Chapter 7
Location Privacy in Spatial Crowdsourcing

Hien To and Cyrus Shahabi

Abstract Spatial crowdsourcing (SC) is a new platform that engages individuals in collecting and analyzing environmental, social and other spatiotemporal information. With 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 entities (SC server) for effective assignments of tasks to workers.

This chapter first identifies privacy threats toward both workers and tasks during the two main phases of spatial crowdsourcing, tasking and reporting. *Tasking* is the process of identifying which tasks should be assigned to which workers. This process is handled by a spatial crowdsourcing server (SC server). The latter phase is *reporting*, in which workers travel to the tasks' locations, complete the tasks and upload their reports to the server. The challenge is to enable effective and efficient tasking as well as reporting in SC without disclosing the actual locations of workers (at least until they agree to perform a task) and the tasks themselves (at least to workers who are not assigned to those tasks).

This chapter aims to provide an overview of the state-of-the-art in protecting users' location privacy in spatial crowdsourcing. We provide a comparative study of a diverse set of solutions in terms of task publishing modes (push vs. pull), problem focuses (tasking and reporting), threats (server, requester and worker), and underlying technical approaches (from pseudonymity, cloaking, and perturbation to exchange-based and encryption-based techniques). The strengths and drawbacks of the techniques are highlighted, leading to a discussion of open problems and future work.

H. To (✉) · C. Shahabi (✉)
University of Southern California, Los Angeles, CA, USA
e-mail: hto@usc.edu; shahabi@usc.edu

A. Gkoulalas-Divanis, C. Bettini (eds.), *Handbook of Mobile Data Privacy*,
https://doi.org/10.1007/978-3-319-98161-1_7

7.1 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) [13]. SC has numerous applications in domains such as environmental sensing (iRain [1]), smart cities (TaskRabbit), journalism, and crisis response (MediaQ [15]). 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). After tasking, workers travel to the locations of the tasks, perform them and report the results to the server. This phase is referred to as *reporting*.

Both tasking and reporting phases often require workers and requesters to reveal locations of workers and tasks to potentially untrusted entities (server, other workers and other requesters). Several studies (e.g., [5, 13, 14, 30]) 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. Similarly, reporting spatial tasks would enable the server to infer the workers' locations since they must have visited the locations of the tasks. However, disclosing individual locations has serious privacy implications. Leaked locations often lead to a breach of sensitive information such as an individual's health (e.g., presence in a cancer treatment center), alternative lifestyles, political and religious preferences (e.g., presence in a church). Knowing user locations, an adversary can stage a broad spectrum of attacks such as physical surveillance and stalking, and identity theft [25]. Particularly, in [36], the authors show that hackers can stalk users in Waze—a popular SC application—by generating fake events such as accidents. Consequently, mobile users may not agree to engage in spatial crowdsourcing if their privacy is violated; thus, ensuring location privacy is key to the success of SC.

The first step of the tasking phase is task publication. There are two modes of task publication in SC: push (e.g., iRain) vs. pull (e.g., TaskRabbit). With the pull mode, the server publishes the spatial tasks and online workers can choose any spatial task in their vicinity without the need to coordinate with the server. With the push mode, online workers send their locations to the server, which then assigns to every worker his nearby tasks (posted by requesters). Each mode shares similar challenges and has its own unique challenge. The common challenges are that a worker should know a task location only if he plans to perform the task; likewise, only requesters who have tasks performed by the worker should know his location. Furthermore, the unique challenge with the push mode is that the server must match workers to tasks without compromising their privacy. This requires strategies to ensure effective task assignment without revealing locations of tasks and workers. On the other hand, the

Table 7.1 Attacks on SC users

	Tasking	Reporting
Push	[12]	[27]
Pull	[Sect. 7.3.2]	[27, 36], [Sect. 7.3.2]

unique challenge with the pull mode is to enable every worker to request tasks, perform them and subsequently post the results to the server without revealing his location and identity. Finally, providing privacy protection simultaneously both tasking and reporting phases introduces another set of challenges to both push and pull modes.

Among the two modes of task publishing, privacy protection in the push mode is more challenging because tasking in the push mode is more complex than that of the pull mode. Countermeasure studies in the pull mode have been the main focus in the past decade with an emphasis on a special class of SC, named *participatory sensing* (PS). PS usually assumes the pull mode of task publication (workers choose tasks); therefore, the main privacy threats to workers occur during reporting. Meanwhile, the most recent studies in SC have focused on the push mode (server assigns tasks to workers); for this reason, main privacy breaches occur during tasking [12]. Consequently, the existing studies in SC can be classified into two groups: (1) preserving privacy during *reporting in the pull mode* [2, 27, 37], and (2) preserving privacy when *tasking in the push mode* [8, 10, 12, 22, 26, 31, 32, 35, 38].

In this chapter we study the privacy threats to workers and requesters[1] in SC, during both tasking and reporting phases with either push or pull mode. Throughout this chapter we also identify three major drawbacks of the existing studies. First, they solely focus on protecting privacy during either phase of tasking or reporting, but not both. Second, most of these studies ensure privacy for workers only. To elaborate, we perform a set of simple attacks on TaskRabbit to demonstrate that locations of workers and requesters can be learned during both tasking and reporting phases. Third, despite the fact that most studies focus on either reporting in the pull mode or tasking in the push mode, privacy threats to SC users may also occur in other scenarios. Table 7.1 shows that there have been known attacks under the tasking and reporting phases with either the push or pull mode of task publishing. We demonstrate such threats in Sect. 7.3.2 via another set of attacks on TaskRabbit. These observations open some new research questions such as: how do we protect location privacy of both workers and tasks, simultaneously, during both the tasking and reporting phases of SC, and what are the promising privacy techniques to be used?

There have been recent surveys in privacy-preserving participatory sensing [4, 24] and mobile crowdsourcing [21]. Unlike these surveys, which provide an overview of a broad range of related problems, this chapter provides an in-depth study of the privacy challenges and the solutions proposed in the prior studies.

[1]Task locations can indirectly reveal requesters' location, i.e., requesters often post tasks in the proximity of their locations.

The remainder of this chapter is organized as follows. In Sect. 7.2 we introduce spatial crowdsourcing and compare it with related concepts. Section 7.3 illustrates potential privacy risks to both workers and requesters. Section 7.4 summarizes existing solutions addressing the privacy concerns in both the tasking and reporting phases of SC. Finally, we present our conclusions and future research directions in Sect. 7.5.

7.2 Spatial Crowdsourcing

In this section we define spatial crowdsourcing and present two modes of task publishing, push vs. pull, with the push mode recently being dominant in the research community. Thereafter, we differentiate SC from the related topic of participatory sensing, which usually assumes the pull mode of task publication.

7.2.1 Generic Framework

Spatial crowdsourcing (SC) [13] is a type of online crowdsourcing where performing tasks requires workers to physically be present at the locations of the tasks, termed *spatial tasks*. A spatial task is a query to be answered at a particular location and must be performed before a deadline. An example of a spatial task is taking a picture of a particular dish in a restaurant. This means that the workers need to physically travel to the location of the restaurant in order to take the picture. A worker is a carrier of a mobile device who will perform spatial tasks for some incentives.

Spatial crowdsourcing has gained popularity in both the research community (e.g., [13, 32]) and industry (e.g., TaskRabbit, Gigwalk). A recent study [34] distinguishes SC from related fields, such as generic crowdsourcing, participatory sensing, volunteered geographic information, and online matching. Research efforts have focused on different aspects of SC, including task assignment, task scheduling, privacy, trust and incentive mechanism.

7.2.2 Task Assignment: The Focus of Spatial Crowdsourcing

The main challenges of spatial crowdsourcing are due to the large-scale, ad hoc and dynamic nature of the workers and tasks. To continuously match thousands of SC campaigns, where each campaign consists of many spatiotemporal tasks with millions of workers, a server must be able to run efficient task assignment (aka tasking). According to [13], there are two types of tasking modes based on how workers are matched to tasks—server-assigned tasks (SAT) and worker-

selected tasks (WST)—which are also known as push and pull modes, respectively. Depending on the choice of a particular mode, the focus of privacy protection is either at the tasking or the reporting stage of SC.

With the *pull* mode, the server publicly[2] publishes the spatial tasks, and online workers autonomously choose tasks in their vicinity without coordinating with the server. One advantage of the pull mode is that the workers do not need to reveal their locations to server. However, one drawback of this mode is that the server does not have any control over the allocation of spatial tasks; this may result in some spatial tasks never be assigned, while others are assigned redundantly. Another drawback of the pull mode is that workers choose tasks based on their own objectives (e.g., choosing the k closest spatial tasks to minimize their travel cost), which may not result in a globally optimal assignment. Examples of the pull mode are TaskRabbit and Waze.

With the *push* mode, requesters post tasks that include locations, while online workers send their locations to the server, which assigns tasks to nearby workers. The advantage of this mode is that unlike the pull mode, the server has the big picture and can assign to every worker his nearby tasks while maximizing the overall task assignment. However, the drawback is that locations of both tasks and workers should be sent to the server for effective assignment, which can pose privacy threats. Examples of the push mode include Uber, iRain [30] and MediaQ [15].

Most SC studies assume the push mode and thus emphasize privacy protection during the tasking phase. With the pull mode, the main focus of privacy protection is shifted to the reporting phase, which has been well studied in the context of participatory sensing (e.g., [2, 12, 27, 35, 37]). With participatory sensing, the goal is to exploit the ability of mobile users to collect and share data using their sensor-equipped phones for a given campaign. Most studies on participatory sensing focus on small campaigns with a limited number of workers; hence, they do not have issues of task assignment. However, with SC, the focus is on devising a scalable, generic and multipurpose crowdsourcing framework, similar to Amazon Mechanical Turk, but spatial, where multiple campaigns can be handled simultaneously. Therefore, the main challenge with SC is to devise an efficient approach to assign tasks to workers given the large scale of an environment.

7.3 Privacy Threats

There have been known attacks on SC applications, such as location-based attacks during tasking in the push mode [12] and collusion attacks during reporting in the pull mode [36] (see Table 7.1). Despite the fact that most studies have solely focused on one of the two major threats, privacy risks to SC users may occur in the other

[2]Exact geographical coordinates of the tasks may not be published; instead, their cloaked locations or representative names are provided.

scenarios: reporting in the push mode and tasking in the pull mode. In this section we present a threat model which characterizes the *full spectrum of privacy threats to workers and requesters during both tasking and reporting phases with either push or pull mode*. Next, we illustrate the privacy risks on TaskRabbit.

7.3.1 Threat Model

As the privacy threats vary according to the modes of task publishing, we discuss possible threats associated with each mode.

7.3.1.1 Privacy Threats with the Push Mode

With the *push* mode, the server takes as input the perturbed locations of both workers and tasks to perform effective task assignment; hence, there is a serious privacy threat from the server which might become a single point of attack. Figure 7.1a depicts the threat model for the push mode of spatial crowdsourcing. The first row means that locations of workers and tasks are protected from the server at all the time. The role of the server is to create the *assignment links* between the workers and the requesters so that they can establish a direct communication channel among themselves. Each worker-requester pair cooperatively decides whether to accept the assignment from the server. If yes, they send a *consent* message to the server, confirming that the worker will perform the requester's tasks. This agreement is illustrated by the first *reporting link* in Fig. 7.1a. We argue that to preserve location privacy during both tasking and reporting phases, task locations need to be protected from the server. Otherwise, the completion of a task reveals that some workers

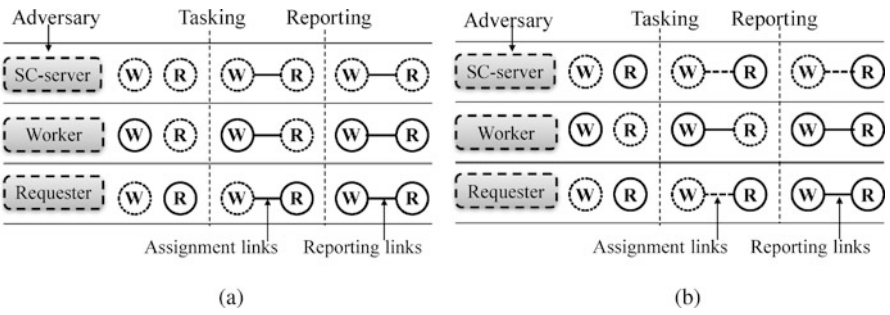

Fig. 7.1 Threat models in spatial crowdsourcing. W and R denote workers and requesters, respectively. The dotted circles surrounding them denote that they are protected from an untrusted entity shown in the first column. After tasking and reporting, the assignment and reporting links between W and R represent the established connections during each phase. The dashed links indicate connections that are oblivious to the corresponding malicious entity. (**a**) Push mode. (**b**) Pull mode

must have visited the task's location. In restrictive privacy settings, workers and requesters can also be malicious to each other. Hence, to ensure minimum disclosure among them, only workers who aim to perform the tasks should know the tasks' locations (see the second row in Fig. 7.1a). Likewise, a requester should only know the workers' locations once her tasks are matched to and then performed by those workers (see the third row in Fig. 7.1a).

We emphasize the minimum disclosure of location information for both workers and tasks. The reason for this is twofold. First, the server knows only the assignment links between workers and tasks. Due to such links, the assigned workers (or tasks) may infer that there exist nearby tasks (or workers). These disclosures are unavoidable in the push mode of SC. Second, the disclosure of workers' locations to their corresponding requester is inevitable at the reporting phase per definition of SC. It is worth mentioning that this threat model is restrictive; hence, weaker variants exist. For example, most existing studies in the push mode assume that workers are trusted [10, 12, 22] and task locations are public [8, 26, 32, 35, 38].

7.3.1.2 Privacy Threats with the Pull Mode

With the *pull* mode, despite the fact that workers do not need to send their locations to the server, the locations can still be learned during both tasking and reporting phases. As long as a worker connects to the server to either *request* some tasks or *report* results, he may reveal to the server patterns of where and when the connections were made and what kind of tasks he wants to perform. Consequently, in [27], the authors show that linking multiple requests or reports of the worker may allow an adversary to trace him since the worker's location information can be tracked through several stationary connection points (e.g., cell towers). In addition, the worker's location trace can be inferred by both the server and requesters since he must be in the vicinity of the tasks in order to perform them. Figure 7.1b depicts the proposed threat model for the pull mode. To preserve privacy and identity of the workers from the server, both assignment links and reporting links should be secure during tasking and reporting phases, respectively. This is because if the connections are discovered by the server, which already knows the locations of tasks, the server learns the locations of workers since they must have visited the locations of the performed tasks. Hence, the workers must request tasks without revealing their identity to the server; once the tasks are performed, the workers must also disassociate their connections with the performed tasks while uploading task content to the server. Similar to the push mode, both workers and requesters themselves can be hostile to one another. Thus, the privacy threats from workers and requesters (rows 2 and 3 in Fig. 7.1a) are similar to those in the push mode (rows 2 and 3 in Fig. 7.1b), except the difference in the assignment links of the two second rows. The reason for this is that the requester is oblivious to the requests between the worker and the server during tasking.

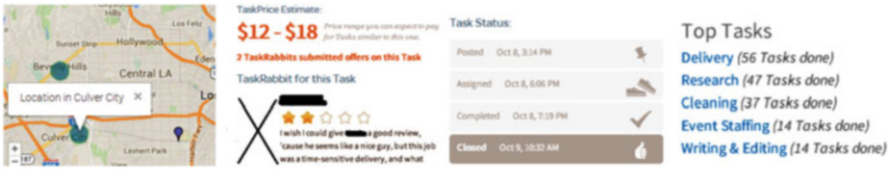

Fig. 7.2 Screenshots of TaskRabbit web application from worker Bob. (**a**) Task locations. (**b**) Task price. (**c**) Task status. (**d**) Performed tasks

7.3.2 Case Study of TaskRabbit

We show that an adversary can perform harmful attacks on a typical SC application without much effort. TaskRabbit is a pull-based[3] online and mobile marketplace that matches workers with requesters, allowing requesters to find immediate help with everyday tasks including, but not limited to, cleaning, moving, and delivery. In the following we discuss the aforementioned threats to TaskRabbit users. Note that the following attacks on TaskRabbit.com were conducted in October 2014; the website has been updated since then.

We first show the breach of task location during tasking. We signed up as a worker account and searched for delivery tasks in Los Angeles; 2381 spatial tasks were found. We obtained various information about a particular task by clicking on it, such as description, price, task status and cloaked locations. Although each location is cloaked in a circle with a radius of half a km[4] (Fig. 7.2a) to protect task locations from workers, the actual drop-off and pick-up locations were mentioned in the task description, i.e., *"Please pick up a box of mini-muffins from (S) promptly at 8 am on Tues, 9/4, and drive them straight to me at (D)."* It is also worth noting that task requests often contain sensitive information, such as health status of the requesters. An example of a sensitive task is one with title *"super easy task deliver a bag to the doorstep of a sick friend."* Nonetheless, these privacy risks are due to the disclosure of task content, which is beyond the scope of this study.

We then show the leak of worker location during tasking and reporting. To gain a competitive advantage, a worker may wish to not disclose locations of his visits to other workers and requesters. The task status (Fig. 7.2c) infers that the worker, referred to as Bob, was at the pick-up and drop-off locations of the task during the 1-h period between his assigned time and his completed time. The risk of precisely inferring Bob's locations is even higher for time-sensitive tasks such as delivery and help at home, which requires him to meet requesters in-person

[3]We present the privacy threats to a pull-based SC system only; however, some of these privacy threats also occur in push-based SC such as iRain.

[4]We obtained this information via JavaScript code.

Table 7.2 Three tasks requested by requester Alice

Task description	Corresponding JavaScript
Quick post-party dishwashing clean up needed	"radius" : "0.5", "geo_center" : {"lat" : "33.xxxxxx", "lng" : "-118.xxxxxx"}
Take down light Christmas decorations	"radius" : "0.5", "geo_center" : {"lat" : "33.xxxxxx", "lng" : "-118.xxxxxx"}
Put up 20 yard sale signs in Mid-Wilshire area	"radius" : "0.5", "geo_center" : {"lat" : "33.xxxxxx", "lng" : "-118.xxxxxx"}

We replaced six digits after the decimal point of "geo_center" by 'x' to protect the privacy of the requester

at a specific place and time. This inference attack shows that TaskRabbit does not guarantee privacy protection for the pull mode in Sect. 7.3.1, which says that Bob's locations are private to the server and only requesters who have their tasks performed by Bob should know his locations. In addition, one can also see much more information about Bob, including his previously performed tasks (Fig. 7.2d) and all reviews from the requesters who hired him. These associations between Bob and his performed tasks indicate that the assignment links and reporting links are known to the server.

Among Bob's requesters, we randomly picked one named Alice. We further show that her home location can be learned by tracking her task requests. We searched for household tasks that Alice requested in the past; three of them are shown in Table 7.2. These tasks were in the proximity of each other and likely situated at her home. Our hypothesis is that the tasks' locations were randomly cloaked such that the cloaking regions covered the actual location of the tasks. The location must be in the overlapped area using triangulation. We validated our hypothesis by confirming that the location of another task, whose location was known, is within the overlapped region. This attack suggests that the more task requests are posted, the more accurately their locations can be learned. This simple attack is against the threat model, which states that the locations of Alice's tasks should only be revealed to the workers who performed her tasks.

7.4 Privacy Countermeasures

In this section we survey some state-of-the-art approaches addressing the privacy issues in spatial crowdsourcing. We first categorize the studies into two groups: *tasking in the push mode* and *reporting in the pull mode*. Subsequently, each subgroup is further classified according to the applied techniques. Within each subgroup we identify one key paper shown in boldface to be presented in depth while follow-up studies are briefly discussed. An overview of these studies is presented in Table 7.3. The table shows that the studies solely focus on location privacy of workers and assume that the locations and content of tasks are public.

Table 7.3 Overview of problem focuses (Re: reporting, Ta: tasking); privacy techniques used (Ps: pseudonym, Cl: cloaking, Pt: perturbation, Ex: exchange-based, En: encryption-based); threats (W: worker, T: requester, S: server); trusted third party (TTP); optimization type (ST: single task, MT: multiple tasks). x and (x) represent primary and secondary aspects, respectively

	Phase		Techniques					Protection		Threats			TTP		Opt. type	
Paper	Re	Ta	Ps	Cl	Pt	En	Ex	W	T	W	R	S	Yes	No	ST	MT
Shin et al. 2011 [27]	x	x	x	(x)		(x)		x	N/A		N/A	x	x		x	
Boutsis et al. 2013 [2]	x		(x)				x	x	N/A	(x)	N/A	x		x		x
Zhanget al. 2016 [37]	x						x	x	N/A		N/A	x		x		x
Kazemi et al. 2011 [12]		x	(x)	x				x			(x)	x	x			x
Vu et al. 2012 [35]		x		x		(x)		x		(x)	(x)	x	x			x
Sun et al. 2017 [28]		x		x				x				x	x			x
Pham et al. 2017 [20]		x		x				x	x			x	x			x
To et al. 2014 [32]		x			x			x		(x)	(x)	x	x		x	
Gong et al. 2015 [8]		x			x			x		(x)	(x)	x	x		x	
Zhang et al. 2015 [38]		x			x			x		(x)	(x)	x	x		x	
To et al. 2016 [31]		x			x			x		(x)	(x)	x	x		x	
Pournajaf et al. 2014 [22]		x		x				x			(x)	x	x			x
Hu et al. 2015 [10]		x		x				x			(x)	x	x			x
Shen et al. 2016 [26]		x				x		x		(x)	(x)	x		x	x	
Liu et al. 2017 [17]		x				x		x				x		x	x	
Liu et al. 2017 [16]		x				x		x	x			x		x	x	

Moreover, the server is regarded as a primary threat in all studies, while some consider workers and requesters as secondary adversaries. We also notice that the most recent studies focus on the push mode, which requires privacy protection during tasking. This problem is considerably more challenging when compared to the problem of privacy-preserving reporting in the pull mode.

7.4.1 Protection in the Pull Mode

Privacy protection in the pull mode has been studied in the context of participatory sensing. In this section we highlight recent studies that often focus on the reporting phase of the pull mode. They use either *pseudonymity* [27] or *exchange-based* techniques [2, 37]. The pseudonymity method disassociates the connections between one's uploaded data and his/her identity while the latter exchanges workers' crowdsourced data and location information before uploading them to a server so that the server is uncertain about locations of individual workers.

7.4.1.1 Pseudonymity Techniques

Shin et al. [27] propose a privacy-preserving framework for the pull mode as illustrated in Fig. 7.3. A requester submits a task to a *registration authority* (RA) that will verify the task before sending it to a *task service* (TS). Also, a worker connects to TS through an anonymizing network such as Tor to request new tasks, referred to as a *task subset*. After receiving the requested tasks, the worker chooses which tasks to accept. He then performs the tasks and uploads the corresponding task reports to a *report service* (RS) via an *anonymous service* (AS). In this framework, RA and AS are trusted while TS, RS and requesters can be hostile. TS and RS can be considered as services performed by the server.

This study [27] provides privacy protection in both tasking and reporting phases. During tasking, the role of the anonymizing network is to disassociate the worker and his requested tasks, depicted by the first and the third assignment links in Fig. 7.1b. To preserve privacy during reporting, a worker typically sends his task report to RS via AS, which routes the report through multiple servers so that the server (i.e., TS and RS) cannot associate multiple locations (i.e., IP addresses) with

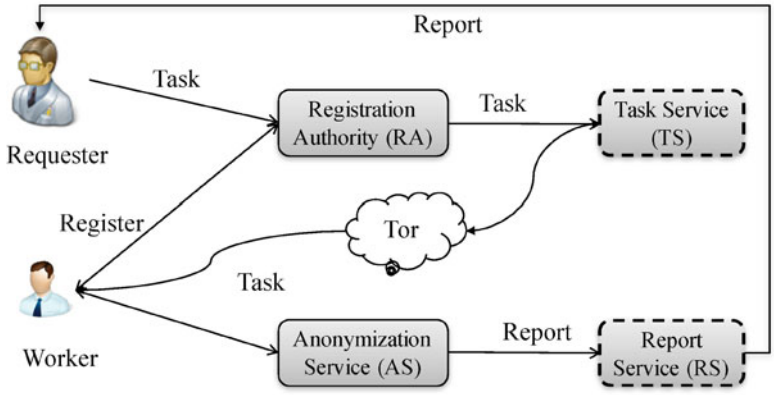

Fig. 7.3 A framework for privacy protection during tasking and reporting in the pull mode. Dashed entities are malicious, while others are trusted

the identity of the same worker. Consequently, the server is oblivious to the first reporting link in Fig. 7.1b. More recently, there has been closely related work in participatory sensing that enables workers to hide their locations and data ownership by passing the collected data through a random neighboring worker multiple times before uploading the data to the server [11].

7.4.1.2 Exchange-Based Techniques

Pseudonymity techniques are ad hoc and do not provide quantifiable privacy protection. For more sensitive tasks that require strong privacy guarantee, k-anonymity [29] is used in [27] to ensure that each report is anonymized with $k-1$ reports generated by other workers with similar sensitive information. However, such techniques may not be applicable to SC because the worker location is part of the report. To address such a problem, Boutsis and Kalogeraki [2] propose the exchange-based technique to obscure the workers by exchanging their reports between them before disclosing the sensitive information to an untrusted server (i.e., server). Such a technique can be used as AS in Fig. 7.3, aiming to protect the first reporting link in Fig. 7.1b from the server.

To provide a quantifiable privacy guarantee, in [2] the authors use location entropy as the measure of privacy or the attacker's uncertainty. The study aims to make all workers' trajectories as equiprobable to contain sensitive locations by maximizing the location entropy of an individual's trajectories to be defined later. To maximize the location entropy, trajectories with sensitive locations are distributed among multiple workers. Particularly, each worker's mobile phone identifies the k most frequently visited locations as *sensitive data* from a *local* trajectory database. A trajectory is selected for exchange if removing the trajectory increases the entropy of the database, computed as follows.

$$H_i = \sum_{loc_{ij} \in L} Pr(loc_{ij}) log(Pr(loc_{ij})))$$

where L is the set of locations and $Pr(loc_{ij})$ is the fraction of total visits to location j that belongs to user i. Consequently, an attacker will not be able to identify sensitive locations or identities of the workers.

For each worker, the trajectories that contain locations with high frequency are exchanged with other workers since removing high-frequency trajectories (trajectories that contain sensitive locations) makes the frequency of the locations in L more homogeneous and thus increases the entropy. Furthermore, as other workers may not be trustful, not only the set of high-frequency trajectories are exchanged but also another set of trajectories that do not contain the sensitive locations. This guarantees that neighboring workers are not able to associate the worker with his sensitive data. Consequently, both frequent and non-frequent trajectories are selected and forwarded to individual workers so that no worker can be certain about the sensitivity of any trajectory.

A drawback of computing entropy locally is that the exchange decisions can be suboptimal due to the lack of a global view of all workers. This is because individual workers try to maximize their own entropy regardless of each other, which goes contrary to the fact that exchanging trajectories alters the location entropy of multiple workers. Thus, the exchange-based technique should consider the entropy with respect to all workers as opposed to individual workers. Therefore, Zhang et al. [37] introduce a similar framework, but here workers coordinate with each other to exchange their sensing data, including locations before uploading to the server. As a result, all sensitive locations are equally likely visited by any worker so that the actual trajectory of each worker cannot be learned. However, unlike [2] where entropy is computed for a single worker, here entropy is calculated for all workers.

Although the exchange-based technique is simple and does not rely on a trusted server, the actual location information is still uploaded to the server. Therefore, this approach is vulnerable to background knowledge attack. For instance, if the server knows that only worker w_i visits a particular location where a report was uploaded, the server is certain that w_i actually made the report.

7.4.2 Protection in the Push Mode

While preserving privacy during reporting in the pull mode has been largely studied in the context of participatory sensing (a recent survey can be found in [4]), recent SC studies focus on the more challenging phase of tasking. These studies generally assume the push mode. We emphasize that focusing on the tasking step in the push mode is the correct approach, given that SC workers have to physically travel to the task location. The completion of a task discloses the fact that some worker must have been at that location, and this is unavoidable in SC. Focusing on tasking also makes sense from a disclosure volume standpoint. During the assignment, all workers are candidates for participation; therefore, locations of all workers are exposed, absent a privacy-preserving mechanism. Nevertheless, after task request dissemination, only a few workers will participate in task completion, and *only if they give their explicit consent* (see the threat model for the push mode in Sect. 7.3.1).

Various techniques have been proposed to protect location privacy of workers during task assignment in SC, including *cloaking* (hide the accurate location in a cloaked region) [10, 13, 22, 35], *perturbation* (distort the actual location information by adding artificial noise) [8, 31, 32, 38] and *encryption* [26, 27].

7.4.2.1 Cloaking Techniques

The studies in this category generally implement spatial k-anonymity by generating a *c*loaking *r*egion (CR) for each worker, which includes $k - 1$ other workers. To guarantee strong privacy protection, peer-to-peer spatial k-anonymity [3] has been

adopted in these studies. In the following we first present a simplified version of tasking without constraints. Next, we survey some recent studies that consider real-world constraints, such as the travel budget of each worker and a worker's willingness to perform tasks.

Task Assignment Without Constraints

In [27], each worker requests a *task subset* of size p at a time; however, choosing an appropriate value of p is not trivial. Large p may lead to not only high communication overhead between workers and TS, but tasks are also *unnecessarily* disclosed to the workers. In contrast, small p may result in some tasks that will never be accepted by any worker. One reason for this is that a worker can browse far-away tasks that he cannot complete before the tasks' deadlines. This redundant disclosure incurs additional privacy threats to the requesters of those tasks.

In order to minimize such disclosure, Kazemi and Shahabi [12] propose a privacy framework that enables each worker w_i to query the server for a set of nearby spatial tasks. Particularly, the server needs to distribute a set of spatial tasks to workers such that each worker is assigned a subset of tasks that are closer to him than to any other worker. Without privacy protection, the server can construct a Voronoi diagram of the workers, including a set of cells where each cell belongs to a worker, and any spatial task in the cell is closer to the worker than to any other worker. Once the server computes the Voronoi diagram of the workers, it forwards to each worker all the spatial tasks lying inside the corresponding cell. However, in such a scenario, an adversary may infer the worker's identity by associating the query to query location (i.e., the location from which the query is issued. This is referred to as *location-based attack*. Consequently, the framework aims to protect worker identity from location-based attacks by disassociating a query from the query location.[5] The framework named PiRi (partial-inclusivity and range independence) has both *query formation* and *query selection*.

Query Formation

In the query formation step, each worker w_i computes his Voronoi cell by communicating with his neighboring peers [3]. The worker forms his CR, where his location is blurred among $k - 1$ other peers (with $k = 3$, the solid-lined rectangle in Fig. 7.4a). The worker can send the CR along with the radius r (i.e., the smallest enclosing circle of w_i's Voronoi cell) to the server to retrieve all the tasks which lay inside his Voronoi cell. However, the range query is dependent on the size of the worker's Voronoi cell (range dependency), which is a potential for information leaks. Considering an extreme scenario where the server knows the

[5]However, this study assumes that workers trust one another. Hence, a more recent study [35] solves a similar problem as in [12] without the assumption of trusted workers.

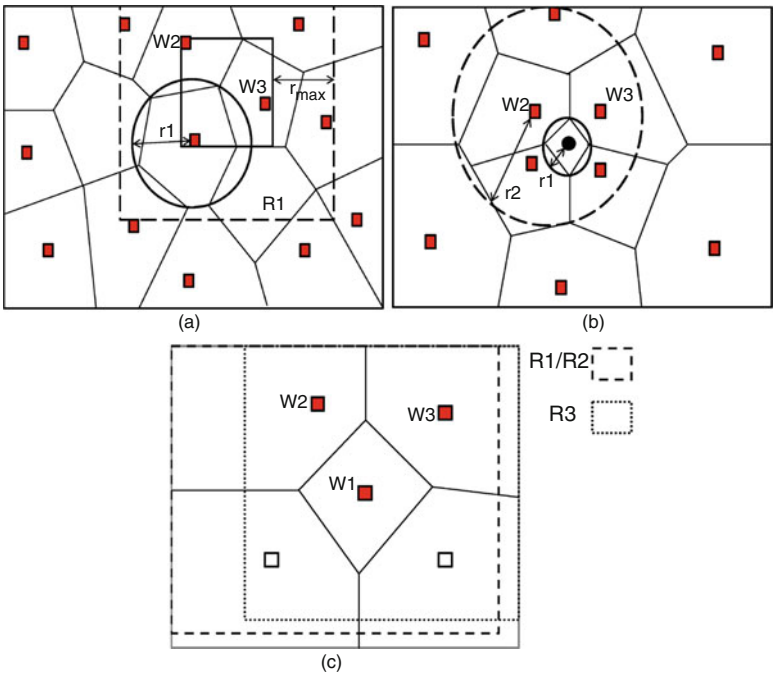

Fig. 7.4 Examples of range dependency and all-inclusivity. (**a**) Query formation. (**b**) Range dependency leak. (**c**) All-inclusivity leak

workers' locations, it also knows their Voronoi cells and therefore the radius r for each of them. Consequently, the server can easily identify the query issuer (i.e, the set of all workers in the CR with radius r). Figure 7.4b depicts such a scenario, where w_1 (black-filled circle) cloaks himself with w_2, and sends the CR along with radius r_1 to the server (see the size of r_1 as compared to r_2). The server, knowing the location of the workers, and hence their Voronoi cells (i.e., r_1, and r_2), relates the query with radius r_1 to its query location (i.e., the location of a worker with the Voronoi cell of the same radius).

In order to avoid the range dependency leak, each worker w_i should cloak not only his location but also his range query among $k-1$ other peers. In other words, instead of forming his range query with radius r_i, the worker forms his query with radius r_{max}–the maximum radius among all the k peers inside the CR. This guarantees the k-anonymity at all times. In Fig. 7.4a, R_1 (the dotted line rectangle) shows the query region formed by r_{max}.

Query Selection

Once all workers have formed their query regions, they can send them out to the server. However, the server can utilize the gathered information (i.e, query regions)

from all workers to attack the system (all-inclusivity leak). Figure 7.4c illustrates such scenario, in which workers $w_{1..3}$ participate in the system. The figure shows that w_1 cloaks himself with w_2. Similarly, w_2 forms a cloaked region with w_1. Subsequently, both w_1 and w_2 form identical query regions. The figure also depicts that w_3 cloaks himself with w_1. Accordingly, the server can easily identify w_3 by relating it to the query region R_3, since w_3 appears only once (i.e., R_3) in all the three submitted query regions to the server. This indicates that the more workers submit queries to the server, the more information the server has to infer the workers' identities. To prevent this leak, the authors attempt to minimize the number of queries submitted to the server while assigning the nearby tasks to *every* single worker.

Since there is a large overlap among the query regions of the workers, a worker can share his result received from the server with all the peers whose Voronoi cells lay completely inside his query region. The problem is how to select the group of representative workers, formally stated as follows. Given a set of workers W, and a set of spatial tasks T, let R and V be the set of query regions and Voronoi cells for the set W, respectively, where R_i corresponds to the query region for worker w_i, and V_i is the Voronoi cell for w_i. The problem is to find a set $C \subseteq R$ that covers the entire set V with minimum cardinality. This problem is shown to be NP-hard by reduction from the minimum set cover problem [12]. One well-known approach for solving the set cover problem is a greedy algorithm that picks a representative worker whose query region covers the largest number of uncovered Voronoi cells from V. However, this approach is applicable only in a centralized setting, where a global knowledge of the environment is available. To address this issue, the greedy heuristic is extended to support the distributed environment. Particularly, a voting mechanism is devised to select the set of representative workers, whose CRs are sent out to the server. These query results will later be shared with the rest of the workers. This step has been shown to prevent the all-inclusivity leak [12].

Task Assignment with Constraints

In [12, 35], spatial tasks are distributed to the corresponding nearest workers. This objective may not necessarily fit SC applications as workers often have various constraints that need to be considered. For example, they may be willing to perform tasks that are far away, but within their daily travel routes. To capture such constraints, each worker w_i has a cloaked area a_i and a limited travel budget b_i, which denotes the maximum distance he is willing to travel [22]. Given the cloaking regions of a set of workers, the objective of the server is to match a set of spatial tasks to the workers such that task assignment is maximized while satisfying the travel budget constraint of each worker.

As *travel cost* (often measured by the distance between tasks and assigned workers) is an important performance metric in SC, in the following we first present

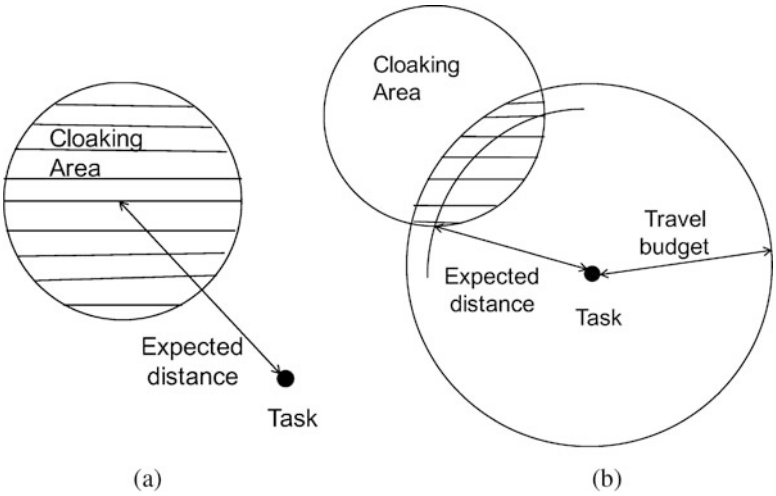

Fig. 7.5 Distance estimation methods. (**a**) Centroid-point method. (**b**) Expected probabilistic method

two methods for estimating the travel cost from the cloaked areas of the workers. Thereafter, we present the problem of *s*patial *t*ask *a*ssignment with *c*loaked locations (STAC) [22].

Distance Estimation

Given the cloaked area a_i of the workers, STAC proposes two methods for estimating the expected distances between pairs of workers w_i and tasks t_j, named $\hat{d}_{i,j}$. The baseline method approximates the worker location as the centroid of his cloaking area as depicted in Fig. 7.5a. Another method uses the travel budget of the worker to prune the cloaking area (i.e., the dashed area in Fig. 7.5b), resulting in a shrunk area that contains only accessible locations of the worker. Consequently, $\hat{d}_{i,j}$ is estimated by the distance between the task location and shrunk areas.

Next, we present a two-phase optimization approach to STAC. The first phase, denoted as G-STAC, globally optimizes task assignment using cloaked locations of the workers. The second phase, referred to as L-STAC, locally optimizes the assignment of individual workers using their own exact locations.

Global Optimization

Given a set of workers and a set of spatial tasks, G-STAC aims to achieve a particular goal of task coverage with the minimum travel cost. G-STAC is formally defined as follows.

$$min\ TC = \sum_{i \in W} \sum_{j \in T} \hat{d}_{i,j} x_{i,j}$$

$$s.t.\ TU = \sum_{j \in T} \frac{\sum_{i \in W} x_{i,j}}{k_j} \geq gm$$

$$\sum_{j \in T} \hat{d}_{i,j} x_{i,j} \leq b_i$$

where task cost (TC) is the total distance traveled by all workers, while task coverage or utility (TU) is the total covered fraction of tasks. $\hat{d}_{i,j}$ is the estimated distance between worker i and task j, $x_{i,j} = 1$ means worker i is assigned to task j, otherwise $x_{i,j} = 0$, k_j is the required coverage of t_j (i.e., the number of workers to perform t_j) while $g \in (0, 1]$ indicates the required fraction of coverage for a task. The last constraint guarantees that w_i's travel distance is within his budget b_i.

G-STAC is shown to be NP-hard by reduction from the minimum set cover problem. Therefore, a greedy algorithm is proposed that iteratively selects the most cost-effective worker-task pair and updates TU until either the coverage goal is achieved or the travel budgets of all workers are spent. A worker-task pair is cost-effective if the ratio of expected distance to the expected coverage contributed by this worker is small.

Local Optimization

The output of G-STAC is the best mapping of tasks to workers, which is sent to workers as suggested assignments. However, a worker may be assigned tasks whose locations exceed his travel budget, or nearby tasks are not assigned to him because their distance has been estimated as being farther away. Thus, the local refinement phase (L-STAC) is performed by individual workers' devices for more coverage and lower travel cost. A caveat is that selecting the closest tasks for each worker may result in over-coverage for some tasks, while the others remain unperformed. Consequently, in addition to minimizing the travel cost, L-STAC also tries to minimize the change in the local optimization when compared to the global optimization. L-STAC is formally defined as follows.

$$min\ TC_i = \sum_{j \in T} d_{i,j} y_{i,j}$$

$$s.t.\ |y_i - x_i| < \epsilon$$

$$\sum_{j \in T} \frac{y_{i,j}}{k_j} \geq \sum_{j \in T} \frac{x_{i,j}}{k_j}$$

$$\sum_{j \in T} d_{i,j} y_{i,j} \leq b_i$$

where for each worker w_i, x_i and y_i are the binary assignment vectors of the global and local phases of STAC, respectively. The first constraint, $|y_i - x_i|$, is the Hamming distance between x_i and y_i, which is bounded by a threshold ϵ aiming to keep minimum changes in the local assignment. The second constraint ensures that w_i's contribution to the task coverage is not decreased when compared to his contribution in the global phase. In the same fashion, L-STAC is NP-hard by reduction from the minimum set cover problem; thus, another greedy algorithm has been proposed to solve L-STAC.

Recently, Hu et al. [10] extended the travel budget constraint in [22] to a *spatial region*, represented by a rectangle R, within which the worker is willing to travel. Similar to [12], workers employ the peer-to-peer cloaking technique [3] to cloak their locations among $k - 1$ other workers. Also, each worker's cloaking area must contain his spatial region R, otherwise the cloaking area is extended to cover R. Observing that workers' cloaking areas often contain multiple spatial regions of other workers, to reduce the communication overhead, only some cloaking areas that could cover all the workers' spatial regions will be sent to the server. This technique limits the disclosure of information when compared to sending all the workers' cloaking areas to the server [22].

The cloaking techniques used in [10, 22] are intuitive; nevertheless, their privacy guarantee is weak. Such obfuscation-based techniques do not provide rigorous privacy protection and are prone to homogeneity attack [18] when all k workers are at the same location. Also, the value k needs to be specified to guarantee the desired level of privacy protection. Unfortunately, choosing an appropriate k value can be difficult because k-anonymity does not consider the frequency of user visits. To elaborate, a location may be visited by many workers—those who have a dominant contribution to the location (i.e., home or office) are most likely to be the subject of attack. Consequently, one with a background knowledge of who visits the location the most can easily perform such an attack.

7.4.2.2 Perturbation Techniques

Methods in this category use differential privacy (DP) to protect workers' locations during task assignment [8, 31–33, 38], which overcomes the aforementioned issues of the obfuscation technique. DP has emerged as the de facto standard with strong protection guarantees rooted in statistical analysis. It provides a *semantic* privacy model as opposed to a *syntactic* model in other sanitization techniques (e.g., k-anonymity, l-diversity). DP has been adopted by major industries for various tasks without compromising individual privacy, e.g., discovering users' usage patterns with Apple [9] or crowdsourcing statistics from end-user client software with Google [7]. DP ensures that an adversary is not able to reliably learn from the published sanitized data whether or not a particular individual is present in the original data, regardless of the adversary's prior knowledge.

The authors in [32] propose system model, privacy model and performance metrics, followed by two main steps that preserve privacy and identity of workers: *sanitization* of workers' locations and *task assignment* on the sanitized data.

System Model

To protect location privacy of workers participating in spatial tasks, the server must only have access to data sanitized according to ϵ-*differential privacy* [6] (ϵ is privacy loss or privacy budget). Figure 7.6a shows the system architecture. Workers send their locations (Step 0) to a trusted *cellular service provider* (CSP) which collects updates and releases a *private spatial decomposition* (PSD) according to privacy budget ϵ mutually agreed upon with the workers. The PSD is accessed by the server (Step 1), which also receives tasks from a number of requesters (Step 2). When the server receives a task t, it queries the PSD to determine a *geocast region (GR)* that encloses with high probability workers close to t. Next, the server initiates a *geocast* communication [19] process (Step 3) to disseminate t to all workers within *GR*. According to DP, sanitizing a dataset requires the creation of fake locations in the PSD. If the server is allowed to directly contact workers, then failure to establish

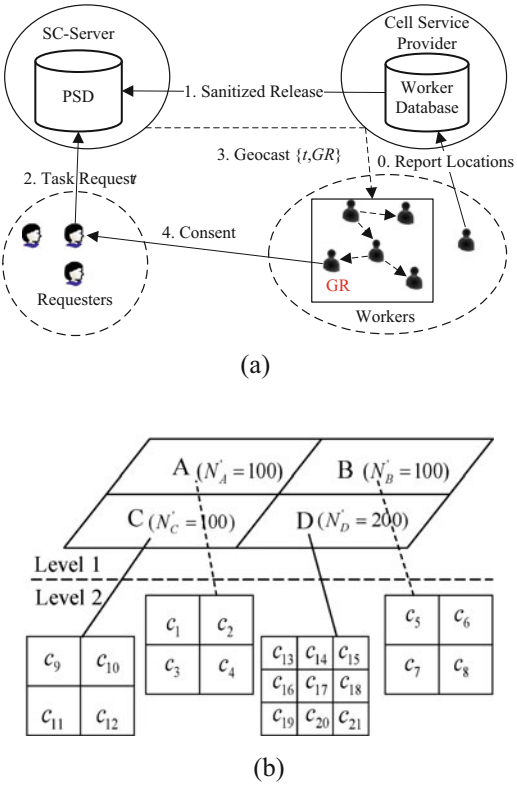

Fig. 7.6 Differentially private framework for spatial crowdsourcing. (**a**) System architecture. (**b**) Worker PSD using adaptive grid

a communication channel would breach privacy, as the server is able to distinguish fake workers from real ones. Using geocast is a unique feature of the framework which is necessary to achieve privacy protection. Geocast can be performed either with the help of the CSP infrastructure, or through a mobile ad hoc network where the CSP contacts a single worker in the *GR*, and then the message is disseminated on a hop-by-hop basis to the entire *GR*. The latter approach keeps CSP overhead low and can reduce operation costs for workers. Upon receiving request t, a worker w_i decides whether to perform the task or not. If yes (Step 4), she sends a *consent* message to the server (or requesters) confirming w_i's availability. If w_i is not willing to participate in the task, then no consent is sent, and no information about the worker is disclosed.

Privacy Model and Assumptions

The objective of the framework is to protect both the *location* and the *identity* of workers *during task assignment*. Once a worker consents to a task, the worker herself may directly disclose information to the task requester (e.g., to enable a communication channel between worker and requester). However, such additional disclosure is outside the scope of this work, as each worker has the right to disclose his or her individual information. Instead, the focus of the framework is on what happens prior to consent, when worker location and identity must be protected from *both* task requesters and the server. This privacy model is a weaker version of the restrictive model in Fig. 7.1a since task locations are public.

Workers cannot trust the server, especially as there may be many such entities with diverse backgrounds, e.g., private companies, non-profits, government organizations, academic institutions. On the other hand, the CSP already has a signed agreement with workers through the service contract, so there is already a trust relationship established, as well as mutually-agreed upon rules for data disclosure. Furthermore, the CSP already knows where subscribers are, e.g., using cell tower triangulation, so worker location reporting does not introduce additional disclosure. In addition, having the CSP expose a PSD release of the user location dataset can benefit applications beyond crowdsourcing. For instance, the PSD can be shared with law enforcement agencies for public safety, or with commercial organizations to increase the revenue of the CSP. Therefore, there is sufficient motivation for the CSP to provide such a location sanitization service.

However, the CSP has no expertise, and perhaps no financial interest, to host an SC service, which needs to deal with a diverse set of issues such as interacting with various task requester categories, managing profiles (e.g., some workers may only volunteer for environmental tasks), etc. The role of the CSP is to aggregate locations from subscribed workers, transform them according to DP, and release the data in sanitized form to one or more servers for assignment. As multiple servers can use the same PSD, it is practical for the CSP to provide PSDs for a small fee, e.g., a percentage of the workers' payment, or a tax incentive in the case of a public-interest SC application.

Design Goals and Performance Metrics

Protecting worker location significantly complicates task assignment and may reduce the effectiveness and efficiency of worker-task matching. Due to the nature of DP, it is possible for a region to contain no workers, even if the PSD shows a positive count. Therefore, no workers (or an insufficient number thereof) may be notified of the task request, and the task may not be completed. Alternatively, the GR may comprise workers who are a long distance away from the task location, whereas nearer workers are not included. Finally, in the non-private case, only one selected worker, whose location and identity is known, is notified of the task request. With location protection, redundant messages need to be sent, increasing overhead. We focus on the following performance metrics:

- *Assignment success rate (ASR).* Due to PSD data uncertainty, the server may incorrectly assign workers to tasks (e.g., no worker is reached, or task is too far and workers do not accept it). *ASR* measures the ratio of tasks accepted by a worker to the total number of task requests.
- *Worker travel distance (WTD).* The server is no longer able to accurately evaluate worker-task distance, hence workers may have to travel long distances to tasks. The challenge is to keep the worker travel distance low, even when exact worker locations are not known.
- *System overhead.* Dealing with imprecise locations increases the complexity of assignment, which poses scalability problems. A significant metric to measure overhead is the *a*verage number of *n*otified *w*orkers (*ANW*). This number affects both the *communication overhead* required to geocast task requests, as well as the *computation overhead* of the matching algorithm, which depends on how many workers need to be notified of a task request.

Sanitization of Workers' Locations Using Adaptive Grid

The first step in the proposed framework consists of building a PSD (at the CSP side) to be used later for task assignment at the server. Building the PSD is an essential step because it determines how accurate the released data is, which in turn affects *ASR*, *WTD* and *ANW*. Worker location data are sanitized at the CSP using a PSD, named *adaptive grid* (AG) [23]. PSD is a sanitized spatial index, where each index node contains a noisy count of the workers rooted at that node. Figure 7.6b shows a snapshot of an adaptive grid with four level-1 cells A,B,C,D. Constructing a differentially private AG requires two steps. First, the noisy counts N' of A,B,C,D are computed by adding calibrated random Laplace noise [6]. Second, based on the noisy counts, level-1 cells are further split into level-2 cells. Cell D, which has a higher noisy count of 200 is partitioned according to a 3×3 grid, while the granularity for other cells is 2×2. Thereafter, AG adds to each level-2 cell (c_i,

$i = 1 \ldots 21$) calibrated random Laplace noise. Finally, their corresponding noisy counts n_{c_i} are published together with the structure of the AG.

Although AG yields small errors for general spatial queries, it is not directly applicable to SC due to its rigidity in choosing parameters. Specifically, the granularity m_2 of the level-2 grid is too coarse, leading to large geocast areas and high communication overhead. Thus, the AG method is extended to address the specific requirements of the SC framework. Particularly, a heuristic is proposed to increase the granularity m_2 in order to decrease overhead, but only to the point where there is at least one worker in a cell [32].

Task Assignment on Sanitized Data

On top of the noisy data, to ensure that task assignment has a high success rate, analytical models that consider task completion rate, worker travel distance and system overhead are developed. When a request for a task t is posted, the server queries the PSD and determines a geocast region *GR* where the task is disseminated. The goal is to obtain a high success rate for task assignment, while at the same time reducing the worker travel distance *WTD* and request dissemination overhead *ANW*.

Acceptance Rate and Analytical Utility Model

Travel distance is critical in SC, as workers need to physically visit the task locations. A worker is more willing to accept nearby tasks [13], so acceptance rate is modeled as a decreasing function of travel distance. Also, we denote by *acceptance rate (AR)* the probability $p^a (1 \leq p^a \leq 1)$ that a worker agrees to complete a task for which he has received a request. Thereafter, an analytical *utility* model is developed that allows the server to quantify the probability that a task request disseminated in a certain *GR* is accepted by a worker. Intuitively, the utility depends on the AR and on the worker count $\bar{w}$ estimated to be enclosed within the *GR*. A server will typically establish an *expected utility* threshold *EU* which is the targeted success rate for a task (this is a system goal, rather than an outcome). Generally, *EU* is considerably larger than an individual worker's p^a, so the *GR* must contain multiple workers.

We define X as a random variable for the event that a worker accepts a received task: $P(X = True) = p^a$ and $P(X = False) = 1 - p^a$. Assuming w independent workers, $X \sim Binomial(w, p^a)$. We define the *utility* of a geocast region covering w workers as:

$$U = 1 - (1 - p^a)^w \tag{7.1}$$

U measures the probability that at least one worker accepts the task. The utility definition can be extended to the case of redundant task assignment, where multiple workers are required to complete a task [31].

Geocast Region Construction

The third step in the framework is the construction and dissemination of GR. By the nature of the DP protection model, fake entries may need to be created in the worker PSD. Thus the server cannot directly contact workers, not even if pseudonyms are used, as establishing a network connection to an entity would allow the server to learn whether an entry is real or not, and this breach privacy. To address this challenge, the geocast mechanism was introduced for the task request dissemination. Geocast is a routing and addressing method, which is used to deliver information to all devices situated within a geographical area. Once a PSD partition is identified by the analytical model outlined above, the task request is geocast to all the workers within that partition.

Particularly, given task t, the GR construction algorithm must balance two conflicting requirements: determine a region that (1) contains sufficient workers such that task t is accepted with high probability, and (2) the size of the geocast region must be small. The input to the algorithm is task t as well as the worker PSD, consisting of the two-level AG with a noisy worker count for each grid cell. The algorithm chooses as initial *GR* the level-2 cell that covers the task, and determines its U value. As long as utility is lower than threshold EU, it expands the *GR* by adding neighboring cells. Cells are added one at a time, based on their estimated increase in *GR* utility. Following the task localness property, we take into account the distance of each candidate neighboring cell to the location of t, and give priority to closer cells. The algorithm stops either when the utility of the obtained *GR* exceeds threshold EU, or when the size of *GR* is larger than a particular threshold; hence, utility can no longer be increased. The *GR* construction algorithm is a greedy heuristic, as it always chooses the candidate cell that produces the highest utility increase at each step. The experimental results show that workers' location privacy is protected without compromising performance, and the extra travel cost is tolerable—a 20% increase when compared to the non-private case.

Next, we present various extensions of the worker PSD, followed by an approach toward PSD for moving workers.

Extensions and Enhancements of Worker PSD

There have been recent studies that adopt the privacy model used in [32], assuming a trusted CSP and differentially private location sanitization. Particularly, Gong et al. [8] propose a framework that can protect the workers' location privacy when allocating tasks to the workers. Similar to [32], they develop analytical models and task allocation strategies that balance privacy, utility, and system overhead. In [8], the CSP not only aggregates workers' locations but also their reputation information, which is used to provide quality control over the reports. Consequently, a new structure called reputation-based PSD is proposed to partition the space based on both reputation and location information.

Another work studies reward-based spatial crowdsourcing that enables task assignment with optimized reward allocation (Zhang et al. [38]). The authors also reuse the privacy framework introduced in [32], in which the server and workers are connected by a trusted CSP. However, unlike [32] that uses the adaptive grid to releases a sanitized location view to the server, this study constructs a contour plot to represent the spatial distribution of workers aiming to introduce less noise than the prior technique. The contour plot is used to perform task assignment. The objective of task assignment is to find the minimum radius r to ensure that the *ASR* of a task is equal to *expected utility* threshold *EU*, i.e., the probability that at least one worker performs the task is no less than the threshold.

Protection for Dynamic Workers' Locations

Previous perturbation techniques [8, 32, 38] assume a static scenario where workers' locations do not change. However, SC systems receive continuous requests for task assignment. Hence, it is important to keep track of the whereabouts of *moving* workers and to release a *sequence* of worker PSDs that allow effective spatial task assignment over multiple timestamps. The challenge is that as workers move, new snapshots of sanitized worker locations must be disclosed to maintain task assignment effectiveness. However, access to sequential releases gives an adversary more powerful attack opportunities. To counter such threats, differential privacy requires more noise injection, which in the worst case may reach amounts that are proportional to the length of the released location history (i.e., the number of disclosed snapshots). Clearly, such large noise would render the data useless, since SC is likely to be a continuously offered service in practice. A recent study [31] extends [32] to address the challenge of moving workers by investigating privacy budget allocation techniques across consecutive releases, and employing post-processing techniques based on Kalman filters to reduce the inaccuracy introduced by addition of noise.

7.4.2.3 Encryption Techniques

In this section we discuss studies that use encryption-based approaches. In [27] the identity and location (i.e., IP address) of workers are hidden from TS through multiple Tor relays using Onion encryption. However, Tor does not try to protect against an attacker who can see or measure both traffic going into the Tor network and also traffic coming out of the Tor network—for example, the end-to-end timing correlation attack. Thus, to prevent TS from performing a timing attack by linking multiple task requests, the workers connect to TS at random intervals. Furthermore, during tasking the workers make sure that TS does not tamper the task request from RS; otherwise, the workers can report TS as fraudulent to RS.

Shin et al. [27], however, focus on the pull mode, which likely results in suboptimal task assignment. Therefore, a recent study [26] proposes a secure task-assignment protocol to protect worker location privacy in the push mode. The privacy framework used in [26] is similar to [32] (Fig. 7.6a), except the CSP is replaced by a *privacy service provider* (PSP)—a semi-honest (i.e., honest-but-curious) third party to provide privacy functionality and collect encrypted data from workers, including encrypted location reports. With the framework, the server needs to perform worker-task matching in the encrypted domain. Particularly, given a task, the server communicates with PSP in the encrypted domain to find the worker with minimum travel cost to the task. The travel cost is evaluated in terms of worker-task distance and the degree of interest of the worker to the task.

The advantage of the proposed protocol is twofold. The framework is not relying on a trusted-third-party and is robust to semi-honest adversaries. Also, the privacy guarantees hold for moving workers. However, when compared to the cloaking and perturbation techniques, cryptographic-based approaches may incur higher computation overhead. In addition, the semi-honest adversary model is restrictive in terms of privacy protection and may not always hold in the real-world SC applications. That is, server and PSP may not follow the specified protocol, or requesters can be malicious. Thus, a stronger privacy protocol that is resilient to malicious adversary model needs to be developed.

7.5 Conclusion and Future Directions

With the popularity of mobile devices, spatial crowdsourcing is rising as a framework that enables human workers to solve tasks in the physical world. With spatial crowdsourcing, requesters outsource a set of spatiotemporal tasks to a set of workers, i.e., individuals with mobile devices that perform the tasks by physically traveling to the specified locations of interest. However, current solutions require a worker to disclose his location to the server and/or to other requesters even before accepting a task—or a requester to disclose his tasks' locations, which can be used to infer his own location, to untrusted entities. In this chapter we identified the privacy threats to both workers and requesters in the two main phases of crowdsourcing: task assignment and task reporting.

We surveyed some of the most notable solutions proposing various privacy techniques, ranging from pseudonym, cloaking, perturbation to exchange-based and encryption-based approaches. These studies have shown encouraging results in protecting the privacy of both workers or requesters in spatial crowdsourcing. However, protecting the privacy of workers and requesters simultaneously using rigorous privacy guarantees such as differential privacy is still an open problem. Another promising direction is to consider powerful adversaries with knowledge about temporal correlations of a moving user's locations or collusion between workers and the server; for example, some workers may work for the SC company or the company may use driverless cars.

References

1. iRain: new mobile app to promote citizen-science and support water management: http://en.unesco.org/news/irain-new-mobile-app-promote-citizen-science-and-support-water-management, 2016.
2. I. Boutsis and V. Kalogeraki. Privacy preservation for participatory sensing data. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 103–113. IEEE, mar 2013.
3. C.-Y. Chow, M. F. Mokbel, and X. Liu. Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments. *GeoInformatica*, 15(2):351–380, 2011.
4. D. Christin. Privacy in mobile participatory sensing: Current trends and future challenges. In *Journal of Systems and Software*, volume 116, pages 57–68, 2016.
5. D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst. - SIGSPATIAL'13*, pages 314–323, New York, New York, USA, 2013. ACM Press.
6. C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.
7. Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *SIGSAC*, pages 1054–1067. ACM, 2014.
8. Y. Gong, C. Zhang, Y. Fang, and J. Sun. Protecting Location Privacy for Task Allocation in Ad Hoc Mobile Cloud Computing. In *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2015.
9. A. Greenberg. Apple's "differential privacy' is about collecting your data - but not your data. https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/, 2016.
10. J. Hu, L. Huang, L. Li, M. Qi, and W. Yang. Protecting Location Privacy in Spatial Crowdsourcing. In *Asia-Pacific Web Conference*, pages 113–124. Springer International Publishing, 2015.
11. L. Hu and C. Shahabi. Privacy assurance in mobile sensing networks: Go beyond trusted servers. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2010*, pages 613–619, 2010.
12. L. Kazemi and C. Shahabi. A privacy-aware framework for participatory sensing. In *ACM SIGKDD Explorations Newsletter*, volume 13, page 43. ACM, aug 2011.
13. L. Kazemi and C. Shahabi. GeoCrowd: Enabling Query Answering with Spatial Crowdsourcing. In *Proc. 20th Int. Conf. Adv. Geogr. Inf. Syst. - SIGSPATIAL '12*, number c, page 189, 2012.
14. L. Kazemi, C. Shahabi, and L. Chen. GeoTruCrowd: Trustworthy Query Answering with Spatial Crowdsourcing. In *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst. - SIGSPATIAL'13*, pages 304–313, 2013.
15. S. H. Kim, Y. Lu, G. Constantinou, C. Shahabi, G. Wang, and R. Zimmermann. Mediaq: mobile multimedia management system. In *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 224–235. ACM, 2014.
16. A. Liu, Z.-X. Li, G.-F. Liu, K. Zheng, M. Zhang, Q. Li, and X. Zhang. Privacy-preserving task assignment in spatial crowdsourcing. *Journal of Computer Science and Technology*, 32(5):905–918, 2017.
17. B. Liu, L. Chen, X. Zhu, Y. Zhang, C. Zhang, and W. Qiu. Protecting location privacy in spatial crowdsourcing using encrypted data. In *EDBT*, pages 478–481, 2017.
18. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.
19. J. C. Navas and T. Imielinski. Geocast: geographic addressing and routing. In *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, pages 66–76. ACM, 1997.

20. A. Pham, I. Dacosta, B. Jacot-Guillarmod, K. Huguenin, T. Hajar, F. Tramèr, V. Gligor, and J.-P. Hubaux. Privateride: A privacy-enhanced ride-hailing service. *Proceedings on Privacy Enhancing Technologies*, 2017(2):38–56, 2017.
21. L. Pournajaf, D. A. Garcia-ulloa, L. Xiong, and V. Sunderam. Participant Privacy in Mobile Crowd Sensing Task Management : A Survey of Methods and Challenges. In *SIGMOD Record*, volume 44, pages 23–34. ACM, may 2015.
22. L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka. Spatial task assignment for crowd sensing with cloaked locations. In *Proceedings - IEEE International Conference on Mobile Data Management*, volume 1, pages 73–82. IEEE, jul 2014.
23. W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 757–768. IEEE, 2013.
24. D. Reinhardt and F. Dürr. Opportunities and risks of delegating sensing tasks to the crowd. In *Handbook on Mobile Data Privacy*. Springer, 2017.
25. J. Scheck. Stalkers exploit cellphone GPS. http://www.wsj.com, 2010.
26. Y. Shen, L. Huang, L. Li, X. Lu, S. Wang, and W. Yang. Towards preserving worker location privacy in spatial crowdsourcing. In *2015 IEEE Global Communications Conference, GLOBECOM 2015*, 2016.
27. M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos. AnonySense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing*, 7(1):16–30, 2011.
28. Y. Sun, A. Liu, Z. Li, G. Liu, L. Zhao, and K. Zheng. Anonymity-based privacy-preserving task assignment in spatial crowdsourcing. In *International Conference on Web Information Systems Engineering*, pages 263–277. Springer, 2017.
29. L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
30. H. To, L. Fan, L. Tran, and C. Shahabi. Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints. In *2016 IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2016*, pages 1–8. IEEE, mar 2016.
31. H. To, G. Ghinita, L. Fan, and C. Shahabi. Differentially Private Location Protection for Worker Datasets in Spatial Crowdsourcing. In *IEEE Transactions on Mobile Computing*, volume PP, pages 1–1, 2016.
32. H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. In *Proceedings of the VLDB Endowment*, volume 7, pages 919–930. VLDB Endowment, jun 2014.
33. H. To, G. Ghinita, and C. Shahabi. PrivGeoCrowd: A toolbox for studying private spatial Crowdsourcing. In *Proceedings - International Conference on Data Engineering*, volume 2015-May, pages 1404–1407. IEEE, apr 2015.
34. H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems*, 1(1):2, 2015.
35. K. Vu, R. Zheng, and J. Gao. Efficient algorithms for K-anonymous location privacy in participatory sensing. In *Proceedings - IEEE INFOCOM*, pages 2399–2407, 2012.
36. G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao. Defending against Sybil Devices in Crowdsourced Mapping Services. *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '16*, pages 179–191, 2016.
37. B. Zhang, C. H. Liu, J. Lu, Z. Song, Z. Ren, J. Ma, and W. Wang. Privacy-preserving QoI-aware participant coordination for mobile crowdsourcing. In *Computer Networks*, volume 101, pages 29–41, 2016.
38. L. Zhang, X. Lu, P. Xiong, and T. Zhu. A Differentially Private Method for Reward-Based Spatial Crowdsourcing. In *International Conference on Applications and Techniques in Information Security*, pages 153–164. Springer Berlin Heidelberg, 2015.