the server. However, with privacy protection, locations of workers and tasks become uncertain, which complicates the task assignment. We propose a privacy-aware framework, named SCGuard, for online tasking that involves three distinct stages as follows (see Figure 2).

In the first stage, to ensure privacy, each worker $w$ perturbs his location $l_w$ with a specified privacy level $(\epsilon, r)$ according to the Geo-I mechanism and sends the noisy location $l_w'$ together with his reachable distance $R_w$ to the server. Upon the arrival of task $t$, the requester of $t$ perturbs its location $l_t$ with different privacy level $(\epsilon, r)$-Geo-I and sends the perturbed one $l_t'$ to the server. The main role of the server is to identify a set of candidate workers for the task and then forward their information (i.e., $l_w'$ and $R_w$) to the requester. We refer to the first stage as uncertain-to-uncertain (U2U) because the locations of both workers and tasks are uncertain to the server—the server knows only perturbed locations of workers and tasks. There is no location disclosure at this stage because locations of both task and worker are hidden from all three adversaries.

The requester, on receipt of the information of the workers, conducts the second stage of SCGuard without any communication with the server. We isolate this stage from the server to ensure no disclosure to the server. During this stage, the requester sends her task location to the candidate worker $w$ who is most likely reachable within distance $R_w$. We call this stage uncertain-to-exact (U2E) because the requester knows the exact location of her task and needs to make a decision as to whether a particular candidate worker is reachable to the task given the worker’s perturbed location. This process repeats until the task is assigned or no candidate worker is left. We discuss three approaches for selecting a candidate worker in Section IV.

The third and final stage, exact-to-exact (E2E), is performed by the selected candidate workers. At this point, the requester releases the actual location of the task to the worker, resulting in some disclosure. The worker can verify if the task is reachable by comparing his distance to the task, i.e., $d(w, t) \leq R_w$. If the task is reachable, the worker performs the task, and we consider that the task is assigned; otherwise, he rejects the task. Table II summarizes the notations used in this paper.

Our approach for task dissemination in U2E is sequential, meaning that the requester sends her task location to one worker in the candidate set at a time. However, one may argue for a parallel approach, where the server first sends the perturbed task location to all workers in the candidate set.

\(^2\)We also consider distance-based ranking for travel cost optimization in Section IV-A1
act maliciously if they want to get the tasks done. We briefly discuss the potential malicious behaviors of different parties and how to mitigate them in Section VII. In what follows, we analyze potential disclosure of location information during each stage of the protocol.

During U2U the server takes as input the perturbed locations of both workers and tasks to perform effective task assignment; the amount of disclosure to the server is strictly controlled by a given level of privacy according to the Geo-I mechanism. Furthermore, locations of both workers and tasks are protected from the server in all stages of the protocol. This is because the server does not participate in U2E and E2E. Specifically, the server recommends a set of candidate workers to a requester so that they can establish a direct communication channel among themselves. From the perspective of the server, the requester and the candidate workers autonomously decide on whether to accept the recommendation from the server.

During U2E and E2E, special emphasis is on limiting the disclosure between the workers and the requesters since they may learn each other’s exact locations during the course of the protocol. From the viewpoint of a curious requester, locations of the candidate workers are not revealed to the requester during U2E. However, the requester may learn the proximity of the worker to the task (but not the exact location of the worker) by observing the response of a candidate worker during U2E (reachable or not). From the viewpoint of a curious worker, due to the uncertainty of workers’ locations during U2E, a task location can be disclosed to multiple candidate workers in E2E before being assigned. This kind of disclosure among requesters and workers, named false hit, is quantified in Section III-C.

After completing an assigned task, the worker reports the result of the task either to the server for quality control, payment, etc. or directly to the requesters to limit the disclosure to the server even further. Nevertheless, privacy threats during reporting are beyond the scope of this protocol.

C. Performance Metrics

Protecting locations of both workers and tasks may reduce the effectiveness and efficiency of task assignment. Due to the noise introduced by Geo-I, a worker-task match observed as reachable in the noisy domain may be unreachable in the actual domain, or vice versa. Both cases may result in tasks remaining unassigned. Thus, to find a reachable worker for a task (i.e., a valid match), multiple messages may need to be sent between the requester and workers, which increases the amount of location disclosure.

To measure these, we introduce the following end-to-end performance metrics:

1. **Utility.** The performance of SCGuard is measured by the number of assigned tasks. Due to data uncertainty, the server may incorrectly identify candidate workers for a task. The challenge is to obtain a high number of assigned tasks in the presence of uncertainty.

2. **Travel Cost.** With imprecise locations, the server is no longer able to accurately estimate the distances between workers and tasks.

### TABLE II: Summary of notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w, t, W, T$</td>
<td>a worker, a task, a worker set and a task set</td>
</tr>
<tr>
<td>$w', t', W, T$</td>
<td>a worker and a task after perturbation</td>
</tr>
<tr>
<td>$d_{w}^{l}$</td>
<td>reachable distance that $w$ is willing to travel</td>
</tr>
<tr>
<td>$l_{w}, l_{t}$</td>
<td>actual locations of worker $w$ and task $t$</td>
</tr>
<tr>
<td>$d_{w}, d_{t}$</td>
<td>noisy (perturbed or observed) locations of $w$ and $t$</td>
</tr>
<tr>
<td>$d_{w}, d_{t}$</td>
<td>the distance between $l_{w}$ and $l_{t}$</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>parameters that specify the privacy level of Geo-I</td>
</tr>
<tr>
<td>$\epsilon, \rho$</td>
<td>a set of candidate workers to $t$</td>
</tr>
<tr>
<td>$\alpha, \beta$</td>
<td>reachability thresholds during U2U and U2E</td>
</tr>
</tbody>
</table>

Fig. 2: SCGuard: privacy-aware framework for spatial crowdsourcing.

Subsequently, the workers simultaneously and independently evaluate whether the task is reachable or not. If so, they send their locations to the requester, who performs the final stage (E2E). This parallel approach may be more efficient but can potentially result in more disclosures. This is because multiple candidate workers may find the task reachable and together send their locations to the requester. Hence, we do not consider this parallel optimization any further.

Another alternative design choice for the U2E stage is for the server to rank the candidate workers rather than the requester. In this case, the candidate workers receive the perturbed task location from the server and return their likelihoods to perform the task to the server. Thereafter, the server matches the task to the worker who most likely will perform the task. This scenario seems to be more efficient in terms of communication cost than the proposed U2E stage; however, the server may be able to learn extra information about the task by observing the responses of one or multiple candidate workers during U2E (reachable or not). The challenge is that these responses can be correlated since they are computed from the same task. Hence, to ensure the same privacy level, the privacy guarantee needs to be extended for a location set [2], which drastically reduces the utility of the privacy mechanism. We further discuss this issue in Section VII. Thus, we do not consider this design option further.

### B. Adversary Model

We assume the semi-honest model, which has two core assumptions. First, all participating entities (server, worker, requester) are curious but not malicious. This means that each entity may learn from what is exposed during the three stages of SCGuard, but they comply with the protocol. Second, the entities do not collude with each other to gain information about the third. These assumptions are realistic in our setting since there is little incentive for the requesters and workers to
workers and tasks. Hence, workers may have to travel long distances to tasks. The challenge is to keep the worker travel distance low, even when exact locations are unknown.

(3) System Overhead. Dealing with imprecise locations increases the complexity of assignment algorithms, which poses scalability problems. A significant metric to measure overhead is the size of the worker candidate set for a task. This number represents both the communication overhead (server sends the candidate set to the requester) and computational overhead (requester ranks the workers in the candidate set based on certain criteria).

In addition, we also report per-stage metrics to better understand the performance as well as potential location disclosure of each stage of SCGuard.

(4) Precision/Recall (U2U). For each task we measure the ratio of the candidate workers who are reachable (precision) and the ratio of the reachable workers in the candidate set (recall).

(5) False Hit/False Dismissal (U2E). A false hit is a privacy leak, occurring when a requester estimates an unreachable worker as reachable, measured by the number of times a task location is revealed to a candidate worker who eventually does not perform the task. A false dismissal occurs when a requester misses a reachable worker.

IV. ONLINE TASK ASSIGNMENT

A. Baseline Solution

In this section we show why existing solutions to the online tasking problem in the non-private setting may not be effective in the private setting. Hence, we introduce a baseline algorithm for the private setting.

1) Ranking Algorithm: With the online assignment, workers are known and tasks arrive online (one-by-one). Upon arrival, each task needs to be immediately matched to an unmatched worker; the goal is to maximize the number of assigned tasks. A well-known solution to the online assignment problem is the Ranking algorithm [10]. Ranking randomly permutes the workers and assigns a random priority (or rank) to them. When a task arrives, it is matched to a worker who is reachable to the task and has the highest rank. The expected size of matching obtained by Ranking is at least \((1 - 1/e)\frac{T}{T} = 0.63\frac{T}{T}\), where \(T\) is the total number of tasks. This result is optimal in the non-private setting [10]. In other words, the competitiveness of any online bipartite matching algorithm is bounded above by 0.63.

However, the ranking algorithm may not work well in the privacy setting. The reason is that a reachable worker-task pair can be observed as unreachable after perturbation, and vice versa. Figure 3b shows the reachability graph of the workers and tasks given their layout shown in Figure 1. The reachable pair \((w_1, t_3)\) becomes unreachable in the noisy domain, while unreachable pairs \((w_3, t_1)\) and \((w_3, t_3)\) become reachable. In addition, Figure 3c shows an optimal matching in the noisy domain: \((w_2', t_1'), (w_4', t_2)\) and \((w_3', t_3')\). Nonetheless, the assignment is actually not optimal because \((w_3', t_3')\) is unreachable in the actual domain.

Algorithm 1: Oblivious Algorithm (Baseline)

1: Input: \(W, T, R_{wi}, \epsilon, r\) (refer to the notations in Table II)
2: Output: a set of valid worker-task matches
3: Perturb locations of workers and tasks using Geo-I [2]:
4: \(l_{w_i} \rightarrow l_{w_i}', l_{t_j} \rightarrow l_{t_j}'\)
5: For \(t_j \in T\) do: {assign it to the highest-rank worker}
6: U2U: Server identifies candidate workers \(N_j\) for \(t_j\):
7: \(N_j = \{w_i : d(w_i', t_j') \leq R_{wi}\}\)
8: If \(N_j = \emptyset\) : \(t_j\) remains unassigned; go to Line 5
9: Server forwards candidate workers \(N_j\) to \(t_j\)'s requester
10: U2E: Requester matches \(t_j\) to \(w_{\text{max}}\)
11: \(w_{\text{max}} = \text{argmax}\{\text{Rank}(w_i) : w_i \in N_j\}\)
12: For each \(N_j\): \(0 < \frac{1}{\pi(w_i, t_j)} < \ell\) or \(t_j\)
13: Requester sends exact task location \(l_{t_j}\) to \(w_{\text{max}}\)
14: E2E: Worker \(w_{\text{max}}\) checks if \(d(w_{\text{max}}, t_j) \leq R_{w_{\text{max}}} :\)
15: If so, match \((t_j, w_{\text{max}})\) is a valid assignment
16: Update \(W\): \(W = W \setminus \{w_{\text{max}}\}\)
17: Otherwise, update \(N_j\): \(N_j = N_j \setminus \{w_{\text{max}}\}\), go to Line 10

2) Baseline Algorithm: Algorithm 1 presents the oblivious algorithm, which considers observed locations as true ones. First, locations of workers and requesters’ tasks are locally perturbed according to a specified privacy level \((\epsilon, r)\)-Geo-I [2] (Lines 3–4). During U2U, the server identifies candidate workers for each task such that the observed location of the task is reachable from the observed locations of the workers (Line 7). The server then forwards the candidate workers to the requester, who performs the U2E phase (Line 9). As mentioned in Section III-A, the reason U2E is performed by the requester instead of the server is that multiple candidate workers may need to be selected until a reachable worker for the task is found. These back-and-forth communications with the workers need to be secure from the server to ensure privacy protection. During U2E, the requester—once receiving the candidate workers from the server—ranks those workers with respect to certain criteria, such as a random rank [10] or the worker-task distance (aka the nearest-neighbor strategy) (Line 12). Thereafter, the task is matched to the worker of the highest rank (Line 10), who subsequently receives the actual location of the task (Line 13). During E2E, the selected candidate worker confirms whether the task is actually reachable (Line 14). If so, this is a valid assignment. Otherwise, the worker rejects the task so that the requester can send the task to the candidate worker of the second highest rank (Line 17). This matching process (U2E and E2E) continues until either the task is assigned or no candidate worker is left (Line 8).
This best-effort strategy clearly provides more opportunity for the task to be assigned, but at the expense of disclosing its location to more workers. This trade-off is illustrated in the following example.

In Figure 3c, when task $t_1$ arrives, $w_1$ and $w_2$ are candidate workers. Location $l_{t_1}$ is sent to $w_1$ because it has a higher rank. $w_1$ finds $t_1$ unreachable and declines $t_1$, introducing a false hit. Subsequently, $l_{t_1}$ is sent to the next candidate worker of the highest rank, $w_2$, $w_2$ finds $t_1$ reachable and performs $t_1$. Next, $t_2$ arrives, and has two candidates, $w_1$ and $w_3$, $w_1$ performs $t_2$ as they are reachable. Finally, $t_3$ arrives and is matched to candidate $w_3$; nevertheless, $w_3$ finds $t_3$ unreachable and rejects $t_3$. In sum, the two assigned tasks $t_1$ and $t_2$ are performed by $w_2$ and $w_1$, respectively. Here, the two false hits include $w_1$ knowing $l_{t_1}$ and $w_3$ knowing $l_{t_3}$.

We emphasize that the oblivious algorithm guarantees $(\epsilon, r)$-Geo-I to both workers and tasks from the untrusted server. The reason is that the process of finding the candidate workers for a task (after location perturbation) is considered post-processing, which does not affect the privacy guarantees of differentially private mechanisms [15].

B. Quantifying Worker-Task Pair Reachability

An issue with the oblivious algorithm is that the reachability between a worker and a task is a binary decision based on the perturbed locations, reachable or not, which does not utilize the planar Laplace distribution of the perturbed locations (see Section II). As a result, the baseline solution may include non-reachable workers and miss reachable workers in the candidate set, e.g., $(w_1, t_3)$ becomes unreachable in Figure 3. Therefore, the worker-task pair reachability should be quantified by the probability of reachability between the worker and the task. This allows a requestor to accurately compare the reachability to her task from multiple candidate workers.

The objective is to compute the reachability probability of a worker-task pair given their observed distance, i.e., $\Pr\left(d(w, t) \leq R_w\right)$ (refer to the notations in Table II). We present two approaches to this problem: one is based on approximation analysis (for efficient computation), while another is based on empirical results (requires precomputation on synthetic or historic data).

1) Analytical Approach: An intuitive approach is to derive the distribution (pdf) of the actual distances between the locations of workers and tasks $d(w, t)$ given the perturbed locations $l_w, l_t$. Recall that the locations are perturbed using the planar Laplace distribution. Once the pdf is derived, the reachability probability can be computed efficiently with numerical libraries, such as Python, R and MATLAB.

The problem of finding the pdf of the distance between two uncertain points is related to a family of line picking problems, such as disk line picking. The disk line picking problem is to choose two points at random in a unit disk and find the distribution of the distances between the two points. Such problems have closed form solutions. However, in our setting, the two points are drawn from a planar Laplace distribution with different centers rather than uniformly distributed on the same disk. This makes our problem more challenging, and a closed form solution may not exist.

Because the planar Laplace distribution is difficult to analyze, we propose a two-phase method to parameterize the pdf: 1) approximating the planar Laplace distribution by a bivariate normal distribution (BND), and 2) deriving the closed form solution to the pdf of $d(w, t)$.

**Approximated BND:** According to [2], the pdf of the noise-adding mechanism follows a planar Laplace distribution (see Section II) with center at the true location. We approximate the planar Laplace distribution by a BND with the same mean and variance. These are the first two moments, which represent the most important information of a distribution. Since the planar Laplace distribution is symmetric to its center, the approximated BND should be symmetric to the same center (i.e., circular bivariate normal distribution). Subsequently, the approximated distribution is $BND(\mu, \Sigma)$, where $\mu$ is a 2-dimensional mean vector $(w_x, w_y)^3$ representing the worker location, and $\Sigma$ is a diagonal variance matrix $\left[\sigma_x^2, \sigma_y^2\right]$, where $\sigma = \sqrt{2/\pi}$ is the standard deviation of the planar Laplace distribution $(\epsilon$ and $r$ are privacy parameters).

Consequently, the distribution of the distance between the perturbed location and its original location is approximated by a normal distribution $N(0, \sigma^2/\epsilon^2)$. This means that when the perturbed (observed) location is known, the original location is approximated by $BND(\mu, \Sigma)$, centering at the observed point with mean $\mu = (w'_x, w'_y)$ and variance $\Sigma = \left[\frac{x^2}{\sigma_x^2}, \frac{y^2}{\sigma_y^2}\right]$.

Given observed $w'$ and $t'$, we can approximate the original location of $w$ and $t$ both with $BND$. Next, we derive the pdf's of $d(w, t)$ for both U2U and U2E stages.

**PDF of $d(w, t)$ for U2U:** In the U2U stage, given the uncertain locations $l_w$ and $l_t$, our goal is to estimate the pdf of $d(w, t)$—the actual distance between the original locations $l_w$ and $l_t$ (see Figure 4a). As presented, $l_w$ is approximated by $BND(\mu_w, \Sigma_w)$, centering at the observed worker location $l_w'$, and $l_t$ is approximated by $BND(\mu_t, \Sigma_t)$, centering at the observed task location $l_t'$. We have $d = \sqrt{\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}}$, where $z$ equals to the difference in vector space $z = l_w - l_t$, which follows $BND(\mu_w - \mu_t, \Sigma_w + \Sigma_t)$. We approximate the distribution of $d^2 = z_x^2 + z_y^2$, and then use the following lemma to derive the reachability probability of $d$: $Pr(d \leq R_w)$.

**Lemma 1:** $Pr(\sqrt{X} \leq \sqrt{C})$ is equal to $Pr(X \leq C)$, where $X$ is non-negative random variable and $C$ is a non-negative constant.

**Proof:** This is true because $\sqrt{X} \leq \sqrt{C} \iff X \leq C$ for $X, C \geq 0$. This means that the set of events $\{A \in \Omega : \sqrt{X} \leq \sqrt{C}\}$ equals the set $\{A' \in \Omega : X \leq C\}$, so are their probabilities $Pr(\sqrt{X} \leq \sqrt{C})$ and $Pr(X \leq C)$.

Applying the lemma to our context where $X = d^2$ and $C = R_w^2$, we have the reachability probability of a worker-task pair $Pr(d \leq R_w) = Pr(d^2 \leq R_w^2)$.

Since $d^2$ has a quadratic form in the bivariate random variable $z$, the moment-generating function (mgf) of $d^2$ has

\[1\text{The subscripts } x \text{ and } y \text{ represent the corresponding axis.}\]
the following form [14]:

\[ M(e^{1D}) = e^{tr\Sigma^2} \prod_{j=1}^{2} (1 - 2\lambda_j t)^{-1/2} \]

where \( \lambda_j \) is the linear function of \( \mu \) (\( b_1 = \mu_{w_s} - \mu_{t}, \)
\( b_2 = \mu_{w_s} - \mu_{w} \)) and \( \lambda_j \) are the eigenvalues of \( \Sigma = \Sigma_{w} + \Sigma_{t} \).

Given the mgf of \( d^2 \), mean and variance of \( d^2 \) can be calculated as follows. Mean \( \mu \) equals to the first derivative of the mgf at \( t = 0 \): \( \mu = E[D] = M'_1(0) \). Variance \( \sigma^2 \) can be computed by evaluating the second derivative of the mgf at \( t = 0 \): \( \sigma^2 = E[D^2] - (E[D])^2 = M''_1(0) - (M'_1(0))^2 \). Consequently, we approximate the pdf of \( d^2 \) by a normal distribution \( \mathcal{N}(\mu, \sigma^2) \), where mean and variance can be computed efficiently using the built-in python library Scipy.

For simplicity, we derive mean and variance for a special case where both the worker and the task’s requester set the same privacy level \((\epsilon, \tau)\). In such case, the eigenvalues are equal, \( \nu = 4\tau^2/\epsilon^2 \) and the mgf can be derived as follows, where \( \nu \) is the observed worker-task distance \( \nu = d(w', t') = \sqrt{b_1^2 + b_2^2} \):

\[ M(e^{1D}) = e^{\frac{\nu^2}{2\lambda \tau}} \frac{1}{1 - 2\lambda t} \tag{2} \]

We derive the first and second derivatives: \( M'_1(0) = \lambda(2 + \nu^2) \)
and \( M''_1(0) = 8\lambda^{3} + 8\lambda^{2}\nu^{2} + \lambda^{2}\nu^{4} \); thus \( d^2 \) follows approximately a normal distribution with mean and variance: \( \mu = \lambda(2 + \nu^2) \) and \( \sigma^2 = 4\lambda^{2}(1 + \nu^2) \).

**PDF of \( d(w, t) \) for U2E:** In the U2E stage, given the true location of task \( l_t \) and the perturbed location of worker \( l_w \), we need to estimate the pdf of the actual distance \( d(w, t) \). When task location \( l_t \) is fixed and worker location \( l_w \) follows BND(\( \mu, \Sigma \)) centering at the observed worker location \( l_w \), the distance between \( w \) and \( t \) follows the Rice distribution [18] with parameters \((\nu, \sigma)\), \( \nu = d(w', t) \) is the distance from the task’s location \( l_t \) (i.e., the reference point) to the center of the approximated BND, \( l_w \) (see Figure 4b), \( \sigma \) is the scale parameter and equals the square root of the variance of the approximated BND, \( \sqrt{2\tau/\epsilon} \). The pdf of the Rice distribution is:

\[ f(x|\nu, \sigma) = \frac{x}{\sigma^2} e^{\frac{-(x^2 + \nu^2)}{2\sigma^2}} I_0(\frac{2\nu x}{\sigma^2}) \]

where \( I_0(\cdot) \) is the modified Bessel function of the first kind with order zero [1]. We used the Scipy library to efficiently compute the pdf of the Rice distribution.

2) **Empirical Approach:** The analytical approach above provides a fast but approximate way to compute the reachability probability. We present an empirical approach that computes the probability from synthetically generated or past data. We show the simulation for each stage of SCGuard as follows.

For U2U, we generate random locations for a large number of worker-task pairs in a certain region of interest (i.e., Beijing City). All generated locations are perturbed according to \((\epsilon, \tau)\)-Geo-I using random seeds. For each worker-task pair, both the actual distance \( d \) and the corresponding noisy distance \( d' \) are calculated. The noisy distances are grouped into disjoint

```
<table>
<thead>
<tr>
<th>Range</th>
<th>No. of Samples</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>0.9</td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>0.7</td>
</tr>
</tbody>
</table>
```

Figure 5b illustrates the pdf of the reachability probability when varying the observed distance \( d' \). Unlike the E2E stage where the worker-task pair reachability is a step function (binary model), the pdf values of U2U and U2E decrease linearly with the increase of \( d' \). We also observe that compared to U2E, U2U underestimates when \( d' \) is small but overestimates when \( d' \) is large. It is worth noting that our empirical approach for precomputing the worker-task pair reachability uses the synthetic location datasets, and therefore does not breach individual information. The precomputed reachability information is used to enhance the performance of task assignment but is not useful to an adversary in gaining information of any individual’s location. It is also possible to use either public datasets or past assignment data of completed tasks in the precomputation.
C. Probability-based Solution

Given the reachability probability, we present a probability-based algorithm that enhances Algorithm 1 at both U2U and U2E stages. The key improvement is to use the probabilistic model (either the analytical or the empirical approach), rather than the binary model, for quantifying reachability between a worker and a task.

1) Improvement to U2U: With Algorithm 1, a worker is a candidate for a task if its observed distance is upper-bounded by the reachable distance of the worker (Line 7). Hence, Line 7 of Algorithm 1 may introduce a false positive and false negative. Figure 6 shows the precision/recall scores by varying privacy guarantee $r$. Ensuring high recall is important because low recall means most reachable workers are not in the candidate set, likely resulting in large utility loss.

Increasing the recall score: Algorithm 1 can be updated to ensure high recall during the U2U stage as follows. The candidate workers are selected such that their probability of reachability to a task is greater than a given threshold $\alpha$, termed U2U threshold (see Line 7 of Algorithm 2). By decreasing $\alpha$, recall is higher but at the cost of lower precision, resulting in an increase of the ratio of unreachable workers in the candidate set. This may incur penalties in the later stage (U2E), such as higher system overhead. We will evaluate the impact of varying $\alpha$ in Section V.

**Algorithm 2: Probability-based Algorithm**

1. Input: $W, T, R_{w_i}, \epsilon, r, \alpha, \beta$ (refer to the notations in Table II)
2. Output: a set of valid worker-task matches
3. Perturb locations of workers and tasks using Geo-l [2]:
4. $l_{w_i} \rightarrow l'_{w_i}$, $l_{t_j} \rightarrow l'_{t_j}$
5. For $t_j \in T$ do: {assign it to the highest-rank worker}
6. U2U: Server identifies candidate workers $N_t$ for $t_j$:
7. $N_t = \{ w_i : Pr(d(w_i, t_j) \leq R_w | d(w_i', t_j') \geq \alpha) \}$
8. If $N_t = \emptyset$: $t_j$ remains unassigned; go to Line 5.
9. Server forwards candidate workers $N_t$ to $t_j$’s requester
10. U2E: Requester matches $t_j$ to $w_{\text{max}}$, where
11. $w_{\text{max}} = \text{argmax}\{ \text{Rank}(w_i) : w_i \in N_t \}$
12. $\text{Rank}(w_i) = Pr(d(w, t) \leq R_w | d(w, t') \geq \beta)$
13. If $\text{Rank}(w_{\text{max}}) < \beta$: go to Line 5
14. Requester sends exact task location $l_t$ to $w_{\text{max}}$
15. E2E: Worker $w_{\text{max}}$ checks if $d(w_{\text{max}}, t_j) \leq R_{w_{\text{max}}}$:
16. If so, match $t_j, w_{\text{max}}$ is a valid assignment
17. Update $W \leftarrow W - w_{\text{max}}$; go to Line 5
18. Otherwise, update $N_t = N_t - w_{\text{max}}$; go to Line 10

Improving runtime performance: Line 7 of Algorithm 2 linearly checks all workers, which may be time-consuming for a large number of workers. Hence, we propose a technique to quickly prune workers who are most likely not reachable to a particular task. The technique has two steps.

First, each worker (or task) corresponds to a disk with radius $r_R$, centering at the perturbed location such that the actual location is within the disk with probability at least $\gamma$ (see Section 5 in [2] on how to compute $r_R$). The disks are depicted by the solid circles in Figure 7, denoted as $\text{disk}(l'_{w_i}, r_R)$ and $\text{disk}(l'_{t_j}, r_R)$. Hence, the outer dashed circle represents a region $\text{disk}(l'_{w_i}, r_R + R_w)$ that encloses any point in the worker’s spatial region $R_w$ with probability at least $\gamma$. Subsequently, we approximate the worker by the larger minimum bounding box (MBR) and the task by the smaller MBR in Figure 7.

In the second step, we build an index to quickly prune far-away workers from a task without a full linear scan by applying existing techniques in the uncertain database field [19], [4]. For example, the fuzzy search technique in [19] is suitable when both data and query (workers and tasks in our context) are represented by rectangles. Applying this pruning technique on the approximated MBRs of workers and tasks gives us a lower bound of $\gamma$ on the reachability probability during U2U. This is because a worker and a task are not reachable from each other if their MBRs do not overlap. Note that pruning workers during U2U makes sense since it is the most time-consuming stage of SCGuard.

2) Improvement to U2E: The oblivious algorithm ranks candidate workers by either their distances to a task or a random rank associated with each worker (Line 12 of Algorithm 1). However, the nearest (or a random) worker may not be reachable to the task. This is because the reachability probability between worker $w$ and task $t$ not only depends on their distance $d(w, t)$ but also on the worker’s spatial region $R_w$. To illustrate, in Figure 3b $t_2'$ is closer to $w_1'$ than to $w_3'$; however, $t_2'$ is more likely to be reachable from $w_3'$ than from $w_1'$ because $R_{w_3}$ is much greater than $R_{w_1}$.

**Ranking candidate workers probabilistically:** We argue that the candidate workers should be ranked based on their reachability to a task. The reason for this is to reduce the number of workers who are being notified of the task location, which results in a small number of false hits (task location disclosures) while reducing travel cost.

Therefore, we modify the U2E stage of Algorithm 1 to capture the probability of reachability, quantified in Section IV-B. Particularly, the requester evaluates the reachability probability between all candidate workers and her task. Subsequently, each candidate worker is associated with a rank $\text{Rank}(w_i)$ that equals the corresponding probability of reachability (see Line 12 of Algorithm 2). The task is matched to the candidate worker of the highest rank $w_{\text{max}}$ (Line 10).

**Reducing false hits:** Algorithm 1 may disclose a task’s location to a large number of candidate workers before the task gets assigned. In the worst case, the task’s location can be disclosed to all candidate workers, yet none of them are reachable to the task. Thus, to reduce the number of false hits (or privacy disclosures) during U2E, we propose a thresholding...
heuristic as follows. A requester cancels her task when the reachability probability from the highest-rank worker $w_{\text{max}}$ is smaller than a given threshold $\beta$ (Line 13 of Algorithm 2). Subsequently, the requester does not send her task’s location to more candidate workers. The choice of $\beta$ affects the number of assigned tasks, false hits and false dismissals. The smaller $\beta$, the more likely the requester sends the actual location of her task to the candidate workers. This results in the task having a higher chance of being assigned, but at the expense of a higher number of false hits. On the other hand, high $\beta$ may lead to false dismissals. We empirically find a good value of $\beta$ in Section V.

V. PERFORMANCE EVALUATION

We present the experimental setup in Section V-A, followed by results in Section V-B.

A. Experimental Setup

We performed experiments on the T-Drive dataset. We used one day of the data on Jan 11, 2012, which contains trajectories of more than 9,019 taxis and hundreds of thousands of passengers. We assumed that T-Drive drivers were SC workers and T-Drive passengers were SC requesters. The workers’ locations were those of the most recent drop-off locations while tasks were at the pick-up locations. The arrival order of the tasks was determined based on the sorting of their pick-up times.

In all of our experiments, we randomly sampled 500 tasks and 500 workers from T-Drive. These numbers are relatively small when compared to the size of the dataset because we focus on privacy and utility trade-offs rather than runtime performance. We chose typical ranges of values for $\epsilon$, $r$, $R_w$, as follows. Without loss of generality, we assumed the requesters and the workers have the same privacy level $(\epsilon, r)$, where $\epsilon \in \{0.1, 0.4, 0.7, 1.0\}$ and $r \in \{2.000, 1400, 800, 200\}$ in meters, ranging from strict to loose privacy requirements. We set the reachable distance of each worker to a random value in meters, $1000 \leq R_w \leq 3000$. We varied the U2U threshold $\alpha \in \{0.05, .1, .15, .2, .25, .3, .35, .4\}$ and U2E threshold $\beta \in \{.1, .15, .2, .25, .3, .35, .4\}$. Default values are shown in boldface. It is our intention to have different default values for $\alpha$ and $\beta$. The reason for this is that, in Algorithm 2, the U2U threshold is applied prior to the U2E threshold; therefore, the values of $\alpha$ must be upper-bounded by the default value of $\beta$, and the values of $\beta$ must be at least equal to the default value of $\alpha$.

In the following, we compare the performance of the proposed algorithms in terms of the performance metrics in Section III-C. In particular, we reported the total number of assigned tasks (utility) and the average travel distance (travel cost) across the assigned tasks. We also calculated the average number of candidate workers per task (system overhead) as well as avergae of the precision and recall scores (utility during U2U). We measured the total number of false hits (aka privacy leak or disclosure) and the total number of false dismissals (system overhead during U2E) over all tasks. All measured results were averaged over ten random seeds.

B. Experimental Results

We compare the performance of the variations of the three algorithms in Section IV: the oblivious algorithm, the probability-based algorithm and the Ranking algorithm that have access to exact location information (ground truth). First, the ground truth has two variants, GroundTruth-RR that uses the random rank strategy, and GroundTruth-NN that uses the nearest-neighbor strategy. Second, Oblivious-RR and Oblivious-RN refer to Algorithm 1 that uses the corresponding strategy (random rank or nearest-neighbor) to rank the candidate workers. Third, Probabilistic-Model and Probabilistic-Data are two variants of Algorithm 2 which correspond to the analytical and empirical approaches for quantifying the worker-task pair reachability in Section IV-B.

1) Overview of Results: We present the overview of results for comparing 1) analytical vs. empirical approaches for quantifying the reachability 2) random rank vs nearest-neighbor strategies for ranking candidate workers, and 3) performance of the algorithms.

We first compare the analytical and empirical approaches. The graphs in the first row of Figure 8 show the results by varying privacy guarantee $r$. We observe that Probabilistic-Model performs as well as Probabilistic-Data in terms of utility, and even slightly better in terms of travel cost and privacy leak. This result shows that the analytical model is as accurate as the empirical counterpart for estimating the worker-task pair reachability. Therefore, we use Probabilistic-Model from now on because it does not require precomputation.

Next, we compare the two strategies for ranking candidate workers, a random rank and the nearest-neighbor. The graphs in the second row of Figure 8 show the results by varying privacy guarantee $r$. We observe that GroundTruth-NN yields marginally lower utility when compared to GroundTruth-RR (321 tasks vs. 314 tasks in Figure 8d) but at a much smaller travel cost (1353 meters vs. 700 meters when $r = 200$ in Figure 8e). This is because GroundTruth-RR focuses solely on the competitive ratio without any spatial consideration, such as distance or reachability of a worker-task pair. For the same reason, when compared to Oblivious-RR, Oblivious-RN yields slightly lower utility at significantly lower travel cost and lower privacy leak. Hence, we will use GroundTruth-NN as the ground truth and Oblivious-RN as the baseline.

Last but not least, Figure 9 compares the performance of different algorithms by varying privacy guarantee $\epsilon$. We report two main results. First, Probabilistic-Model outperforms Oblivious-RN in all key metrics, including higher utility ($\times 2$ in Figure 9a), smaller travel cost ($\times 2/3$ in Figure 9b), and much lower disclosure of task location ($\times 500$ in Figure 9c), with only a slight increase in overhead (20% in Figure 9d). These improvements are more significant with higher privacy level (low $\epsilon$). The results confirm that the proposed probabilistic models are superior to the binary model in estimating the worker-task pair reachability. Second, when compared to the ground truth, privacy provided by the probability-based algorithm does not significantly affect utility (Figure 9a) and...
travel cost (Figure 9b), proving that tasks can be effectively assigned to nearby workers without compromising the key metrics. This result is significant because utility and travel cost are perhaps the most important factors in SC.

2) Details of Results: We further compare the algorithms with respect to each performance metric.

a) Utility (Number of Assigned Tasks): Figure 9a shows the results when varying privacy loss $\epsilon$. GroundTruth-NN achieves the highest utility followed by Probabilistic-Model, which obtains up to 200% higher utility than Oblivious-RN, especially with higher privacy level (smaller $\epsilon$). We also observe that when $\epsilon$ increases (less privacy), the utility of both Probabilistic-Model and Oblivious-RN asymptotically increases to the utility of GroundTruth-NN. This is because when less noise is injected, the perturbed locations tend to be closer to the actual ones. This yields a higher number of assigned tasks.

b) Worker Travel Cost: Figure 9b shows the results when varying privacy loss $\epsilon$. It is expected that GroundTruth-NN obtains the lowest travel cost as it has access to actual location data. Probabilistic-Model achieves significantly lower travel cost (up to 30%) when compared to Oblivious-RN. The improvement is higher at smaller $\epsilon$ (higher privacy). We also observe that as $\epsilon$ grows (less privacy), the worker travel cost of both Probabilistic-Model and Oblivious-RN asymptotically reduces to the travel cost of GroundTruth-NN.

c) Overhead and Privacy Leak: Figures 9c, 9d show the results when varying privacy loss $\epsilon$. Although the overhead of Probabilistic-Model is slightly higher than Oblivious-RN’s (i.e., up to 500 workers vs. up to 400 workers in the candidate set), Probabilistic-Model has a much smaller disclosure of location information (i.e., up to 2 false hits vs. up to 1500 false hits). This means that before a task can be assigned, Oblivious-RN needs to send the task to ~4.75 workers on average while the number is ~1.04 for Probabilistic-Model. Unlike Oblivious-RN, Probabilistic-Model usually identifies a candidate worker who is reachable to a task at the first try, without the need of sending the task to multiple workers during U2E. This result shows that our proposed approaches, the probability-based ranking and the thresholding heuristic in Section IV-C2, effectively limit the disclosure of location information. This is crucial because the disclosure of location information (among requesters and workers) is the only privacy leak in SCGuard.

We also show the impact of privacy loss $\epsilon$ on system overhead and privacy leak. As expected, when $\epsilon$ increases (less privacy), system overhead and privacy leak decrease while both precision and recall increase.

3) Effect of Parameter Settings: We evaluate the performance of the Probabilistic-Model by varying the U2U and U2E thresholds ($\alpha$, $\beta$).

a) Impact of varying U2U threshold $\alpha$: We first show the impact of varying $\alpha$ on both the U2U and U2E stages (see Figure 10). The main observation is that by decreasing the value of $\alpha$, the algorithm achieves higher utility (Figure 10a), lower travel distance (Figure 10b) at the expense of higher system overhead (Figure 10a). The reason for this is that the smaller U2U threshold $\alpha$, the larger the worker candidate set (see Line 7 of Algorithm 2), providing the task more chances to be assigned at the U2E phase. The impact of $\alpha$ on recall in Figure 10c confirms our intuition of achieving higher recall in Section IV-C1. Note that $\alpha$ does not directly impact false hit and false dismissal because they are U2U metrics (Figure 10d).

Hence, to achieve high utility, U2U threshold $\alpha$ should be as small as possible, while the size of the worker candidate set is manageable by the requester in terms of runtime (e.g., $\alpha = 0.1$). Figure 10e shows the runtime of the U2E stage when varying $\alpha$. As expected, the smaller $\alpha$, the higher the runtime due to having a larger candidate set per task.

b) Impact of varying U2E threshold $\beta$: We present the impact of varying U2E threshold $\beta$ on the U2E stage (see Figure 11). In most cases, we observe that as U2E threshold $\beta$ grows, utility, travel cost and the number of false dismissals decrease slightly while location disclosure decreases linearly. This result confirms our aim to reduce privacy leak by introducing U2E threshold $\beta$ in Section IV-C2. However, false dismissal increases at a certain value of $\beta$ (i.e., 0.25), which obviously decreases utility. The reason for this is that the higher U2E threshold $\beta$, the more likely a requester misses a candidate worker who is reachable to a task. In sum, to reduce privacy leak, the value of U2E threshold $\beta$ should be as large as possible, but at the same time should not incur significant utility loss (e.g., $\beta = 0.25$).

VI. RELATED WORK

Spatial crowdsourcing (SC) has gained popularity in both the research community (e.g., [12], [21], [24], [13]) and industry (e.g., TaskRabbit, Gigwalk). We review the literature from the following two aspects: location privacy threats in SC and privacy-preserving task assignment in SC.

Location Privacy Threats: Protecting location privacy in SC has attracted much interest. There have been known attacks on SC applications, such as location-based attacks during...
In contrast, our framework aims to protect locations of both workers and tasks. We also emphasize the online setting rather than the offline variant as in [26], in which the server must wait for a number of tasks to arrive prior to task assignment. In [9], Jin et al. propose a differentially private framework to select participants while participating in the spectrum-sensing tasks. Unlike ours, this study addresses a different problem where the task locations are predetermined and publicly known.

A few recent studies use encryption-based approaches [13], [16]. In [13], the locations of workers and tasks are protected by homomorphic encryption (HE). The platform performs task assignment based on the worker-task distances computed from the encrypted data. The workers receive the encrypted location of the assigned task and decrypt it to obtain the task location. While this approach guarantees exact assignment, its computational overhead is high. Unlike our study, the reason [13] could afford to use HE is that this work assumes batch assignment (server waits for multiple tasks to arrive and assign them to workers in batch). The experiment results in [13] show that it takes from 10 minutes to two hours for a batch assignment which is not feasible for our online setting. A closely related topic of research is the privacy-preserving ride-sharing service [16]. This work designs a cryptographic protocol to protect riders’ and drivers’ identifiable information, including location. However, this study uses a cloaking technique, which is vulnerable to background knowledge attack.

VII. EXTENSIONS AND OPEN PROBLEMS

Protection from malicious adversaries: Our framework assumes the semi-honest model, which is an important first step towards constructing protocols with stronger security under the malicious model. Using general crypto tools such as zero-knowledge proofs, the protocols can be usually transformed into secure protocols under the malicious model. Under the malicious model, the requesters, for example, can send multiple fake tasks to estimate the workers’ locations. We can avoid or mitigate such threats by complementary measures.
such as: (1) a reputation system that rates requesters (or workers) and helps detect those who send fake locations, and (2) a payment scheme that requires a payment for each task and increases the cost for attacks. Recent studies proposed techniques for detecting fake events in the SC app Waze [8] which can also be potentially adapted in our setting to detect malicious requesters or workers. We believe these are important research challenges for future research.

Protection for dynamic workers and tasks: Our current framework deals with the task assignment at one time point, i.e. given the locations of workers and tasks at that time point. When the locations of tasks and workers change dynamically, if we assume the location sets of each worker or task are independent of each other, the same guarantees of Geo-I still hold. However, the locations can be correlated in practice, for example, the workers’ traces can follow a specific (movement, or driving) pattern, and the task locations of individual requesters can be in the proximity of their whereabouts. In this case, we can use the extended Geo-I, as discussed in Section III.E of [2], which gives a privacy guarantee for the location set and the amount of noise for each location increases linearly with the size of the location set. This inevitably will reduce the utility of the system drastically.

Our three-stage framework is orthogonal to the underlying privacy mechanism and can be extended to work with other privacy notions for better utility in dynamic situations. For example, if we use δ location set-based differential privacy [27], we only need to modify the component of quantifying worker-task reachability and estimating worker-task distance (Section IV.B) based on the corresponding perturbation mechanisms, which can be an interesting future research direction.

Redundant task assignment: Our framework focus on SC apps (e.g., Uber, TaskRabbit) where performing a spatial task does not require multiple workers, such as taxi ride sharing, package delivery. However, there are spatial tasks that may need to be performed redundantly to ensure the quality of response, such as taking a picture of a particular restaurant, reporting how crowded a restaurant or a gym is at a certain time. In those cases, our proposed algorithm can be extended such that one task can be performed by multiple workers.

Particularly, the U2E stage of Algorithm 2 can be updated while the U2U and E2E stages stay the same. During U2E, given the perturbed locations of candidate workers $N_j$, a requester identifies the most likely reachable $K$ workers and sends the task location to these workers where $K$ is the number of workers required to perform the task. If $K$ is greater than or equal to $|N_j|$, the requester sends the task location to all candidate workers. Thereafter, during E2E, the selected workers accept the task if it is enclosed within their spatial regions; otherwise, they reject the task. This process may repeat until either the task is assigned to $K$ workers or the candidate set is exhausted.

VIII. CONCLUSION

We introduced SCGuard, a novel privacy-aware framework to protect locations of both workers and tasks in spatial crowdsourcing without any trusted entity. Our study enables the participation of workers and requesters without compromising their location privacy. We proposed models for quantifying the probability of reachability between a worker and a task, from which the probability-based algorithm was introduced to assign tasks to workers in an online manner. Our experimental results on real data demonstrated that the proposed techniques, algorithms, and heuristics achieve high utility, small worker travel cost, and low disclosure of location information.

IX. ACKNOWLEDGEMENT

This research has been funded in part by NSF grants CNS-1461963 and CNS-1618932, Caltrans-65A0533, AFOSR grant FA9550-12-1-0240, the USC Integrated Media Systems Center (IMSC), and the USC METRANS Transportation Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors such as NSF.

REFERENCES