Optimal Task Recommendation for Mobile Crowdsourcing with Privacy Control

Yanmin Gong, *Student Member, IEEE,* Yuguang Fang, *Fellow, IEEE,* and Yuanxiong Guo, *Member, IEEE*

Abstract—Mobile crowdsourcing (MC) is a transformative paradigm that engages a crowd of mobile users (i.e., *workers*) in the act of collecting, analyzing, and disseminating information or sharing their resources. To ensure quality of service, MC platforms tend to recommend MC tasks to workers based on their context information extracted from their interactions and smartphone sensors. This raises privacy concerns that are hard to solve due to the constrained resources available on mobile devices. In this paper, we point out fundamental trade-offs among three criteria–utility, privacy, and efficiency–in a MC system and propose a flexible optimization framework that can be adjusted to any desired trade-off point with the joint efforts of MC platform and workers. We show that the underlying optimization problems are NP-hard and present efficient approximation algorithms to solve them. Since worker statistics are required for tuning the previous optimization models, we design an efficient aggregation protocol to collect worker feedbacks while providing differential privacy guarantees. Both numerical evaluations and performance analysis are conducted to show the effectiveness and efficiency of the proposed framework.

Index Terms—Mobile crowdsourcing, task recommendation, differential privacy.

1 Introduction

Mobile crowdsourcing (MC) is the combination of crowdsourcing and mobile technologies that leverages the sensing, computing, and communication capabilities of mobile devices to provide crowdsourcing services. According to the statistics given by Statista [1], the number of smartphones sold to end users worldwide is 1.2 billion in the single year of 2014. New smartphones are usually equipped with various sensors including GPS units, accelerometers, gyroscopes, and touch screen sensors, which can provide a wide range of sensing data. The ubiquity and advanced sensing capabilities of mobile devices enable mobile users to gather rich information everywhere/anytime, and therefore to perform tasks that can hardly be completed by web-based crowdsourcing. In MC, a crowd of mobile users are engaged to provide pervasive and cost-effective services of data collecting, processing, and computing. These mobile users have shifted from the traditional role of service consumers to the new role of service providers, and they usually collect a small fee (or other forms of reward) for providing services. The applications of mobile crowdsourcing have developed rapidly. Existing commercial MC applications include traffic monitoring (e.g., Waze [2]), ride sharing (e.g. Uber [3]), environmental monitoring (e.g., Stereopublic [4]), and wireless coverage mapping (e.g. OpenSignal [5]). Nonetheless, MC is still in its infancy, and there are many undergoing research exploring applications such as epidemics monitoring and prediction [6] and urban sensing [7].

In MC, a spatio-temporal task is outsourced to a group of mobile users (i.e., *workers*) who perform the task within a deadline, and only workers under certain contexts are qualified for the task. However, it is quite inefficient for workers to select tasks by themselves when there are a huge number of crowdsourcing tasks, especially on a mobile device due to its limited screen and keyboard. Hence, MC platforms must provide task recommendation services which proactively push a task to qualified workers. In current solutions, workers have to reveal their exact contexts to MC platforms in order to receive personalized task recommendation.

Depending on the application scenario, the context of a worker can be defined along multiple dimensions, including geographical (e.g., along a street), temporal (e.g., within hours), activity (e.g., moving speed), and profile (e.g., gender) [8]. These contexts contain private and sensitive information that may be used to uniquely identify an individual, reveal his/her health status, or track his/her daily routines. However, the MC platforms are potentially untrustworthy in the sense that they may be operated by various organizations and companies and may also be compromised by malicious adversaries. Hence, allowing the MC platforms to learn exact contexts may put worker privacy at risk [9]. It is imperative to protect worker privacy in order to enable the large-scale deployment of mobile crowdsourcing applications.

An MC system has three components that may reveal private worker information: *offline statistics collection* to learn recommendation rules based on worker contexts and historical task completion performance, *online task selection* to select the most suitable tasks to a worker based on his current context, and *task completion* for a worker to accept and perform a task, and to return the result back. Each component exposes worker contexts and raises privacy concerns in different ways. In this paper, we focus on the privacy issue of the first two components due to the following reasons. First, context disclosure is more severe during task recommendation, which consists of the first two components, from the standpoint of the disclosure volume since all workers are candidates for task recommendation, while only few workers are eventually involved in the *task completion* component. Second, in the *task completion* component, a worker sends explicit consent

- *Y. Gong and Y. Fang are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. E-mail: {ymgong@, fang@ece.}ufl.edu.*
- *Y. Guo is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA. E-mail: richard.guo@okstate.edu.*

A preliminary version of this work has been published in IEEE GLOBECOM'14.

to perform a task, and context disclosure is unavoidable since merely performing a task indicates that he/she is in the required context. Note that identity protection during *task completion* could be provided through anonymous routing or pseudonyms such as Tor, and it is not the focus of this paper.

In this paper, we propose a framework for protecting privacy of worker contexts while enabling effective task recommendation in MC systems. Specifically, our proposed framework contains two main components that may operate in parallel: *privacy-aware online task selection* which selects the best MC tasks for workers based on their current noisy contexts, and *privacy-preserving offline statistics collection* which aggregates historical information about worker contexts and task completion activities needed for task selection while preserving worker privacy.

Privacy-Aware Online Task Selection. Current MC systems select tasks by collecting personal data at a server. Workers have to reveal their exact context information to the server in order to participate. To address the privacy concerns of such *server-only* recommendation, an alternative approach would be *worker-only*, where workers' mobile devices keep their own personal context information and perform recommendation. Indeed, it has been proposed for personalization in mobile advertising systems [10]. The problem with this approach is the huge computation and communication overhead for resource-constrained mobile devices. Thus, some recent papers propose hybrid solutions that jointly consider both sides to address privacy issues in mobile systems [11]–[14]. For example, in [11], the server returns a superset of the results and let end users to filter useful information by themselves. These solutions have a variety of optimization goals, which motivates us to consider the fundamental trade-offs in these mobile systems.

In this paper, we formulate the task selection from a MC server to a worker as an optimization problem that considers three criteria: (1) privacy that is related to the amount of the worker's context information shared with the MC server, (2) utility that represents the benefits of recommending the tasks in terms of relevance or revenue, and (3) efficiency that measures the communication and computation overhead imposed on the worker's mobile devices by recommending a certain amount of tasks. We show in Section 3 that these three criteria can not be optimized simultaneously. Note that the aforementioned solutions only present some discrete trade-off points: recommendation only at the server side provides best efficiency and utility at the cost of privacy, while recommendation at the worker side provides privacy guarantees and utility at the cost of efficiency. In contrast to them, we propose an optimization model that can be adjusted to any desirable trade-off point. In the proposed optimization framework, a worker can decide how much information about his context to share with the MC server. Based on this limited information, the MC server selects and sends a set of tasks to the worker. The size of the task set is pre-defined by the worker considering the associated communication and computation overhead. After the worker receives the task set, he picks and completes the best task based on his private information. The most challenging part in the whole process is to select the task set sent by the MC server that maximizes the total expected utility of the MC server given constraints on privacy and efficiency. There are also other trade-off points we can consider, such as jointly optimizing utility and efficiency given a constraint on privacy. Since the priorities of privacy and efficiency criteria can be arbitrarily selected by the worker, the framework is quite flexible and can be used in different MC systems.

Privacy-Preserving Offline Statistics Collection. Recommended tasks are chosen based on statistics including both historical performance of workers and the distribution of their contexts. These statistics are collected offline and are used to calibrate the online task selection component. However, estimating these statistics often poses a privacy challenge: workers may be unwilling to reveal the required information such as their exact contexts and tasks that they have completed successfully. Therefore, we need to provide a privacy-preserving solution that can estimate these statistics from distributed worker data. Some previous works propose to address privacy issues in statistical queries by anonymizing data; however, there are possibilities that data owners may be de-anonymized with auxiliary information [15], [16]. Another approach, differential privacy, adds noise in the querying results of statistical databases so that even with auxiliary information, one can not infer the presence or absence of individuals. Although differential privacy has become popular recently, most of the solutions are proposed for a centralized database, which is not suitable for our case in which data are distributed among workers. On the other hand, existing distributed solutions [17]–[20] are impractical for large systems. For example, the computation cost per user in [18] is $\mathcal{O}(M)$ where M is the number of users, which becomes prohibitive for a large population of users. The computation cost is reduced to $\mathcal{O}(1)$ in [19], [20], but they use an expensive secret sharing protocol that is not scalable to a large group of workers. Additionally, the solution in [21] uses two servers to collaboratively calculate statistics and handle user dynamics. However, all these solutions can not prevent a single malicious user from greatly distorting the querying results.

In this paper, we provide a privacy-preserving statistics collection protocol to reliably compute the required statistics from a dynamic set of workers who are potentially malicious. Our solution relies on a semi-honest third party, a *proxy*, which provides differential privacy guarantees by adding blind noise to the encrypted worker data. Worker data are encrypted by workers with the public key of MC server and are sanitized by the proxy. Hence neither the proxy nor the MC server can learn the accurate statistics or individual worker data.

In summary, the main contributions of this paper are as follows.

- We identify the specific privacy challenges of task recommendation in MC systems, and we propose a framework that protects worker context privacy. To the best of our knowledge, this is the first work to study privacy in task recommendation for MC.
- We propose an optimization model for task selection that explores fundamental trade-offs among three design criteria–privacy, utility, and efficiency–in MC systems, and we present efficient approximation algorithms to solve it.
- We introduce an efficient statistics collection protocol that preserves differential privacy in a distributed setting with tolerance of malicious or dynamic workers.
- We conduct both numerical evaluations and performance analysis to show the effectiveness and efficiency of our proposed framework.

The remainder of this paper is organized as follows. We present our framework in Section 2. Next we represent the task selection process as a constrained optimization problem in Section 3. Section 4 develops an approximation algorithm to

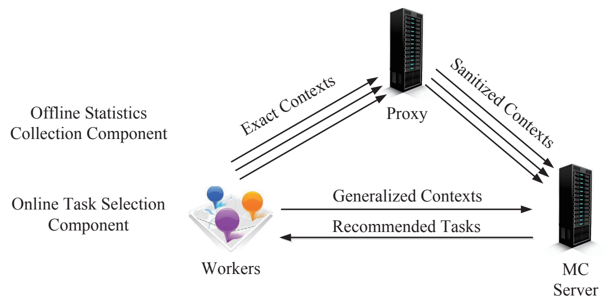


Fig. 1. Basic system model for task recommendation in MC.

solve the optimization problem. A privacy-preserving protocol for statistics collection is presented in Section 5. We discuss the experimental results and analyze the system overhead in Section 6 and Section 7, respectively. Section 8 summarizes related work. Finally, we conclude the paper in Section 9.

2 The Proposed Framework

In this section, we describe the basic system model for task recommendation in MC systems and design goals.

2.1 System Model

Fig. 1 shows the basic model of the proposed framework consisting of the following two components:

- *Statistics Collection.* In this component, the server collects various statistics from workers periodically in the background. A semi-honest third party (to be elaborated later in Section 5) is employed to protect the private context information of participating workers.
- *Task Selection.* In this component, based on the statistics collected in the statistics collection component and worker's current context, the server selects and delivers a set of tasks to the worker. Note that we allow workers to decide how much private information they are willing to share with the server. The server selects a set of tasks, where the set size is constrained by a bounded communication overhead, based on this limited information and sends them to the worker. The worker [1] then chooses the most relevant one to complete based on all his private information and returns the answer to task requesters.

Privacy Guarantees. Our framework can protect worker privacy in both online task selection and offline statistics collection. Note that task selection and statistics collection use private worker contexts in different ways, and therefore require different privacy-preserving techniques.

In *task selection*, a single worker's current context is used, and we ensure worker privacy through limited information disclosure as used in many mobile systems [13], [22], [23]. We allow the worker to share a generalized context with the server rather than his exact context. The generalization of worker context is done according to a predefined hierarchy. Quantifiable contexts such as location can be simply divided into different intervals based on their values. For instance, location information represented

1. For brevity, we use "he" to refer to the worker without meaning any distinctions about the worker's gender in the rest of the paper.

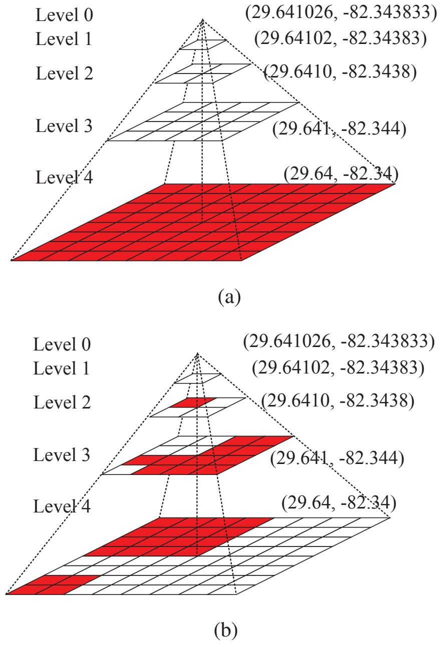


Fig. 2. (a) A basic generalization approach; (b) An adaptive generalization approach.

by the latitude and longitude with a total of 6 decimal digits can be generalized by keeping $6 - a$ decimal digits for level-a generalization. An example of level-4 generalization is shown in Fig. 2(a). A worker can also choose different (i.e., adaptive) levels of generalization for different intervals of contexts with existing approaches [11], as illustrated in Fig. 2(b). With the adaptive generalization approach, the worker can protect his location privacy at a relatively low cost of utility. If a context information is not quantifiable such as activities, it can be described in a tree taxonomy. Fig. 3 shows an example describing activities with different precisions. In this case, if a worker dining in a restaurant chooses level-2 generalization, he would only tell the server that he is static. In this paper, we focus on the challenge of recommending tasks based on the generalized context rather than how to generalize different contexts. Interesting readers may refer to [24] for details about different generalization methods.

In *statistics collection*, historical context and task completion information from workers is used. A worker can choose whether to participate in the statistics collection protocol or not. If he decides to participate, we ensure that no other party, except the worker himself, could know his private information during the statistics collection process. Moreover, we give the worker differential privacy guarantees [18], which ensures that the resulting statistics do not significantly change with the presence or absence of a single worker. Therefore, an adversary with arbitrary background knowledge cannot trace or de-anonymize a worker from multiple runs of the statistics collection protocol.

2.2 Design Goals

We aim to provide good privacy, utility, and efficiency in the proposed framework. Since task selection and statistics collection differ in nature, we describe their design goals separately.

Goals for task selection. We have three design goals for task selection: privacy, utility, and efficiency.

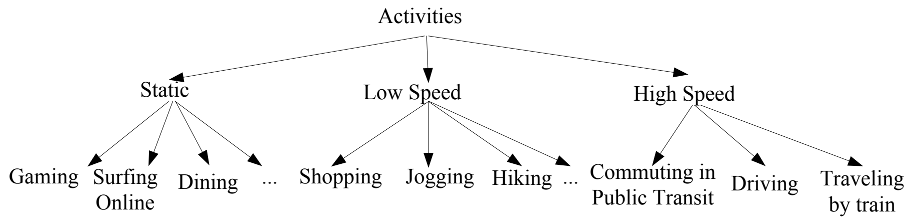


Fig. 3. Taxonomy of worker activities.

- *Privacy*. Worker contexts are needed for task recommendation, which may be leveraged by the server to uniquely identify an individual worker. To reduce the risk of being identified, the worker limits the information shared with the server. Instead of providing an exact context, the worker provides a generalized context which obfuscates privacy sensitive information such as location and activity.
- *Utility*. Utility represents the value of a set of recommended tasks. From the perspective of the server, the utility is the expected revenue (or commission) of the recommended tasks. From the perspective of the worker, the utility is reflected in the payment he would obtain from completing the recommended tasks. The utility for both stakeholders is related to the payment of the task that is selected and completed successfully by the worker.
- *Efficiency*. When a worker receives a set of recommended tasks, he tries to select the best task from the set. A larger set takes more time to select from, which contradicts the intention of recommendation. Thus the efficiency of task recommendation is directly related to the set size. The recommendation system should recommend a reasonable number of tasks at a time to ensure the efficiency of task selection by the worker.

Goals for Statistics Collection. There are mainly three design goals for statistics collection:

- *Privacy.* Worker privacy should be protected during statistics collection.
- *Robustness.* The final statistics should not be distorted largely by a small portion of malicious workers.
- *Scalability.* The system should be scalable to a large population of workers, which means that the proposed protocols should be highly efficient.

3 Optimization Model for Task Selection

In this section, we investigate fundamental trade-offs among three design goals and formulate two optimization problems to model them in the task selection component.

3.1 Definitions

Before proceeding further, we give the definitions for notations used in the rest of the paper as follows.

Definition 1. Contexts and Tasks

- Denote by $\mathcal{C} = \{c : c = 1, 2, \ldots, |\mathcal{C}|\}$ the set of all exact contexts. Each worker has an exact context c.
- Denote by $\hat{\mathcal{C}} = \{\hat{c} : \hat{c} = 1, 2, \ldots, |\hat{\mathcal{C}}|\}$ the set of all generalized contexts. Each exact context is mapped into a generalized context, and a generalized context may correspond to multiple detailed contexts.
- Denote by $\mathcal{T} = \{t : t = 1, 2, \ldots, |\mathcal{T}|\}$ the set of all tasks. For simplicity of notations, we treat tasks that have the same requirements for worker contexts and the same payment as one task. Each task may have multiple instances. The payment for successfully completing a task t is denoted as ρ_t.

Definition 2. Complete-and-Approve Rate (CAR): Both workers and the MC platform can earn some money when tasks are completed successfully (i.e., answers approved by task requesters). This can be characterized by the complete-and-approve rate (CAR), which can be calculated as N_1, the total number of workers with context c who have successfully completed task t, divided by N_2, the total number of workers with context c, i.e., $\text{CAR}(t|c) = N_1/N_2$.

3.2 Trade-Offs among Utility, Privacy and Efficiency

The optimization model of task selection specifies how to choose tasks based on limited information about a worker. There are three conflicting design goals in this model: utility, privacy, and efficiency. These three goals cannot be optimized simultaneously. First, suppose that privacy and efficiency are optimized, which means that the worker provides no context about himself to the system and expects to receive a single task tailored for him. In this case, as long as the utility of tasks varies across different contexts, it is impossible for the recommendation server to choose a task that is of high utility for the worker. Second, consider the case that efficiency and utility are optimized. In order to find a task that has the highest utility for the worker, the recommendation server needs to know the exact worker context, compromising his privacy. Finally, if we want to ensure the optimality of utility and privacy, the recommendation server needs to recommend, without any prior knowledge of worker context, a set of tasks within which the worker can find one to maximize his utility. In this case, the efficiency becomes suboptimal since the recommended task set would be very large. If any of the above three goals is dropped, it is trivial to optimize the other two. Therefore, in practice, we have to find a good trade-off among these three goals.

3.3 Optimization Problem Formulation

In our framework, the worker first decides the amount of information about his private context to share with the server. Based on this limited information, the server selects L tasks $T \subset \mathcal{T}$ and sends them to the worker. Here, L determines the efficiency. Then the worker selects a task from the recommended L tasks, completes it, and returns the result back to the sever or task requester. Therefore, the task is selected jointly by the server and the worker in our framework.

As mentioned before, there are three conflicting goals. Although these goals cannot be optimized simultaneously, there are several candidate objective functions that optimizes the goals from different aspects. In the following, we choose an optimization objective function representing the utility and model the other two goals as constraints. In other words, we optimize the utility while allowing the worker to determine the efficiency and privacy requirements. Alternative objective functions are also discussed.

3.3.1 Computation at the worker side

Given a set of recommended tasks T, the worker selects one to complete. The behavior of the worker is supposed to be rational. In other words, the worker with exact context c would select the task that maximizes his own revenue, which can be modeled as

$$t^* = \operatorname*{argmax}_{t \in T} \rho_t \cdot \mathrm{CAR}(t|c). \tag{1}$$

3.3.2 Computation at the server side

Since the worker knows his own context, he can easily make the selection by maximizing his revenue. This is not true for the server as it can only select tasks based on the limited information provided by the worker. To increase the relevance between recommended tasks and the worker, the server needs to recommend multiple tasks at the same time.

Assume that the server already has prior knowledge on the context-dependent click-and-approve rates, $\mathrm{CAR}(t|c)$, and the probability distribution over contexts. From the perspective of the server, its utility (i.e., revenue) depends on the task that the worker chooses, and we use the expected revenue of the set of tasks to quantify it. Since the server does not know the exact context c of the worker, it considers the probability of each of the exact contexts that generalize into $\hat{c}$ and calculates the expected revenue R of the set of tasks T as follows:

$$\mathbb{E}[R(T|\hat{c})] = \sum_{c:c\rightarrow\hat{c}} \Pr[c|\hat{c}] \cdot \alpha \cdot \max_{t \in T} \rho_t \cdot \mathrm{CAR}(t|c), \tag{2}$$

where α is the portion of revenue that the platform can obtain for each successful transaction. Let L denote the size of the task set. The server needs to select L tasks that maximize the expected revenue given a generalized context $\hat{c}$, i.e.,

$$T^* = \operatorname*{argmax}_{T\subseteq\mathcal{T}:|T|=L} \mathbb{E}[R(T|\hat{c})]. \tag{3}$$

3.3.3 Alternative Objectives

The above optimization model contains the extreme cases when task selection is taken solely at the server side ($L = 1$) or solely at the worker side ($L = |\mathcal{T}|$). For the former case, if the server recommends a single task based on a very generalized context provided by the worker, it is likely that the recommendation has a low utility. For the latter case, the server sends all the available tasks to the worker. The selection becomes inefficient, and the recommendation service is meaningless. Hence, the parameter L should be selected cautiously.

Instead of setting L as a predefined parameter, we can also include it as one of the design variables. This can be done by substituting $\mathbb{E}[R(T|\hat{c})] - \lambda \cdot L$ for the original objective $\mathbb{E}[R(T|\hat{c})]$ in (3), where λ is the weight of the efficiency metric L in the total objective function. As a result, the server selects a set of tasks that maximizes the new objective, i.e.,

$$T^* = \operatorname*{argmax}_{T\subseteq\mathcal{T}:|T|=L} \mathbb{E}[R(T|\hat{c})] - \lambda \cdot L. \tag{4}$$

In this way, the efficiency and the utility can be optimized jointly.

There are other options to model the utility as well. For example, we can incorporate the cost of a task into the objective such as time or other resources needed for completing a task. In this case, the selection process among a set of tasks for the worker becomes more complicated. A possible formulation might be $\max_{t\in T}(\rho_t - \mathrm{cost}_{t,c}) \cdot \mathrm{CAR}(t|c)$, where $\mathrm{cost}_{t,c}$ denotes the cost to complete task t by workers with context c. In addition, there might be a reservation wage w_r [25] below which the worker would not pick the task. Considering this, the process of task selection for a worker can be modeled as $\max_{t\in T}(\rho_t - \mathrm{cost}_{t,c}) \cdot \mathbf{1}_{\{\rho_t - \mathrm{cost}_{t,c} \geq w_r\}} \cdot \mathrm{CAR}(t|c)$.

4 SOLUTION ALGORITHMS

In this section, we propose algorithms for the server and the worker to optimize their objectives efficiently. We first consider the specific scenario which optimizes the objective of utility as in (2) and then discuss how to jointly optimize utility and efficiency as in (4).

4.1 Approximation Algorithm for Optimizing the Utility

The optimization part for the worker can be easily solved since he only needs to choose a task among L tasks, where L is usually designed to be a small number. However, it is nontrivial for the server to select L tasks from the entire task space $\mathcal{T}$. Actually, we have the following fact:

Proposition 1. *Given a generalized context $\hat{c}$, it is NP-hard to find a set of tasks T^* such that:*

$$T^* = \operatorname*{argmax}_{T\subseteq\mathcal{T}:|T|=L} \sum_{c:c\rightarrow\hat{c}} \Pr[c|\hat{c}] \cdot \alpha \cdot \max_{t \in T} \rho_t \cdot \mathit{CAR}(t|c). \tag{5}$$

Proof. We prove the NP-hardness of the task recommendation problem by providing a polynomial time reduction from the NP-hard maximum coverage problem:

INSTANCE: A universe $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, a family $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of subsets of $\mathcal{U}$ and a positive integer L. QUESTION: Find a subset $\mathcal{S}' \subseteq \mathcal{S}$ of size L, such that $|\cup_{S_i\in\mathcal{S}'} S_i|$ is maximized. This problem can be shown as follows:

$$\mathcal{S}'^* = \operatorname*{argmax}_{\mathcal{S}'\subseteq\mathcal{S}:|\mathcal{S}'|=L} \sum_{u_j\in\mathcal{U}} \mathbf{1}_{u_j\in\mathcal{S}'}.$$

We construct a corresponding instance of the task recommendation problem as follows: Let $\{c_j : c_j \rightarrow \hat{c}\}$ be the context set corresponding to the universe $\mathcal{U}$, where there is a context c_j that generalizes to $\hat{c}$ for each element $u_j \in \mathcal{U}$. The size of the context set equals the size of the universe $|\mathcal{U}|$. Corresponding to each subset $S_i \in \mathcal{S}$, define a task t_i such that $\mathrm{CAR}(t_i|c_j) = 1$ for all elements $u_j \in S_i$ and $\mathrm{CAR}(t_i|c_j) = 0$ for $u_j \notin S_i$. Moreover, we set $\Pr[c_j|\hat{c}] = 1/|\mathcal{U}|, \forall c_j \rightarrow \hat{c}$, $\rho_t = 1, \forall t$ and $\alpha = 1$. The instance of the task recommendation problem can be shown as follows:

$$T^* = \operatorname*{argmax}_{T\subseteq\mathcal{T}:|T|=L} \frac{1}{\mathcal{U}} \cdot \sum_{c_j\rightarrow\hat{c}} \max_{t\in T} \mathrm{CAR}(t|c_j).$$

It is trivial to show that any instance of the maximum coverage problem can be reduced in polynomial time to an instance of the task recommendation problem in the above way. An optimal solution to the above instance of task selection problem yields an optimal solution to the maximum coverage problem. Therefore, the task recommendation problem is at least as "hard" as the maximum coverage problem. Since the maximum coverage problem is known to be NP-hard, our problem is also NP-hard. □

Since the problem (5) is NP-hard, we propose a greedy algorithm as shown in Algorithm 1 below.

Algorithm 1 Greedy Algorithm for Profit Maximization

```
Input: $\mathcal{T}$, $\hat{c}$, $L$
Output: $T$
    // initialization
 1: $T \leftarrow \emptyset$;
 2: repeat
 3:     $t \leftarrow \mathrm{argmax}_{t\in\mathcal{T}}\, \mathbb{E}[R(T \cup t|\hat{c})] - \mathbb{E}[R(T|\hat{c})]$;
 4:     $T \leftarrow T \cup \{t\}$;
 5: until $|T| = L$
 6: return $T$
```

By repeatedly choosing a task that maximizes the utility improvement, the greedy algorithm can be proved to approximate the optimal value within $1 - 1/e$. Note that in [26], a greedy algorithm that solves the maximum coverage problem provides the same approximation ratio. However in their problem, the set either fully includes the element or not at all, while in our problem a task can partially matches the context, which complicates the problem and requires additional analysis. The proof of this approximation ratio for our approximation algorithm is given below.

Proposition 2. *The greedy algorithm approximates the optimal solution within a factor of* $1 - 1/e$.

Proof. Define a marginal utility function of adding set T' to T as follows:

$$f(T, T') = \mathbb{E}[R(T \cup T'|\hat{c})] - \mathbb{E}[R(T|\hat{c})].$$

The function $f(T, T')$ is submodular in the sense that $f(T_1, T') > f(T_2, T')$ for all sets $T_1 \subset T_2$. For $l = 1, 2, \ldots, L$, let $T_l = \{t_1, t_2, \ldots, t_l\}$ be the greedy solution constructed up to the end of the l-th stage; thus T_L is the final greedy solution returned. Similarly, let $T^*_L = \{t^*_1, t^*_2, \ldots, t^*_L\}$ be the optimal solution of any fixed order and $T^*_l = \{t^*_1, t^*_2, \ldots, t^*_l\}$ represents the first l tasks. Denote by $m(l) = \sum_{i=1}^{l} m_i$ the utility of T_l, where $m_l = f(t_l, T_{l-1})$ is the marginal utility by adding t_l to T_{l-1}. Similarly, denote by $m^*(l) = \sum_{i=1}^{l} m^*_i$ the utility of T^*_l, where $m^*_l = f(t^*_l, T^*_{l-1})$. Our aim is to prove

$$m(L) \geq m^*(L) \cdot (1 - 1/e). \tag{6}$$

To this end, we first prove

$$m_l \geq (m^*(L) - m(l-1))/L, \forall l \in [1, L]. \tag{7}$$

The marginal utility of adding set T^*_L to set T_{l-1} is $f(T_{l-1}, T^*_L)$, which equals $\sum_{i=1}^{l} f(T_{l-1} \cup T^*_{i-1}, t^*_i)$. By the averaging argument, there exists an i such that $f(T_{l-1} \cup T^*_{i-1}, t^*_i) \geq (m^*(L) - m(l-1))/L$. We can then obtain $m_l = f(T_{l-1}, t_l) \geq f(T_{l-1}, t^*_i) \geq (m^*(L) - m(l-1))/L$, where the first inequality comes from how we choose t_l, and the second comes from submodularity.

We can then prove $m(l) \geq (1 - (1 - 1/L)^l)m^*(L), \forall l \in [1, L]$ by induction. When $l = 1$, the result holds: $m(l) = m_1 \geq m^*(L)/L = (1 - (1 - 1/L)^l)m^*(L)$ from (7). Suppose the inequality holds for l, i.e., $m(l) \geq (1 - (1 - 1/L)^l)m^*(L)$, we have

$$\begin{aligned} m(l+1) &= m(l) + m_{l+1} \geq m(l) + (m^*(L) - m(l))/L \\ &= m^*(L)/L + m(l)(1 - 1/L) \\ &\geq m^*(L)/L + m^*(L)(1 - (1 - 1/L)^l)(1 - 1/L) \\ &= (1 - (1 - 1/L)^{l+1})m^*(L). \end{aligned}$$

Let $l = L$ in the above inequality, we have $m(L) \geq (1 - (1 - 1/L)^L)m^*(L) \geq (1 - 1/e)m^*(L)$, which completes the proof. □

4.2 Approximation Algorithm for Jointly Optimizing the Utility and Efficiency

As mentioned before, there are alternative objectives for the optimization problem. We now discuss how can we jointly optimize the utility and efficiency in (4). As we show below, this is also an NP-hard problem.

Proposition 3. *Given a generalized context* $\hat{c}$*, it is NP-hard to find a set of tasks* T^**, such that:*

$$T^* = \underset{T\subseteq\mathcal{T}:|T|=L}{\mathrm{argmax}} \sum_{c:c\rightarrow\hat{c}} \Pr[c|\hat{c}] \cdot \alpha \cdot \max_{t\in T} \rho_t \cdot \mathit{CAR}(t|c) - \lambda \cdot L. \tag{8}$$

Proof. In Proposition 1, we have proved the NP-hardness of finding L tasks that maximizes the expected revenue defined in (2). This is actually a special case of the new optimization problem (8). If one can solve the new optimization problem in polynomial time, and the resulting list of tasks is of size L_0, then he can solve the optimization problem defined in Proposition 1 with $L = L_0$ in polynomial time, which contradicts Proposition 1. Hence, the new optimization problem is also NP-hard. □

Below, we describe Algorithm 2 that approximately solves the above optimization problem (8) in polynomial time and give the analysis of approximation ratio in Proposition 4. In Algorithm 2, L_{max} denotes the maximum number of recommended tasks chosen by the worker beforehand.

Algorithm 2 Greedy Algorithm for Jointly Utility and Efficiency Optimization

```
Input: $\mathcal{T}$, $\hat{c}$, $\lambda$, $L_{\text{max}}$
Output: $T$
    // initialization
 1: $L \leftarrow 1, \theta \leftarrow 0, T \leftarrow \emptyset$;
 2: while $L \leq L_{\text{max}}$ do
 3:     $Q \leftarrow \emptyset$;
 4:     repeat
 5:         $t \leftarrow \mathrm{argmax}_{t\in\mathcal{T}}\, \mathbb{E}[R(Q \cup t|\hat{c})] - \mathbb{E}[R(Q|\hat{c})]$;
 6:         $Q \leftarrow Q \cup \{t\}$;
 7:     until $|Q| = L$
 8:     if $\theta \leq \mathbb{E}[R(Q|\hat{c})] - \lambda \cdot L$ then
 9:         $\theta \leftarrow \mathbb{E}[R(Q|\hat{c})] - \lambda \cdot L$;
10:         $T \leftarrow Q$;
11:     end if
12:     $L \leftarrow L + 1$;
13: end while
14: return $T$
```

Proposition 4. *The greedy algorithm approximates the optimal solution within a factor of* $1 - 1/e$.

Proof. Following the notations in the proof of Proposition 2, let $m(L)$ and $m^*(L)$ denote the objective function value for the greedy solution at a fixed L and the objective function value for the optimal solution at a fixed L, respectively. Denote by m_G and m_G^* the objective function value over all L for the greedy solution obtained by Algorithm 2 and the objective function value over all L for the optimal solution, respectively. Our aim is to prove $m_G \geq m_G^* \cdot (1 - 1/e)$.

Suppose that the optimal objective function value m_G^* is reached when $L = \tilde{L}$, we have $m^*(\tilde{L}) = m_G^*$. Now, from (6), $m(\tilde{L}) \geq m^*(\tilde{L}) \cdot (1-1/e) = m_G^* \cdot (1-1/e)$. Since $m_G \geq m(L)$, $\forall L = 1, \ldots, L_{\max}$, we have $m_G \geq m_{\tilde{L}} \geq m_G^* \cdot (1-1/e)$, which completes the proof. □

5 Privacy-Preserving Statistics Collection

In the previous sections, we have assumed that the server has prior information about worker statistics in the task selection component such as $\Pr[c|\hat{c}]$ and $\text{CAR}(t|c)$. In this section, we describe how to obtain these statistics while achieving design goals described in Section 2.2.

5.1 Problem Overview

There are three parties in the offline statistics collection component: the MC server, workers, and a semi-honest third party (proxy). The server makes statistics queries and collects the results. Workers locally store their historical contexts as well as performance records, and answer queries. The proxy plays a mediation role between the server and the workers in order to protect worker privacy. The idea of using a semi-honest proxy to ensure distributed differential privacy has been used previously in different applications [21], [27], [28]. Here, we use it for privacy-preserving statistics collection in our framework.

5.1.1 Threat Model

The server is assumed to be potentially malicious in the sense that it intends to violate worker privacy. The server may attempt to use the statistics collection protocol to learn private information about workers, or deploy its own workers and manipulate their answers. Moreover, the server may also publish its collected worker statistics.

Workers are also assumed to be potentially malicious in the sense that they may distort the final statistics learned by the server by submitting false or illegitimate answers.

The proxy is assumed to be semi-honest or "honest-but-curious", which means it will faithfully follow the specified protocol, but may attempt to exploit additional information learned in executing the protocol. The proxy does not collude with other parties.

In practice, as suggested in [27], the server may pay the proxy to execute the statistics collection protocol. Such a proxy has been used in previous papers [29], [30] and the relationship between the proxy and the MC server pre-exists in industry today which usually does not lead to collusion. For example, pharmaceutical companies pay an independent organization who evaluates the safety, quality, or performance of their products and may give unfavorable results against the pharmaceutical companies. Therefore, we believe that it is reasonable to have such a semi-honest proxy in our protocol. Note that some distributed differential privacy designs [18]–[20] do not require such a proxy. However, they have a high computation or communication cost, which is impractical in most applications. With the semi-honest proxy, we can provide differential privacy guarantees in a distributed setting with a much higher efficiency.

5.1.2 Assumptions

We assume that workers have correct public keys for the server and the proxy, that the server and the proxy have correct public keys for each other, and that all the corresponding private keys are securely kept. We also assume secure, reliable, and authenticated communication channels among the server, the proxy, and workers.

Workers are assumed to be dynamic, which means that they may quit in the middle of the statistics collection process due to unstable wireless connection or power saving. Moreover, the computation and communication resources of worker devices are assumed to be limited.

5.1.3 Privacy Definition

Our framework allows workers to choose whether to participate in statistics collection, and protect the privacy of participating workers. We consider the privacy risk for participating workers from two aspects. We first guarantee that no other party, except the worker himself, would know his private information during statistics collection. This can be achieved through data encryption as shown in [31]–[33].

Moreover, we also consider privacy leakage that can not be solved by data encryption. A potential privacy leakage is due to multiple runs of statistics collection when a worker does not participate in all runs, e.g., because he has reached home. Hence, we should protect every worker from an adversary (with arbitrary background knowledge) who tries to trace or de-anonymize a user between several runs of statistics collection protocol. To this end, we adopt the privacy notion of (ϵ, δ)-differential privacy [18], which ensures that the result of our protocol does not significant change with the presence or absence of a single worker. The formal definition of (ϵ, δ)-differential privacy is given as follows [18]:

Definition 3. A statistics collection algorithm $\mathcal{F}$ satisfies (ϵ, δ)-differential privacy if, for all datasets D_1 and D_2 differing on one record, and for all outputs $O \subseteq \text{range}(\mathcal{F})$, the following inequality holds:

$$\Pr[\mathcal{F}(D_1) \in O] \leq \exp(\epsilon) \times \Pr[\mathcal{F}(D_2) \in O] + \delta. \quad (9)$$

In other words, if the statistics are differentially private, the adversary would not change his probabilistic belief about an individual even with knowledge of the published statistics and any other auxiliary information.

The strong guarantees provided by differential privacy is not free. Privacy is enforced by adding noise to the outcome of the algorithm. There are two privacy parameters ϵ and δ. The first parameter ϵ bounds the ratio of output distributions on input of D_1 and D_2, with higher ϵ representing weaker privacy guarantees. The latter parameter δ relaxes the relative shift at events that are unlikely to happen.

5.2 Computation of Worker Statistics Based on Counting

Based on a differentially private counting procedure, the statistics collection protocol gathers responses from workers and

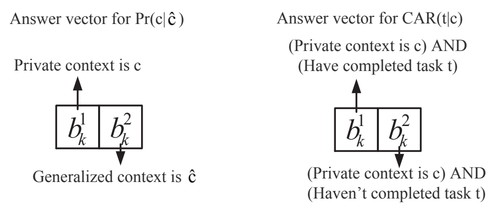

Answer vector for Pr(c|ĉ)
Private context is c
b_k^1 b_k^2
Generalized context is ĉ
Answer vector for CAR(t|c)
(Private context is c) AND (Have completed task t)
b_k^1 b_k^2
(Private context is c) AND (Haven't completed task t)

Fig. 4. Illustration of the answer vector for worker k.

transforms the responses into statistics $\Pr[c|\hat{c}]$ and CAR$(t|c)$. We first describe how these statistics can be computed over contexts with a counting procedure. We will give the details of the counting procedure in Section 5.3.

Calculating $\Pr[c|\hat{c}]$. The statistic $\Pr[c|\hat{c}]$ is calculated as the number of workers with context c divided by the number of workers with generalized context $\hat{c}$. Hence, the MC server should count the numbers of workers with context c and generalized context $\hat{c}$, respectively. To this end, the MC server constructs a statistics query which asks two questions: (1) "Is your private context c?" and (2) "Is your generalized context $\hat{c}$?". Both questions expect binary answers "yes" (represented by 1) or "no" (represented by 0). The answer from each worker k is a vector (b_k^1, b_k^2) that consists of two bits, each corresponding to a question. An example of the answer vector is shown in Fig. 4. Therefore, given a privacy-preserving counting procedure, we can aggregate answers to these two questions from workers, and calculate $\Pr[c|\hat{c}]$ in a privacy-preserving manner.

Calculating CAR$(t|c)$. The statistic CAR$(t|c)$, as defined in Section 3.1, is calculated as the total number of workers with context c who have completed task t divided by the total number of workers with context c. The MC server also generates a query that consists of two questions: (1) "Is your context c?" and (2) "If your context is c, have you successfully completed task t?". The answer to these two questions is contained in a two-bit vector as well. If the context of the worker is not c, the answer of the worker would be $[0, 0]$; if the context of the worker is c, and he has successfully completed the task, his answer would be $[1, 1]$; if the context of the worker is c but he does not complete the task, his answer would be $[0, 1]$. Here, the first bit of the answer vector indicates that the worker satisfies both context c and completion of task t, while the second bit indicates whether the worker's context is c as shown in Fig. 4. Similarly, we can compute CAR$(t|c)$ using a counting procedure over answers to these questions from workers.

In practice, the MC server first sets M and ϵ, where M indicates the number of workers that need to be queried and ϵ is the privacy budget that controls the amount of noise. The queries and the parameters M and ϵ are then broadcasted to workers, whose answers will be added bit by bit as shown in Fig. 5 by a privacy-preserving counting procedure explained in the next section.

5.3 Distributed Differentially-Private Counting Procedure

We now describe our differentially private counting procedure which is the key part of the statistical collection protocol as explained before. The counting procedure takes answers from workers as the input data, and outputs a noisy sum with differential privacy guarantees, i.e., a sum that does not significant change with the presence or absence of a single worker. It is trivial to achieve this privacy goal in a centralized setting, where the data owner fully possesses the data and can easily add noise to the sum before revealing it to others. However, in our distributed setting where the data are owned by workers themselves, it is non-trivial to add the noise to the distributed data. There are a few works which provide differential privacy in a distributed setting [18]–[20]. However, they either have a high computation cost on each user [18] or requires users to be online during the whole computation process [19], [20], rendering them impractical for a large-scale setting as our scenario.

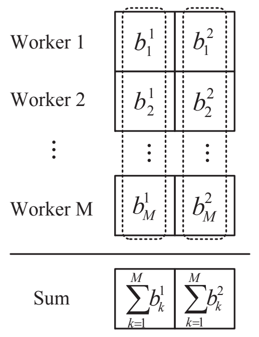
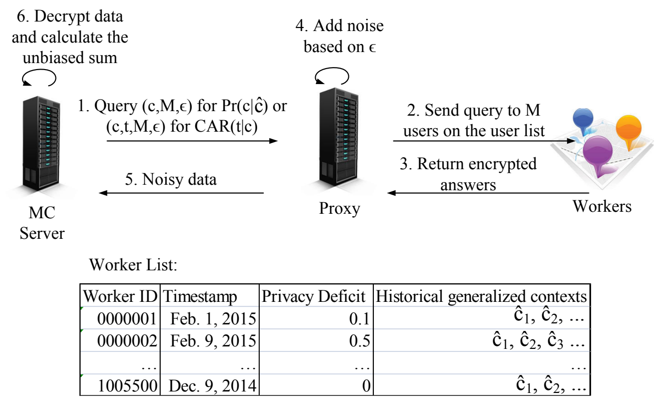


Fig. 5. Aggregation process of answer vectors from M workers.


6. Decrypt data and calculate the unbiased sum
1. Query (c,M,ϵ) for Pr(c|ĉ) or (c,t,M,ϵ) for CAR(t|c)
5. Noisy data
MC Server
4. Add noise based on ϵ
Proxy
2. Send query to M users on the user list
3. Return encrypted answers
Workers
Worker List:

Worker ID	Timestamp	Privacy Deficit	Historical generalized contexts
0000001	Feb. 1, 2015	0.1	ĉ$_1$, ĉ$_2$, ...
0000002	Feb. 9, 2015	0.5	ĉ$_1$, ĉ$_2$, ĉ$_3$ ...
...	...	...	...
1005500	Dec. 9, 2014	0	ĉ$_1$, ĉ$_2$, ...

Fig. 6. Schematics of our privacy-preserving counting process.

To ensure the scalability of our protocol, we employ a semi-honest proxy to achieve differential privacy under distributed setting. The proxy will aggregate answers from workers and add noise to the sum; however, it is unable to learn the value of answers or their sum. Moreover, since the final statistics are open to public, the proxy cannot learn the added noise. Otherwise, it may subtract the noise from the noisy sum and obtain the accurate count. This requires that the proxy can only "blindly" add the differentially private noise.

As depicted in Fig. 6, the counting protocol works as follows:

Step 1: Depending on the type of statistics it wants to calculate, the server formulates a query request and specifies the number of queried workers M and the privacy parameter ϵ for this query.

Step 2: The proxy has access to a worker list which records the historical general contexts $\hat{c}$ of workers, their current privacy loss, and the timestamps when they last connected to the proxy. For a query with a specific context c, the proxy sends it to M workers from the worker list whose historical general contexts may cover context c and whose privacy loss has not exceed a predefined threshold. After workers return their answers, the proxy will add ϵ and δ to the privacy loss of queried workers in the worker list.

Step 3: After a worker receives the query, he constructs an answer vector as described in Section 5.2. The worker needs to prevent the proxy from learning his answer, thus he encrypts his answer with the public key of the MC server and sends the ciphertext to the proxy. For instance, if the answer is $(0, 1)$, the encrypted answer would be $e(0)||e(1)$, where $e(\cdot)$ is the encryption operation under the public key of the MC server. For ease of presentation, we call the encrypted binary bit $e(\cdot)$ as a *coin*. We will explain the used cryptosystem at this step in Section 5.4.

Step 4: The proxy aggregates the coins into buckets with each bucket corresponding to a bit in the answer vector. The proxy also adds binomial noise to each bucket based on the privacy budget ϵ, where the noise consists of coins with random binary values to be specified in Section 5.4. For example, if the amount of noise is N, then N coins are added to each bucket.

Step 5: The proxy forwards the buckets to the MC server.

Step 6: After the MC server receives the buckets for a certain query, it first decrypts all the coins in each bucket with its private key and sums up the decrypted values in each bucket. The final results are calculated as $\sum_k b_k^1 - N/2$ and $\sum_k b_k^2 - N/2$, where $N/2$ is used to cancel the added noise. Since the MC server cannot tell who constructs the coins, the identities of workers are anonymized.

Note that when workers communicate with the proxy, standard cryptographic protocols such as TLS are used. Therefore, only the proxy can learn the content of messages sent from workers. Similarly, standard cryptographic protocols are used during the communication between the proxy and the MC server.

5.4 Coin Generation and Noise Addition

In this section, we give the details about coin generation, that is, encryption of binary data. We use the Goldwasser-Micali (GM) cryptosystem [34] due to its high efficiency in encrypting binary values and its XOR-homomorphic property.

For completeness, we briefly describe the GM encryption process as follows.

- *Key Generation*: Let $N \equiv p \cdot q$ where p and q are two large prime numbers independent of each other. Choose a non-residue x such that the Legendre symbol $x/p = x/q = -1$, and hence the Jacobi symbol $x/N = +1$. The public key is (N, x), and the private key is (p, q).
- *Encryption*: Let b be the message we want to encrypt. Choose a non-zero random number $r \in \mathbb{Z}_N^*$. The ciphertext e is given by
$$e \equiv r^2 \cdot x^b \bmod N. \tag{10}$$
- *Decryption*: Given the ciphertext e, the receiver uses the prime factorization (p, q) to check whether e is a quadratic residue. In order to do this, the receiver first computes $e_p = e \bmod p$ and $e_q = e \bmod q$. If both $e_p^{(p-1)/2} = 1 \bmod p$ and $e_q^{(q-1)/2} = 1 \bmod q$ hold, then e is called a quadratic residue. The receiver sets $b = 0$ if e is a quadratic residue, and sets $b = 1$ otherwise.

5.4.1 Differentially Private Noise

The amount of noise required to achieve (ϵ, δ)-differential privacy is calculated in [18] and described as follows.

Proposition 5. *Let N be the number of unbiased coins added in a bucket, i.e., the amount of Binomial noise. The statistics collection algorithm achieves (ϵ, δ)-differential privacy if*

$$N \geq \frac{64 \ln(\frac{2}{\delta})}{\epsilon^2}. \tag{11}$$

The parameters δ and M is selected by the server. Suppose that any query of each person is sensitive, then $\delta > 1/M$ indicates the disclosure of at least one person's privacy. Therefore, δ is selected to be smaller than $1/M$. With this constraint, the amount of noise added into each bucket should satisfy

$$N \geq \frac{64 \ln(2M)}{\epsilon^2}. \tag{12}$$

5.4.2 Blind Noise Generation

Note that the proxy should collaborate with workers to generate unbiased and blind coins. If the proxy generates the unbiased coins by itself, it would know the accurate value of the noise. When the server publishes the obfuscated statistics later, the proxy could recover the accurate statistics. On the other hand, if the workers are trusted to generate the noise coins, they may intentionally distort the final statistics by generating biased noise coins.

To address this issue, following a flipping approach proposed in [27], we let the workers generate N coins first, which are flipped by the semi-honest proxy. This is made possible with the XOR-homomorphic property of GM encryption, where the result of XOR operation on plaintexts can be obtained by decrypting the product of the ciphertexts. Note that for any $b, b' \in \{0, 1\}$, we have

$$e(b) \cdot e(b') = e\left(b \oplus b' \bmod N\right), \tag{13}$$

where $e(\cdot)$ is the encryption operator. With this homomorphic property, two parties can collaboratively generate an encrypted value of either 0 or 1 while no single party can know or control the final results. As long as one of the two parties is unbiased, the final results would be unbiased. Hence multiplying a coin generated by workers and an unbiased coin generated by the proxy always results in a flipped coin that is both unbiased and hidden from the proxy. In this way, we can generate a pool of unbiased coins for noise addition.

5.5 Analysis of Design Goals

We have listed three design goals in Section 2.2: privacy, scalability, and robustness. In the following, we analyze how can we achieve these goals in the proposed protocol.

Privacy. Our protocol ensures differential privacy for all workers. Whenever a worker participates in the statistics collection procedure, he reveals some information about himself. Such kind of privacy loss is quantified by the privacy budget [18], [35]. The privacy loss is accumulated across queries until it surpasses the worker's privacy budget. Then the worker stops contributing any data in the statistics collection procedure. This provides the best privacy for the worker. However, it influences the lifetime of our protocol because after all existing workers reach their privacy budgets, the statistics can only be learned from new workers. Note that in our framework, the database is adaptive due to the following reasons. First, worker context may change over time. Second, the set of workers that answer the same query would changes. If a worker stops contributing his data, the influence of his data to

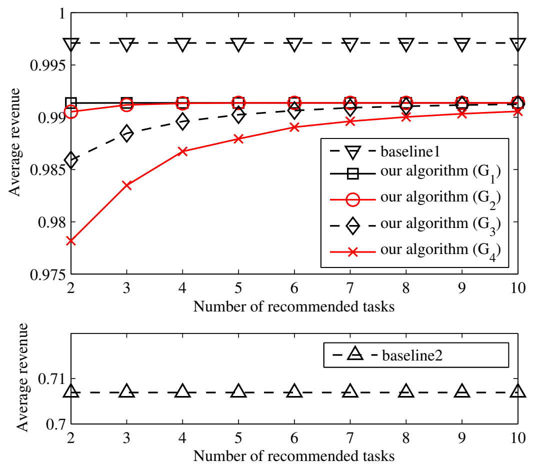


Fig. 7. Effect of generalization level in Problem (3).

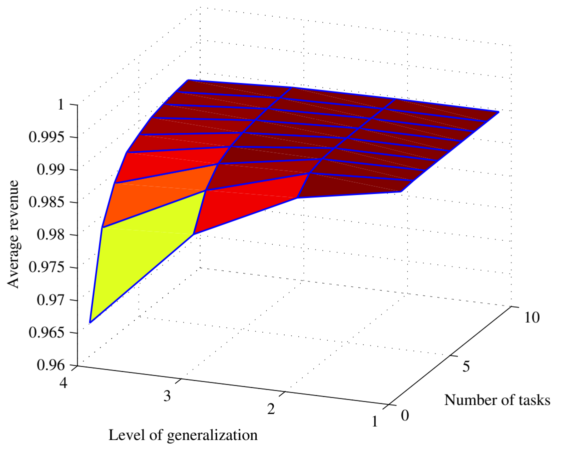


Fig. 8. Trade-offs among privacy (generalization level), utility (average revenue), and efficiency (number of tasks) in Problem (3).

the final result will decrease over time. With such observations, we can treat the privacy loss as the worst-case measurement. In practice, the proxy maintains a worker list which records the latest login time of each worker. If a worker does not contribute data for a long time, his privacy loss can be set back to 0.

Scalability. To achieve the scalability goal, we should first ensure low per-worker computation cost so that even when the number of workers is large, the cost for individual worker does not change much. In our protocol, the cost per worker is $\mathcal{O}(1)$. When the number of workers is large, it is hard to ensure that all workers are online during the process. Hence we should ensure that the statistics can be computed even when some workers leave in the middle of the computation process. In our protocol, workers only need to submit answers once and no further communication is required after that. Therefore, our protocol allows workers to leave after they submit their answers.

Robustness. With the GM encryption, we are able to bound the error brought by malicious workers due to the following reasons. First, the data encrypted by GM encryption is guaranteed to be either 0 or 1. This can by done by checking the Jacobi symbols of ciphertexts at the proxy (Jacobi symbol of legitimate ciphertexts is "$+1$") or checking the decrypted value at the MC server. Second, each worker could only add a single coin to each bucket. Therefore, a malicious worker would be unable to distort the final sum by more than 1. If an adversary tries to substantially distort the final sum, it should employ a large number of malicious workers, which is difficult in practice. Suppose 1% of workers are malicious, the error introduced by malicious worker would be less than 1%.

6 Performance Evaluation

To evaluate the performance of the proposed optimization algorithms, we generate a synthetic dataset to simulate the statistics $\Pr(c)$ and CAR$(t|c)$. Without loss of generality, we assume the frequency of worker contexts is uniformly distributed. The data set includes 2048 exact contexts and 10000 different tasks. The detailed contexts can be generalized at four different levels. There are 512 level-1 generalized contexts denoted as "G$_1$", 128 level-2 generalized contexts denoted as "G$_2$", 8 level-3 generalized contexts denoted as "G$_3$", and 2 level-4 generalized contexts denoted as "G$_4$". The statistic CAR$(t|c)$ is generated in a way

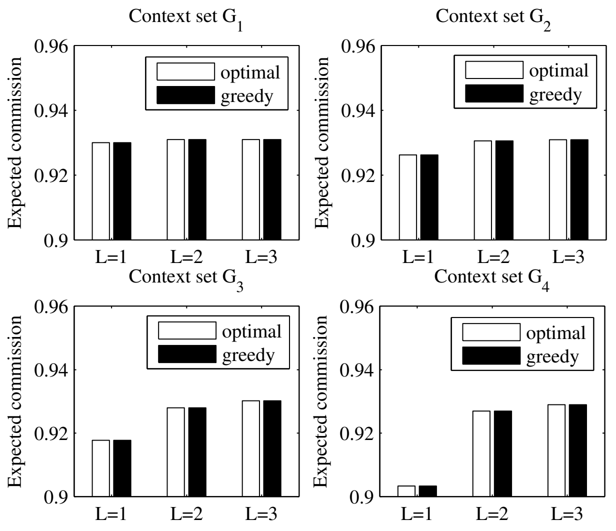


Fig. 9. Performance of our approximation Algorithm 1.

such that the closer two exact contexts are, the more similar the distributions of CAR$(t|c)$ would be. The CAR$(t|c)$ of a task t for workers with the same exact context c follows a uniform distribution. The payments of tasks are set as a random value between 0 and 10, and the ratio of commission α is chosen to be 0.1.

Firstly, we test the effectiveness of the task recommendation model. To this end, we compare our proposed algorithm (Algorithm 1) with two baseline algorithms, "baseline1" and "baseline2". The first baseline algorithm uses the exact worker context as the input. With the exact worker context, the algorithm directly chooses the task that maximizes the revenue gained by the MC platform. worker privacy is compromised in this algorithm to trade for utility and efficiency. On the contrary, in the second baseline algorithm, no context information is used, and therefore worker privacy is maximized. This algorithm does not consider the difference of worker contexts and recommends tasks that have highest payments.

Fig. 7 shows the expected revenue of the MC platform by

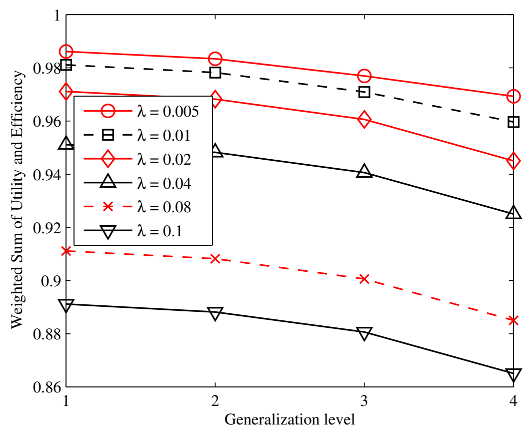


Fig. 10. Effect of the generalization level and the weight for efficiency in Problem (4).

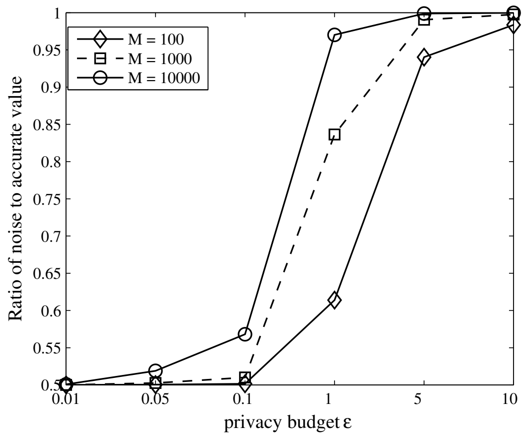


Fig. 11. Effect of number of workers M and privacy budget ϵ on the accuracy of statistics.

adjusting the size of the recommended task set L. We run the experiments using six different algorithms, including two baseline algorithms and Algorithm 1 with four different levels of generalized contexts. Intuitively, the two baseline algorithms serve as a upper bound and a lower bound of other algorithms, respectively, which is clearly shown in the figure. The expected revenue of Algorithm 1 increases when more context information is used. For a specific level of generalization, the expected revenue increases with L. For example, when the generalization level is 3 (which corresponds to "G_3" in the figure), the revenue increases from 0.986 to 0.991 as L increases from 2 to 10. Note that the performances of the two baseline algorithms do not change with L because they always select the task that maximizes the expected revenue regardless of L.

Fig. 8 illustrates the trade-offs among utility, efficiency and privacy. We can observe that the deficiency in privacy can be compensated by increasing the size of the recommendation set L. This aligns with the fact that with a larger L, a worker receives more tasks and is more likely to get a suitable task. Moreover, when L is large, there are already enough tasks in the recommendation set. Thus, the marginal improvement in revenue becomes smaller, which means that increasing L would not improve the utility much after L approaches a reasonably large number. We can also see the impact of varying privacy levels on the expected revenue. It is expected that as the requirement for privacy increases, the difference of revenue obtained by using the exact context and generalized context would increase, which can be observed from the figure. Since the differences are always within a small range, we conclude that the expected revenue of our privacy-preserving approach is close to that of the privacy-oblivious one.

Secondly, we evaluate the performance of the proposed approximation algorithms. Due to the NP-hardness of the original optimization problem, the optimal solution becomes intractable in practice when either L or the task space is large. Therefore, we use a reduced size of data set for this experiment (i.e., 100 tasks and $L = 1, 2, 3$). Fig. 9 compares the performances of Algorithm 1 and the optimal algorithm. We see that there is little difference between the two algorithms for $L = 1, 2, 3$ and $|\mathcal{T}| = 100$. The difference between the two algorithms may grow as L becomes larger, but we have proved in previous sections that our approximation algorithm has an approximation ratio of $1 - 1/e$.

Thirdly, we show the performance of Algorithm 2, which jointly optimizes utility and efficiency. Fig. 10 plots the weighted sum of utility and efficiency with the weight coefficient λ ranging from 0.005 to 0.1. For each λ, the x-axis represents the level of context generalization, and the y-axis represents the the weighted sum of utility and efficiency. Same as what we get from Fig. 7, the weighted sum decreases as the level of generalization increases, which shows a clear trade-off between utility and privacy. With the increase of λ, the optimized weighted sum decreases. This is reasonable because it is shown in (4) that for the same list of recommended tasks, the weighted sum decreases with the increase of λ. As a result, the optimal weighted sum is expected to decrease as well.

Lastly, we analyze the privacy and accuracy of the statistics collection component. Since noise needs to be added to provide (ϵ, δ)-differential privacy, the privacy is achieved at the cost of accuracy. We illustrate the trade-off between the privacy parameter ϵ and the accuracy of the statistics in Fig. 11. We can see from the figure that, with the increase of the total number of queried workers M, higher accuracy is obtained. To achieve an acceptable accuracy of, for example, 0.8, the value of ϵ should be within the range of $(0.1, 1)$ for most M. We need to keep such a trade-off between accuracy and privacy in mind when we choose the privacy parameter ϵ.

7 System Overhead

In this section, we analyze the system overhead of the proposed framework, including both task selection and statistics collection components.

We list the estimated running time for the task selection component in Table 1. Since the time for the optimal algorithm grows exponentially as L grows, we only run this algorithm at $L = 2$ or 3 with a small dataset where the number of tasks is 100. We can see that the time to get an optimal solution grows rapidly with L, while the time for the proposed greedy algorithm is linear with respect to L. Fig. 12 further illustrates the running time for Algorithm 1. Obviously, the running time depends on both the size of recommendation set L and the level of context generalization. Each line in the figure represents a level of context generalization. We can clearly see that for a fixed level of context generalization, the running time increases almost linearly with

TABLE 1
Running time (in unit of seconds) of recommending L tasks from a set of 100 tasks ($L = 2$ or 3)

L	G_1		G_2		G_3		G_4	
	Optimal	Greedy	Optimal	Greedy	Optimal	Greedy	Optimal	Greedy
2	13.0	8.1	13.3	2.0	14.0	0.1	24.9	0.04
3	1125.2	15.7	1267.4	3.8	1275.6	0.3	1838.1	0.1

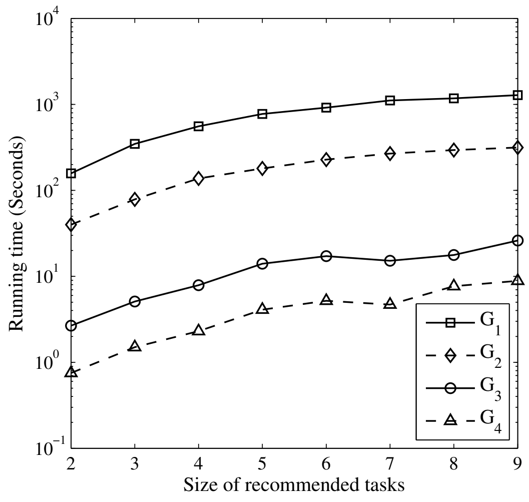


Fig. 12. Running time of recommending tasks for Algorithm 1.

respect to L, which is important for practical systems. In fact, we can also infer from the figure that the running time also increases linearly with respect to the size of generalized contexts (the sizes of generalized contexts for "G_1", "G_2", "G_3", and "G_4" are 512, 128, 8, and 2, respectively).

In the following, we analyze the computation, storage and communication overhead for statistics collection. Firstly, we analyze the computation overhead for the GM cryptosystem. With a 1024-bit key length, a smartphone running Android 2.2 with 1GHz processor can execute more than 800 encryptions within one second [27]. Since workers only need to execute the encryption process once for each query request, the computation cost is negligible for them. The proxy is implemented with Apache Tomcat $6.0.33$, which can execute more than $15,000$ GM encryptions, or $123,000$ homomorphic XORs per second, and the server is implemented with Java source code, which can execute more than 6000 GM decryptions per second. Consider a normal setting where there are 5000 workers with 100 different exact contexts which generalize to the same generalized context, and there are 90 tasks relevant to this generalized context. Suppose 10% of the workers participate in the statistics collection process, the proxy needs to execute 18 encryptions and 18 homomorphic XORs for a single statistic query when the privacy parameter ϵ is set to 5 according to (12). In order to calculate all the statistics needed for the task selection model, the proxy needs to execute 18×27200 encryptions and 18×27200 homomorphic XORs, which takes 31 seconds and 4 seconds, respectively. For the same setting, the server needs to decrypt a total of $(500 + 18) \times 27200$ coins, which takes 36 minutes. Note that the statistics can be calculated offline and are reusable among workers with similar contexts. By contrast, if the protocol is implemented with the Paillier system, in order to calculate statistics for a task selection model, it takes the mobile worker, the proxy, and the server 4 seconds, 139 minutes, and 4500 minutes, respectively. Therefore, the GM crytosystem use in our framework is highly efficient.

Next, we discuss the storage and communication bandwidth requirements. Since a worker transmits no more than 3 coins for each statistics collection query and a periodically generated coin for noise addition, the storage requirement for him is in the order of kB. Considering that workers can selectly respond to the requests, the storage overhead is quite acceptable. Suppose the coins should be sent out within one second, the bandwidth requirement would be around 1 kB/s. As for the proxy, since it needs to store all queried coins and noise coins before sending them to the server, which is about 518×27200 coins in total in the above setting, the storage overhead would be about 1.7 GB. Since the statistics collection process are computed beforehand, we assume the maximum transmission time is 30 minutes. Therefore, the bandwidth for sending these data is 1 MB/s. Note that although the storage requirement for computing a statistic is not small, in practice, the statistic only needs to be computed once and updated at a low frequency after it has been calculated. The overheads for the proxy to update the statistics are at the same order of the overhead for workers.

8 Related Work

In this section, we review some works related to our problem in the literature.

Privacy issues in mobile applications. Previous works on privacy issues of mobile applications mainly focus on location privacy in location-based services, and they use either obfuscation to hide true locations [36], [37] or aggregation to hide individual sensitive information [38]. However, none of them discuss how to recommend tasks in the absence of accurate private information. In this paper, we consider the fundamental trade-offs among privacy, utility, and efficiency, and provides a flexible framework to select tasks at different trade-off points.

Privacy issues in statistics computation. There are multiple approaches addressing privacy issues in statistics computation such as k-anonymity and ℓ-diversity. k-anonymity [39] ensures that each individual is indistinguishable from at least $k - 1$ other individuals, and thus cannot be uniquely identified. ℓ-diversity [40] requires that there are at least ℓ well-represented values for every sensitive attribute. However, these two privacy notions can only guarantee syntactic properties of the released data, and cannot protect against an adversary with certain background knowledge. On the other hand, differential privacy [41] makes no assumption of the adversary and is a very strong guarantee. However, the traditional notation of differential privacy is designed for centralized databases and cannot be directly used in a distributed setting. There are a few approaches which have been proposed to provide differential privacy over distributed data [17]–[20], but they are impractical for a large-scale setting. For example, the computation cost per user in [18] is $\mathcal{O}(M)$ where M is the number of users, which becomes prohibitive for a large population of users. The computation cost is reduced to $\mathcal{O}(1)$ in [19], [20], but they use an expensive secret sharing protocol that is not scalable to a large group of workers. Additionally, the solution in [21] uses two servers to collaboratively calculate statistics and handle user dynamics. However, a single malicious user could greatly distort the querying result in this approach. Our approach provides a practical solution to compute differentially-private statistics over distributed data that are both scalable and robust against malicious workers.

Task assignment in crowdsourcing. There are a few works in task recommendation for web-based crowdsourcing applications. Ho and Vaughan [42] address the scenario where heterogeneous tasks are assigned to workers with unknown skill sets with an exploration-exploitation trade-off. Yuen et al. [43] utilize performance history and task search history to model user preference and recommend tasks for a user based on his/her preference. Ambati et al. [44] implicitly model user skills and interests, and recommend tasks based on user preference. However, these works have not addressed the specific privacy concerns in MC scenarios where tasks should be recommended to workers based on private, sensitive information.

9 Conclusion

We have considered the privacy issues in task recommendation for mobile crowdsourcing. We have proposed a task recommendation framework which recommends mobile crowdsourcing tasks without violating worker privacy. The proposed framework is comprised of two components: task selection component and statistics collection component. In the task selection component, we have developed a privacy-aware optimization model of task selection that considers the intrinsic trade-offs among utility, privacy and efficiency and selects tasks based on the limited information of worker context. workers have the choice of how much private information they are willing to share with the server. In the statistics collection component, we have proposed a protocol that gathers necessary statistics about worker contexts while guaranteeing differential privacy. We have evaluated the effectiveness and efficiency of the proposed framework and analyzed the system overhead. For future work, we intend to incorporate other popular task recommendation algorithms such as collaborative filtering. We also plan to jointly consider task assignment and task recommendation problems in MC systems.

Acknowledgements

This work was partially supported by the U.S. National Science Foundation under Grants ECCS-1129062, CNS-1343356, and CNS-1423165.

References

[1] Statista. [Online]. Available: http://www.statista.com/

[2] Waze. [Online]. Available: https://www.waze.com

[3] Uber. [Online]. Available: https://www.uber.com/

[4] Stereopublic. [Online]. Available: http://www.stereopublic.net/

[5] OpenSignal. [Online]. Available: http://opensignal.com/

[6] C. Freifeld, R. Chunara, S. Mekaru, E. Chan, T. Kass-Hout, J. Brownstein *et al.*, "Participatory epidemiology: use of mobile phones for community-based health reporting," *PLoS medicine*, vol. 7, no. 12, pp. e1 000 376–e1 000 376, 2009.

[7] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "A large-scale study of mobile crowdsourcing with smartphones for urban sensing applications," in *Proc. of ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13), Zurich, Switzerland*, 2013.

[8] A. Tamilin, I. Carreras, E. Ssebaggala, A. Opira, and N. Conci, "Context-aware mobile crowdsourcing," in *Proc. of UbiMi Workshop*. ACM, 2012, pp. 717–720.

[9] Y. Wang, Y. Huang, and C. Louis, "Respecting user privacy in mobile crowdsourcing," *SCIENCE*, vol. 2, no. 2, pp. pp–50, 2013.

[10] X. Shen, B. Tan, and C. Zhai, "Implicit user modeling for personalized search," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 824–831.

[11] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: query processing for location services without compromising privacy," in *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 763–774.

[12] S. Guha, A. Reznichenko, K. Tang, H. Haddadi, and P. Francis, "Serving ads from localhost for performance, privacy, and profit," in *HotNets*, 2009.

[13] M. Fredrikson and B. Livshits, "Repriv: Re-imagining content personalization and in-browser privacy," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 131–146.

[14] S. Chakraborty, K. R. Raghavan, M. P. Johnson, and M. B. Srivastava, "A framework for context-aware privacy of sensor data on mobile systems," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*. ACM, 2013, p. 11.

[15] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, M. K. Reiter *et al.*, "Playing devil's advocate: Inferring sensitive information from anonymized network traces." in *NDSS*, vol. 7, 2007, pp. 35–47.

[16] B. F. Ribeiro, W. Chen, G. Miklau, and D. F. Towsley, "Analyzing privacy in enterprise packet trace anonymization." in *NDSS*, 2008.

[17] G. Acs and C. Castelluccia, "I have a dream!(differentially private smart metering)," in *Information Hiding*. Springer, 2011, pp. 118–132.

[18] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology-EUROCRYPT 2006*. Springer, 2006, pp. 486–503.

[19] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 735–746.

[20] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *NDSS*, vol. 2, no. 3, 2011, p. 4.

[21] M. Hardt and S. Nath, "Privacy-aware personalization for mobile advertising," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 662–673.

[22] Y. Xu, K. Wang, B. Zhang, and Z. Chen, "Privacy-enhancing personalized web search," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 591–600.

[23] C. S. Jensen, H. Lu, and M. L. Yiu, "Location privacy techniques in client-server architectures," in *Privacy in location-based applications*. Springer, 2009, pp. 31–58.

[24] E. Baralis, L. Cagliero, T. Cerquitelli, P. Garza, and M. Marchetti, "Context-aware user and service profiling by means of generalized association rules," in *Proceedings of the 13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Part II*. Springer-Verlag, 2009, pp. 50–57.

[25] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *Proceedings of the 11th ACM conference on Electronic commerce*. ACM, 2010, pp. 209–218.

[26] D. S. Hochbaum and A. Pathria, "Analysis of the greedy approach in problems of maximum k-coverage," *Naval Research Logistics*, vol. 45, no. 6, pp. 615–627, 1998.

[27] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke, "Towards statistical queries over distributed private user data," in *Proceedings of the 9th Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[28] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," in *40th International Conference on Very Large Data Bases*, 2014, pp. 919–930.

[29] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*. Ieee, 2010, pp. 1–9.

[30] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*. Ieee, 2010, pp. 1–9.

[31] J. Shi, Y. Zhang, and Y. Liu, “Prisense: privacy-preserving data aggregation in people-centric urban sensing systems,” in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[32] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher, “Poolview: stream privacy for grassroots participatory sensing,” in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 281–294.

[33] F. D. Garcia and B. Jacobs, “Privacy-friendly energy-metering via homomorphic encryption,” in *Security and Trust Management*. Springer, 2011, pp. 226–238.

[34] S. Goldwasser and S. Micali, “Probabilistic encryption & how to play mental poker keeping secret all partial information,” in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM, 1982, pp. 365–377.

[35] C. Dwork, “A firm foundation for private data analysis,” *Communications of the ACM*, vol. 54, no. 1, pp. 86–95, 2011.

[36] M. Duckham and L. Kulik, “A formal model of obfuscation and negotiation for location privacy,” in *Pervasive computing*. Springer, 2005, pp. 152–170.

[37] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, “Protecting location privacy: optimal strategy against localization attacks,” in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 617–627.

[38] J. W. Brown, O. Ohrimenko, and R. Tamassia, “Haze: Privacy-preserving real-time traffic statistics,” in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 530–533.

[39] L. Sweeney, “k-anonymity: A model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[40] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[41] C. Dwork, “Differential privacy,” in *Automata, languages and programming*. Springer, 2006, pp. 1–12.

[42] C.-J. Ho and J. W. Vaughan, “Online task assignment in crowdsourcing markets,” in *AAAI Conference on Artificial Intelligence*, 2012.

[43] M.-C. Yuen, I. King, and K.-S. Leung, “Task recommendation in crowdsourcing systems,” in *Proceedings of the First International Workshop on Crowdsourcing and Data Mining*. ACM, 2012, pp. 22–26.

[44] V. Ambati, S. Vogel, and J. Carbonell, “Towards task recommendation in micro-task markets,” in *Proceedings of The 25th AAAI Workshop in Human Computation*, 2011.