Optimizing Task Location Privacy in Mobile Crowdsensing Systems

Xuewen Dong, *Member, IEEE*, Wen Zhang, Yushu Zhang, *Member, IEEE*, Zhichao You, Sheng Gao, Yulong Shen, *Member, IEEE*, and Chao Wang

***Abstract*—The location information for tasks may expose sensitive information, which impedes the practical use of mobile crowdsensing in the industrial Internet. In this article, to our knowledge, we are the first to discuss the privacy protection of task locations and propose a codebook-based task allocation mechanism to protect it. Considering the cost of system utility caused by privacy protection technology, the tradeoff between local privacy and system utility is formalized a multiobjective optimization problem. The optimal solution is theoretically derived, and the optimal task allocation scheme is obtained. In addition, the selected allocation codebook (SAC) method is introduced to solve the problem of high computational resource consumption in the task allocation process and protect the task location privacy to some extent. The experimental results show that the SAC method sacrifices system utility but improves the privacy protection for task locations by 60% on average.**

***Index Terms*—Information theory, location privacy protection, mobile crowdsensing (MCS), privacy exposure measure, task allocation.**

Manuscript received May 22, 2021; revised August 6, 2021; accepted August 22, 2021. Date of publication September 3, 2021; date of current version December 27, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61972310, Grant 61941114, Grant 61972017, and Grant 62072487, in part by the Key R&D Program of Shaanxi Province under Grant 2019ZDLGY12-03 and Grant 2019ZDLGY13-06, in part by the Basic Research Program of Jiangsu Province under Grant BK20201290, and in part by the National Statistical Science Foundation of China under Grant 2020LD01. Paper no. TII-21-2146. *(Corresponding authors: Yulong Shen; Sheng Gao.)*

Xuewen Dong, Wen Zhang, Zhichao You, and Yulong Shen are with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China, and also with the Shaanxi Key Laboratory of Network and System Security, Xi'an 710071, China (e-mail: xwdong@xidian.edu.cn; wzhang_21@stu.xidian.edu.cn; zcyou@stu.xidian.edu.cn; ylshen@mail.xidian.edu.cn).

Yushu Zhang is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210016, China (e-mail: yushu@nuaa.edu.cn).

Sheng Gao is with the School of Information, Central University of Finance and Economics, Beijing 100081, China (e-mail: sgao@cufe.edu.cn).

Chao Wang is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Shaanxi Key Laboratory of Network and System Security, Xi'an 710071, China (e-mail: wangchao@xidian.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3109437.

Digital Object Identifier 10.1109/TII.2021.3109437

I. Introduction

With the rapid development of Internet technology, mobile devices are becoming increasingly popular and equipped with powerful embedded sensors (e.g., compasses, global positioning systems, thermometers, microphones, and cameras) [1]. These mobile devices with wireless sensing abilities can be used to monitor a wide variety of human activities and environments, thus creating a new Internet of Things (IoT) sensing paradigm called mobile crowdsensing (MCS). Due to their low cost and comprehensive coverage, MCS systems have become common data collection tools for various industrial IoT applications and services. In terms of data collection, MCS relies on the contributions of mobile devices from a large number of participants or groups of people. Compared with traditional sensor networks, MCS networks utilize existing sensing and mobile communication infrastructures to provide unprecedented coverage of time and space. Due to the powerful perception and communication abilities of MCS systems, such systems have been widely developed and used in various applications, including traffic management [2], road surface condition monitoring [3], [4], daily lifestyle monitoring in the elderly population [5], and air pollution detection [6]. The typical system architecture of MCSs includes three parts: service platforms, requesters, and data providers (i.e., workers) [7]–[9]. The process of MCS is shown in Fig. 1, which involves the basic functions of data perception, data acquisition, and information service provision in a distributed and independent service mode.

Privacy protection has always been a popular issue in MCS systems, and has been widely studied by researchers. In a general MCS system, tasks are location-related, and the sensing task location information should be reported to the platform to facilitate the matching of tasks and workers according to location in task allocation. However, reporting location information during the task allocation process can lead to serious privacy issues, especially if there are dishonest service platforms and malicious attackers in the system. If participants upload the correct data unaware, attackers can take advantage of it. Attackers can pose as workers, or they can act as more powerful attackers directly in the task allocation platform and then lead to privacy data leakage. Therefore, location privacy protection in MCS systems is important. The existing works mainly focused on privacy protection for workers and proposed privacy protection schemes from different perspectives to prevent security threats in MCS systems. However, they may neglect to take another

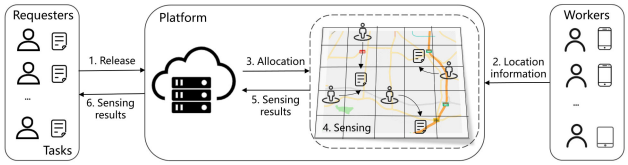


Fig. 1. MCS system.

critical factor, task location privacy, into consideration. There has been little work related to protecting the privacy of sensing task locations. With the exposure of task location information, dishonest servers, and malicious attackers can learn requesters' habits and obtain sensitive information. Therefore, it is necessary to provide privacy protection for sensing service requesters and combat potential security threats.

Existing works have proposed many methods to protect privacy, and the most intuitive way to measure the effectiveness of these methods is the degree of privacy protection. Therefore, how to measure the degree of privacy protection or exposure is a significant problem that must be considered. In information theory, mutual information can be regarded as the amount of information contained in a random variable about another random variable, which can be used to calculate the degree of privacy exposure. Additionally, effectively allocating tasks while protecting task location privacy has always been a challenge. In general, the realization of privacy protection in MCS systems usually comes at the cost of system utility. However, privacy protection and system utility are challenging to balance, and simple privacy protection mechanisms are challenging to implement.

Our work focuses on and discusses the challenges of MCS noted above. In this article, service requesters maintain privacy by obscuring information reported to a third-party platform. However, a typical cybersecurity approach alone is insufficient, as the normal assignment operations of the platform may allow an adversary to infer critical aspects of a sensing task. For example, an attacker could legitimately manipulate multiple smartphones and use the task allocation results from the platform to infer the characteristics of tasks and requesters. Moreover, the accuracy of the communication data transmitted among users and platforms affects the protection of user privacy and the utility of system allocation. Thus, there is a potential tradeoff between protecting task location privacy and maximizing the utility of the MCS system.

A. Our Contributions

Considering all of the above problems, this article proposes a scheme to protect the privacy of requesters' task locations and designs a general MCS system that meets the relevant requirements for sensing task allocation while protecting location privacy. To our knowledge, we are the first to discuss the task location protection problem in MCS systems. The main contributions of this article are as follows.

1) We construct two sets of discrete values, namely, a set of the information obtained by an attacker and a set of the actual task possible location information. By using these two sets, we successfully apply the concept of mutual information in information theory to an MCS system, measure the degree of privacy exposure of the system, and achieve privacy quantification.
2) We solve the sensing task allocation problem in the MCS system while protecting task location privacy. In the process of task allocation, an obfuscation mechanism is used to protect the privacy of task locations and avoid privacy exposure to some extent.
3) We model the MCS system discussed above as a sensing task allocation problem under privacy protection constraints. This problem is, in essence, a multiobjective optimization problem. The optimal solution of the optimization problem is derived theoretically, which is used to calculate the optimal sensing task allocation result in further.
4) The selected allocation codebook (SAC) method is proposed to solve the problem of high computational resource consumption during the selection of optimal task allocation results. Notably, the allocation result is chosen from within a small range. The SAC method can aid in task privacy protection to some extent and achieves a tradeoff between location privacy protection and utility in the studied MCS system.

B. Related Works

There are few studies on the protection of the privacy of task locations, so this section focuses on the privacy protection of participants in the MCS system. Many existing works have focused on protecting the location privacy of participants in MCS systems (e.g., [10]–[18]). For instance, Wang *et al.* [10] used the k-anonymity method to reduce the risk of location privacy leakage. To protect the locations of users, Wang *et al.*[11] proposed a task assignment framework for location privacy protection with a geographic ambiguity processing function. In [12], a task allocation framework for personalized privacy protection in MCS systems was proposed, and the protection scheme was implemented according to the personal privacy levels of workers. Zhang *et al.* [13], using homomorphic Pailler encryption technology, designed two privacy protection schemes according to whether a user is stable. Wang *et al.* [14] designed a differential privacy protection framework to mitigate the loss of information quality caused by location confusion. Zhang *et al.* [15] proposed a novel differentially private geocoding mechanism to preserve workers' location privacy, and then workers can use obfuscated geocode to describe their locations. Xiong *et al.* [16] combined machine learning with game theory and proposed an artificial intelligence-enabled three-party game framework for guaranteed data privacy in the mobile edge crowdsensing of the IoT. Xiong *et al.* [17] proposed a personalized privacy protection framework based on game theory and data encryption to tackle the issue of low-quality crowdsensing services in MCSs. Zou *et al.* [18] proposed an effective blockchain-based location-privacy-preserving crowdsensing model. These works all contribute to protecting privacy; however, all of them ignore the protection of task location privacy. Tasks' location

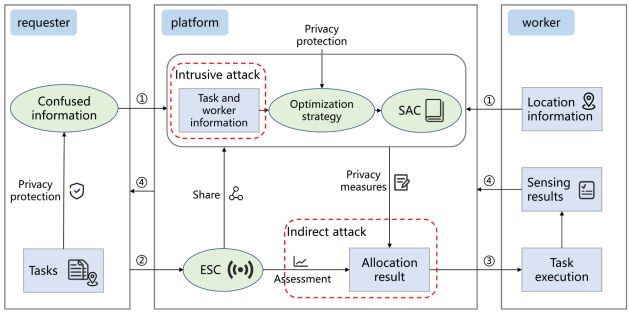


Fig. 2. System model.

information has the risk of exposing the privacy of the requesters, which deserves our attention.

There have also been some works on the measurement of privacy exposure. However, many works use the gap between the obtained data and the actual data to measure the degree of privacy protection and pay little attention to the amount of information in the dataset. Andres *et al.* [19] introduced the concepts of location-based privacy and geo-indistinguishability, which encompass the intuitive concept of protecting a user's location within a certain radius. Other privacy quantification methods, such as the average error and conditional entropy methods, have also been introduced [20]. Zhang *et al.* [21] considered the tracking level in individual user location privacy protection, used the concept of mutual information to quantify the tracking level of location privacy leakage and studied the privacy-utility tradeoff. In the process of dynamic spectrum allocation, the privacy of primary users was protected in [22]. Notably, mutual information and the average distance error were used to quantify the degree of privacy leakage for primary user locations. These works proposed various privacy measurement methods and are applied to help quantify the effectiveness of privacy protection methods. Therefore, mutual information can also be applied to the MCS system to measure the effectiveness of the protection method of task location privacy.

II. System Model

This section first discusses the execution process of the traditional MCS system and then builds the system model to protect the privacy of the task location, as shown in Fig. 2. The model consists of three main parts, which are defined and set up in detail in this section. It should be noted that the green parts of the figure are associated with privacy protection.

A. Traditional MCS System

As shown in Fig. 2, the general execution process of an MCS system without privacy protection (i.e., blue parts of the system) is as follows. First, the workers who are recruited to perform sensing tasks and the requesters who publish sensing tasks to obtain sensing services send requested tasks and location information to the platform, as noted in step 1. The platform acts as a third party in a cloud server to allocate sensing tasks to workers, as shown in step 3. The chosen workers perform the allocated tasks and send the collected sensing data to the platform. Then, the platform feeds the data to the corresponding requesters in step 4. The complete execution process of the traditional MCS system without considering privacy protection can be found in Fig. 1. In the MCS system, fairness is a key indicator used to evaluate the results of task allocation, that is, whether all tasks have been completed and whether the number of workers completing each task is approximately equal.

B. Private Requesters

The set of requesters in an MCS system is denoted as $\mathcal{R} = \{r_1, r_2, \ldots, r_{|\mathcal{R}|}\}$. Each requester has one task that needs to be completed. Similar to [23] and [24], this article considers homogeneous tasks, that is, they belong to the same class, differ little in execution, and consume the same amount of perceptual and computational resources. For example, in an environmental noise sensing system, a worker with a mobile device can simply open specific software to detect it, and they consume roughly the same amount of computing resources. Similarly, workers are assumed to be homogenous, and there is no difference in their ability to perform tasks; moreover, they can perform any task. For any task, there is no limitation on the number of workers performing it. It is assumed that the quality of the sensing service is positively related to the number of workers performing the corresponding sensing task. However, if too many workers are involved in completing a task, the impact on fairness will be negative. We define the quality of the collected sensing data as n_{r_j} $(r_j \in \mathcal{R})$, which is related to the number of workers performing the task r_j. When calculating the quality of a task allocation result, both the quality of task completion and the fairness of tasks are taken into consideration.

The set of tasks published by these requesters is $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{R}|}\}$, and the corresponding position set is expressed as $\mathcal{L}_t = \{l_{t_1}, l_{t_2}, \ldots, l_{t_{|\mathcal{R}|}}\}$. Because a platform may not be trustworthy, requesters do not want to send real task location information to the platform. To protect the above private information, confusion operations are performed before location information is sent to the platform. We assume that the "confused" location set sent to the platform is $\hat{\mathcal{L}}_t = \{\hat{l_{t_1}}, \hat{l_{t_2}}, \ldots, \hat{l_{t_k}}\}$, where $k \geq |\mathcal{R}|$; this set includes confused locations and real locations. In other words, in the map, there are $|\mathcal{R}|$ real task locations and $k - |\mathcal{R}|$ confused locations for tasks; therefore, it is difficult for an attacker to determine, which task locations are real, which is the basic objective of privacy protection.

C. Dynamic Workers

Suppose the set of workers recruited to perform sensing tasks is denoted as $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$. These workers may be smartphone users equipped with sensors that can perform sensing tasks; they post their location information $\mathcal{L}_w = \{l_{w_1}, l_{w_2}, \ldots, l_{w_{|\mathcal{W}|}}\}$ to the platform, and the platform uses this information to allocate tasks.

A worker w_i $(w_i \in \mathcal{W})$ may be assigned one task or no task at all, and if no task is assigned, the worker stays at her current location l_{w_i} and does nothing. In performing assigned tasks, the loss of worker resources is divided into two aspects: path loss and perceptual calculation loss. We assume that the distance from worker w_i to task t_j is $dist(l_{w_i}, l_{t_j})$, where l_{w_i} is the location of

worker w_i and l_{t_j} is the location of task t_j to be performed by worker w_i. Additionally, the workers assigned to tasks inevitably consume resources in the process of performing a task; this consumption of resources is called the cost c_i of worker $w_i \in \mathcal{W}$. Here, $c_i = dist(l_{w_i}, l_{t_j}) + \alpha \cdot com_i$, where com_i represents the computational resources consumed by worker w_i to perform a task. Because the resources consumed to complete any sensing task are assumed to be the same, com_i is a constant. Additionally, com_i has little effect on the cost calculation, so we focus on the distance cost of a worker performing a task; that is, the weighting coefficient α is very small. Of course, α can be changed according to different requirements of the system. For example, if the perceived computing loss is a large part of the cost calculation, α can be 3 or 5 or some number greater than 1.

D. Service Platform

As shown in Fig. 2, the environmental sensing capability (ESC) module differs from that in the typical system architecture for MCS [7] and can detect real task locations on the map generated in step 2. There may be errors in the detection results of the ESC module due to the existence of obfuscation locations. The ESC module detects the location information $\tilde{\mathcal{L}}_t^{\text{esc}}$ in the map; here, $\tilde{\mathcal{L}}_t^{\text{esc}} \in \hat{\mathcal{L}}_t$. Then, the location distribution of tasks is estimated based on the detected information $\tilde{\mathcal{L}}_t^{\text{esc}}$ and historical allocation records, denoted as $\mathcal{L}_t^{\text{esc}} = \{l_1, l_2, l_3, \cdots\}$. The ESC module analyzes and shares these data with the platform to assist in assigning tasks by evaluating the task allocation results based on $\mathcal{L}_t^{\text{esc}}$ without knowledge of the real location set $\mathcal{L}_t$.

Acting as an information collector and task distributor, the platform plays an important role in the overall process. In each period, first, the platform receives location information $\hat{\mathcal{L}}_t$, $\mathcal{L}_w$, and $\mathcal{L}_t^{\text{esc}}$ from requesters, workers, and the ESC module, respectively. Second, the platform allocates tasks to workers, and the result of assignment is denoted as P_t^w. Simultaneously, a low distance cost, high quality, and high fairness should be achieved in task allocation. The real location distribution of a task is $\bar{l}$, and the result of task assignment is $\bar{p}$. Based on these variables, the platform can calculate the total cost of the corresponding task allocation result: $C_{\text{sum}}(\bar{l}, \bar{p}) = \sum_{i=1}^{|\mathcal{W}|} c_i$, and the corresponding quality calculation is as follows: $Q_{\text{sum}}(\bar{p}) = \sum_{i=1}^{|\mathcal{W}|} a_i + \sum_{r_j \in \mathcal{R}} n_{r_j}$. If worker $w_i \in \mathcal{W}$ is allocated a task and goes to the corresponding location l, where the ESC module has detected the task, $a_i = 1$; i.e., $l \in \mathcal{L}_t^{\text{esc}}$. In all other cases, $a_i = 0$.

III. Privacy Preliminaries

In this section, the adversary model used by the studied MCS system is defined, and we describe how to quantify privacy exposure. This information is used to analyze the privacy problem and assess the effectiveness of the privacy protection method proposed in the next section.

A. Adversary Model

We assume that there may be malicious adversaries in the system, who can hack into the components of the system, as shown in the red parts in Fig. 2. We define the information observed by the attacker as $\mathcal{A}$. In this article, we consider the following two types of attack patterns.

1) Indirect Attack: Observation of allocation results. After obtaining the task assigned from the platform, a dishonest worker may leakage this information to an attacker. In addition, an attacker can hack into a worker's host or service for a relatively long time, and steal a series of task location information. Or an attacker can manipulate his/her device to be a legitimate worker and obtain task allocation results assigned to him/her. The collected information is analyzed comprehensively to infer the real location information for tasks.

2) Intrusive Attack: Direct observation of the platform. Compared to an indirect attack, the attacker who makes a direct attack is more powerful, and she can invade the platform where the task is assigned, potentially eavesdropping on communications with requesters and the ESC module. Although the location information sent by requesters to the platform is confused, an attacker can perform a comprehensive analysis of the confused locations and ESC measurements and implement an inference attack based on the real task location information.

B. Privacy Metrics

The existing works on privacy protection mostly use self-information to measure the degree of privacy disclosure (e.g., [19], [20]). The measurement of privacy by self-information is usually defined as the gap between the obtained data and the actual data but does not pay attention to the amount of information, which is more suitable for measuring the privacy exposure of data encryption or fuzzy radius mechanisms. Thus, this article turns attention to mutual information to measure the degree of task location privacy exposure since the obfuscation mechanism of adding fake data to a real dataset is adopted. To better use mutual information in the MCS system, we first learn the basic definition of mutual information [25]. Mutual information in information theory is related to the correlation of information associated with two random variables X and Y. Mutual information is defined as follows:

$$I(X,Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x,y) log \frac{P(x,y)}{P(x)P(y)}. \tag{1}$$

Mutual information reflects the amount of information contained in a random variable that is related to another variable. The mutual information value for two unrelated random variables is zero.

The location information consists of discrete values, and the mutual information calculation introduced above also aims at discrete variables. Therefore, the obfuscation mechanism is adopted in this article to protect the privacy of task location and ensure that the protected data are still discrete. In this article, we measure the correlation between the information A obtained by an attacker and the actual task location information L. Specifically, based on the task possible location information published by the requesters, which includes actual locations and confused locations, a set of possible location combinations L for real tasks is genereted. For an indirect attack, the information obtained by the attacker is the result of task assignment. Workers

are combined with different locations to create a set of possible assignment results, and A is a subset of this set. For an intrusive attack, the information obtained by the attacker is the distribution of task locations, thus A is a subset of L. Then, the two sets of discrete values are used to calculate the mutual information. Here, L and A are discrete random variable sets and $\mathcal{L}$ and $\mathcal{A}$ are the corresponding value spaces. Therefore, privacy exposure (i.e., mutual information) can be expressed by the following formula:

$$I(L,A) = \sum_{x\in\mathcal{L}}\sum_{y\in\mathcal{A}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \tag{2}$$

$$= \sum_{x\in\mathcal{L}}\sum_{y\in\mathcal{A}} P(y\mid x)P(x)\log\frac{P(y\mid x)}{P(y)}. \tag{3}$$

A small mutual information value reflects a low level of privacy leakage or excellent privacy protection; in contrast, larger mutual information values indicate considerable privacy leakage issues.

IV. Optimization of Privacy

Considering the privacy of task locations and system utility, a multiobjective optimization problem is proposed. The mutual information value in privacy quantification is usually relative rather than absolute. It is difficult to define the range of mutual information values that corresponds to acceptable privacy exposure. In contrast, the scope of system utility is well defined, and defining thresholds is relatively simple and intuitive. Therefore, in the optimization problem in this article, the goal is to minimize privacy leakage, and system utility is used as a constraint condition for the optimization goal.

To optimize task location privacy, we use mutual information as the privacy metric of the system for the following reasons. First, mutual information represents an information theory-based measure of the correlation between two random variables, and a low mutual information value represents low privacy leakage. Second, the mutual information related to the actual task locations and the observed attacker information are independent of the adversary models adopted by the system. In other words, mutual information is useful in privacy quantification based on various adversary models. Thus, it is feasible to perform privacy quantification with mutual information.

A. Optimization Goal

Unlike previous works on privacy protection, this article focuses on protecting the privacy of task locations. First, task locations are protected by an obfuscation mechanism, and then the degree of privacy exposure is quantified by mutual information. However, the privacy protection mechanism comes at the cost of some system utility. To achieve a tradeoff between system utility and task location privacy, the optimization objective is expressed as follows:

$$\min I(L,A) = \sum_{x\in\mathcal{L}}\sum_{y\in\mathcal{A}} P(x,y)log\frac{P(x,y)}{P(x)P(y)} \tag{4}$$

subject to

$$\sum_{y\in\mathcal{A}} P(y)Q_{\text{sum}}(y) \geq Q_w, \forall x\in\mathcal{L} \tag{5}$$

$$\sum_{y\in\mathcal{A}} P(y\mid x)C_{\text{sum}}(x,y) \leq C_w, \forall x\in\mathcal{L} \tag{6}$$

$$P(y\mid x)\geq 0, \forall x\in\mathcal{L}, \forall y\in\mathcal{A} \tag{7}$$

$$\sum_{y\in\mathcal{A}} P(y\mid x) = 1, \forall x\in\mathcal{L}. \tag{8}$$

Equation (4) shows the optimization goal of the problem: minimizing task location privacy exposure. Equation (5) is a constraint on the quality and fairness of task completion, where Q_w is the lower bound of the constraint. Equation (6) is a constraint on the resources consumed by workers after task assignment, and C_w is the upper bound of the constraint. Equations (5) and (6) are combined to constrain the utility of the system. Equations (7) and (8) ensure that the probability distribution is valid. These constraints are linear, and the mutual information is convex for conditional probabilities, so this optimization problem is a convex optimization problem.

B. Optimal Solution

For general optimization problems with equality and inequality constraints, Karush–Kuhn–Tucker (KKT) conditions are necessary to obtain the optimal solution. For the inequality-constrained optimization problem in this article, the KKT conditions of the optimal solution are as follows:

$$P(x)\log\left(\frac{P^*(y\mid x)}{P(y)}\right) + C_{\text{sum}}(x,y)\mu_x^* - \mu_{x,y}^* - \mu_w^* \times P(x)Q_{\text{sum}}(y) + \lambda_x^* = 0 \tag{9}$$

$$\mu_w^*(Q_w - \sum_{y\in\mathcal{Y}} P^*(y)Q_{\text{sum}}(y)) = 0 \tag{10}$$

$$\mu_x^*(-C_w + \sum_{y\in\mathcal{Y}} C_{\text{sum}}(x,y)P^*(y\mid x)) = 0, \forall x\in\mathcal{L} \tag{11}$$

$$\mu_{x,y}^* P^*(y\mid x) = 0, \forall x\in\mathcal{L}, \forall y\in\mathcal{A} \tag{12}$$

$$\mu_w^*\geq 0, \mu_x^*\geq 0, \mu_{x,y}^*\geq 0 \tag{13}$$

where * represents the optimal solution; μ_w^* and μ_x^* are the multipliers of the task completion quality and worker cost constraints in the system utility context, respectively; and $\mu_{x,y}^*$ and λ_x^* are constraint multipliers that guarantee the validity of conditional probability distributions.

Lemma 1: $P^*(y\mid x)$ be any optimal solution to the optimization problem described in the previous section. If the utility constraint is not active, i.e., if

$$\sum_{y'\in\mathcal{L}} P^*(y')Q_{\text{sum}}(y') > Q_w$$

then

$$P^*(y\mid x) = P^*(y), \forall x\in\mathcal{L}, \forall y\in\mathcal{A}$$

i.e., the adversary observations produced by the optimal strategy based on obfuscation are independent of the actual task location state.

Algorithm 1: Codebook Construction.

Input: SAC size: s, the number of codebook optimization steps: t, task state space: z, allocation results space: $\mathcal{V}$

Output: codebook: $\mathcal{C}b$

// Phase 1: Random selection
for $i = 0 : s-1$ **do**
  Randomly generate $v_i \in \mathcal{V}$ and put v_i in $\mathcal{C}b$
  for $j = 0 : z-1$ **do**
    Compute $w_i += Q_{sum}(v_i) - C_{sum}(j, v_i)$
// Phase 2: Codebook optimization
$\mathcal{C}_{alt} \leftarrow \emptyset$;
for $i = 0 : t-1$ **do**
  Randomly generate $v \in \mathcal{V}$
  for $j = 0 : z-1$ **do**
    Compute $w += Q_{sum}(v) - C_{sum}(j, v)$
  Put v in $\mathcal{C}_{alt}$
  Select one with the largest w (w_{max}) of $\mathcal{C}_{alt}$, v_{max}
  Select one with the smallest w (w_{min}) of $\mathcal{C}b$, v_{min}
  if $w_{max} \geq w_{min}$ **then**
    Take v_{min} out of $\mathcal{C}b$ and put it in $\mathcal{C}_{alt}$
    Take v_{max} out of $\mathcal{C}_{alt}$ and put it in $\mathcal{C}b$
Return codebook $\mathcal{C}b$

Proof: The proof is established by contradiction. Suppose $P^*(y \mid x) \neq P^*(y)$ for some x and y. Specifically, since $P^*(y) = \sum_{x \in \mathcal{L}} P^*(y \mid x) P^*(x)$, there is an x and y such that

$$P^*(y \mid x) > P^*(y) \tag{14}$$

or an x' such that $P^*(y \mid x') < P^*(y)$. Furthermore, given (8), then (14) implies there is some y' such that $P^*(y' \mid x) < P^*(y')$. Let y_0 be a report state that does not compromise privacy. First, if for some x' we have

$$P^*(y_0 \mid x') < P^*(y_0) \tag{15}$$

for some small $\epsilon > 0$, we can construct a new reporting strategy P' that satisfies $P'(y_0 \mid x') = P^*(y_0 \mid x') + \epsilon$ and $P'(y \mid x') = P^*(y \mid x') - \epsilon$ where have (14), and where $P' = P^*$ otherwise. Since y_0 has no effect on the system and constraint (5) is not tight, P' satisfies the constraints of the optimization problem and strictly reduces the mutual information by construction. This also holds for the case of $P^*(y_0 \mid x') > P^*(y_0)$ since this implies (15) for some x'.

The second and only other possible case is that $P^*(y_0 \mid x') = P^*(y_0), \forall x \in \mathcal{L}$. In this case, (14) holds for some $y \neq y_0$. We construct a P' starting with P^* and set $P'(y_0 \mid x) = P^*(y_0 \mid x) + \epsilon$. We then iteratively set $P'(y \mid x) = P^*(y \mid x) - \epsilon$ for $\forall x \in \mathcal{L}$ and some y that satisfies (14). Similar to the first case, this construction yields a valid P' that satisfies the constraints of the optimization problem and strictly reduces the mutual information by construction. Since in both cases, we can construct a solution that strictly reduces the mutual information, the statement that P^* is an optimal solution that satisfies $P^*(y \mid x) \neq P^*(y)$ for some x and y must be a contradiction.

Theorem 1: The optimal solution to the optimization problem in this article is

$$P^*(y \mid x) = (P^*(y)\exp(\mu_w^* Q_{\text{sum}}(y) - C_{\text{sum}}(x, y)\mu_x^*/P(x)))/ \left(\sum_{y' \in \mathcal{A}} P^*(y')\exp(\mu_w^* Q_{\text{sum}}(y') - C_{\text{sum}}(x, y')\mu_x^*/P(x))\right). \tag{16}$$

Proof: After analyzing the above KKT conditions, (9) can be rewritten as

$$P^*(y \mid x) = P^*(y)\exp((\mu_{x,y}^* + \mu_w^* P(x) Q_{\text{sum}}(y) - C_{\text{sum}}(x, y)\mu_x^* - \lambda_x^*)/P(x)). \tag{17}$$

By applying (8) to (17), we obtain

$$\sum_{y' \in \mathcal{A}} P_(^* y')\exp((\mu_{x,y}^* + \mu_w^* P(x) Q_{\text{sum}}(y) - C_{\text{sum}}(x, y)\mu_x^* - \lambda_x^*)/P(x)) = 1 \tag{18}$$

which can be written as

$$\exp(\lambda_x^*/P(x)) = \sum_{y' \in \mathcal{A}} P^*(y')\exp(\mu_w^* Q_{\text{sum}}(y) + (\mu_{x,y}^* - C_{\text{sum}}(x, y)\mu_x^*)/P(x)). \tag{19}$$

By substituting (19) into (17), we obtain the result in (16).

In a real situation, the vast number of possibilities makes it impossible to list all potential results for assigned tasks. However, we can use theoretically optimal strategies to inform the development of comparatively practical strategies.

V. Selected Allocation Codebook

A. Selected Allocation Codebook

When the multiobjective optimization formula described above is solved, a computational method for selecting task assignment results satisfying the constraints can be obtained, which requires calculating all possible combinations of task assignment results. Considering the large number of workers and many potential allocation results, in the process of task allocation, it is impossible for the platform to calculate the cost and utility of each allocation result and then select the optimal result. Notably, the corresponding computational burden would be enormous. Moreover, as the number of tasks and the number of workers increase, the quantity of possible task allocation results exponentially increases. The practical privacy protection approach we propose is to sample the task allocation result space and construct a codebook that can be used to allocate tasks; this codebook is called the SAC. It is worth noting that the codebook only contains partial task distribution results, so the SAC method can protect privacy to some extent against indirect attacks. At the same time, even though the size of the codebook is limited, the SAC method designed to select task assignment results from the codebook does not significantly degrade the quality of the final result.

Codebooks are generated for a given number of tasks and workers. Based on the locations and numbers of tasks and

Algorithm 2: SAC Method.

Input: SAC size: s
Output: codeword: c_i

1 **for** *True task location state* x *at each time slot* **do**
2 **for** $i = 0 : s-1$ **do**
3 Compute $weight_i = exp(\mu_w Q_{sum}(y_i) - C_{sum}(x, y_i)\mu_x))/(\sum_{j=0}^{s-1} exp(\mu_w Q_{sum}(y_j) - C_{sum}(x, y_j)\mu_x))$
4 Select the codeword c_i, where $i \in \{0, \cdots, s-1\}$ with the largest weight $weight_{max}$
5 return c_i

workers, the allocation of tasks can be performed. A codebook contains multiple codewords, each of which is one result of task allocation. The size of a codebook can be changed without limitation, so we can use the codebook size as a design parameter in the following method. The main objective of the SAC method is to calculate the cost consumption and system utility of each codeword in the codebook, determine the corresponding weights, and select an appropriate codeword according to the weight results.

B. Construction and Use of Codebooks

Algorithm 1 describes the process of codebook construction. In the usual method, codebook construction is a process of randomly selecting a certain number of codewords from a selectable range. However, a major drawback of this random approach is that the results can be either good or bad, depending on sheer luck. Therefore, a new method is proposed to select codewords for codebooks, and the concept of the number of codebook optimization steps is introduced. The number of codebook optimization steps refers to the replacement and selection of codewords through many rounds of random selection to improve the constructed codebook. First, we generate a codebook $\mathcal{C}b$ by randomly selecting some codewords from the optional space (lines 2–3). Second, the weights of these codewords are calculated separately using the following formula (lines 4–5): $w = \sum_{j=0}^{z}(Q_{\text{sum}}(v) - C_{\text{sum}}(j, v))$, where z is the task state space and v is the current codeword used to calculate the corresponding weight. Finally, within the number of codebook optimization steps, random codewords are generated and added to the alternative set $\mathcal{C}_{\text{alt}}$ (lines 8–12). The optimal codeword in the alternative set is selected to replace the worst codeword in the codebook (lines 13–17). It is worth noting that the eliminated codewords in each round will not be directly discarded but stored in the alternative set. A codeword is randomly generated in each round and added to the alternative set, and then the optimal codeword is selected to add to the codebook. If the eliminated codeword in each round is directly discarded, a new codeword is randomly generated in the next round. Because the codewords are all randomly selected, this approach can be considered a random method.

Algorithm 2 describes the SAC method in pseudocode. Once the codebook is created, the platform uses it to allocate tasks to workers. The codebook is much smaller than the total task allocation result space, so it is relatively easy for the platform to calculate the weight of each codeword in the codebook. In a real situation, the weight of each codeword in the codebook is calculated (lines 1–3), and the codeword c_i with the largest weight is selected as the result of the current round (lines 4–5).

C. Complexity Analysis

There are two parameters related to codebook construction, namely, the codebook size $\mathcal{C}$ and the number of codebook optimization steps $\mathcal{S}$. The complexity of codebook construction is $\mathcal{O}(\mathcal{S})$, that is, the size of the codebook optimization steps. After the codebook construction is complete, only the task assignment result in the codebook needs to be selected without considering anything outside the codebook. Therefore, the complexity of the SAC method is $\mathcal{O}(\mathcal{C})$, which is the size of the codebook.

D. Privacy Analysis

This article attempts to confuse opponents with inaccurate or imprecise positions. Task requesters add fake entries when uploading their task information, and actual and fake messages are mixed together to make it difficult to distinguish. When the ESC module is exploring, there is no guarantee that the data obtained are entirely correct, so the information it detects is also challenging to distinguish between true and false. Therefore, the information obtained by the platform may be true or false. Whether pretending to be a worker or attacking the platform data center, the attacker does not directly obtain the actual task location information. The following content analyzes the lower limit of mutual information values under the two cases, according to the activity of tasks in the system.

Theorem 2: When the number of tasks is fixed and known, if the occurrence of each task location combination and each task assignment result have an equal probability, the amount of information in this case is 0.

Proof: Assume that there are N_t task location information instances uploaded to the platform, and the number of workers is N_w. Meanwhile, suppose that the number of active tasks is M, where $M \leq N_t$. Therefore, it can be calculated that there are $n1 = \frac{N_t!}{M!}$ possible combinations of correct tasks, while for M active tasks (considering the situation where the worker chooses not to perform the task), there are $m1 = (N_t + 1)^{N_w}$ possible results assigned by workers. Since each situation is equally likely, according to the previous description of mutual information, the following calculation can be made:

$$Mut = \sum_{n1}\sum_{m1} \frac{1}{n1 * m1} log \frac{1/(n1 * m1)}{(1/n1)(1/m1)} = 0. \quad (20)$$

Mutual information represents the same amount of information contained in two random variables. If all the cases are equally likely to occur, no helpful information can be obtained from the data, so the mutual information is 0.

Theorem 3: When the number of tasks is unknown, if the occurrence of each task location combination and each task assignment result have an equal probability, the amount of information in this case is 0.

Proof: The assumption is the same as *Theorem 2*, except that the number of active tasks is unknown. In this case, there are

$n2 = 2^{N_t}$ possible combinations of correct tasks and $m2 = (N_t + 1)^{N_w}$ possible results assigned by workers. The rest of the proof is the same as *Theorem 2*.

E. Evaluation of Results

How to measure the performance of the results chosen from the codebook is also a question we need to consider. In this article, two different adversary models were mentioned earlier; for the two adversary models, we calculate the performance of the result in different ways.

1) For Indirect Attacks: In an indirect attack, the attacker can only observe the allocation of tasks by the platform. The codewords stored in the codebook are also the results of task assignment. Therefore, the performance codeword selection from the codebook can be considered from two perspectives: the quality and fairness of task completion and the resources consumed by workers in the process of task completion. In terms of quality and fairness, we consider the gap between a given result and the optimal result, and in terms of resource loss, we consider the average loss. The sum of the two is a description of system performance, and the smaller this value is, the better the selection result. For true task state $\bar{l}$ and the selected codeword v, the calculation of performance is expressed as follows:

$$\text{performance} = (N - Q_{\text{sum}}(v)) + (C_{\text{sum}}(\bar{l}, v)/w_{\text{num}}) \quad (21)$$

where w_{num} is the number of workers and N is the maximum value of quality and fairness that can be achieved in the ideal state for fixed numbers of real tasks and workers.

2) For Intrusive Attacks: For intrusive attacks, powerful attackers directly hack into the platform database and observe the distribution of the real locations of tasks. Therefore, system performance needs to be measured by a different indicator than that used in an indirect attack. Here, we only need to consider the gap between the observed situation and the real situation, that is, how many observed tasks are associated with the correct location and correspond to the real situation. The specific calculation is as follows:

$$\text{utility} = t_{\text{num}} - \text{obs}_{\text{num}} \quad (22)$$

where t_{num} is the number of tasks and obs_{num} is the number of correct task locations observed. It is important to note that the utility defined in this case is only related to quantity, so the value can only be an integer.

VI. Performance Evaluation

To verify the performance of the SAC method, a simulation experiment is designed and discussed based on different attack models.

A. Simulation Setting

We consider an MCS task allocation problem in a 5 km by 5 km area and divide the area map into regions 1 km in length and 1 km in width. The worker and task locations are randomly distributed and not completely fixed. We use small areas of the map to represent task location coordinates, and they are numbered from top to bottom, left to right and one to num, where num represents the total number of locations and workers. In this article, considering two different adversary models, there are two attacker knowledge cases based on the number of real tasks: the number of tasks is fixed and known, and the number of tasks is unknown. These two situations are analyzed according to different adversary models. Considering that the detection results of ESC are not completely accurate, the case in which the ESC false detection rate is 1% is considered in this article, and the figures of the corresponding results are drawn.

In actual experiments, even a small number of tasks and workers can combine to produce a large number of results. Therefore, a convenient and straightforward method is needed to number or name each result to facilitate the experiment. In this article, we use the decimal equivalent of binary numbers to represent the various possible cases, and suppose there are three possible task positions and four workers. Each task location is represented by a binary number, where 1 indicates a real task and 0 means there is none. For example, if the first and second positions have a task, and the third position does not, the binary representation of this situation is 011 and is 3 when converted to decimal. Each worker in the task allocation result is represented by two binary numbers representing the task number allocated to the worker. For example, the result is that worker 1 and worker 4 are assigned to position 1 (the position represented by 01) to perform the task, while worker 2 and worker 3 are assigned to position 2 (the position represented by 10), which can be encoded as the binary number 01101001 and converted to decimal as 105.

B. For Indirect Attacks

In indirect attacks, the attacker may appear to be a worker or break into the workers' system to observe how the platform allocates tasks and workers, i.e., $\mathcal{A} = P_t^w$.

In addition to the SAC method, we also assess an optimization method and a method in which Gaussian noise is added for comparison. The optimal method chooses the best result among all possible task allocation results with only false location information added and is similar to the enumeration method. In the second method, Gaussian noise with a mean value of 0 and variance of 3 is added to the task location information. These two methods calculate and select among all possible task assignment results so that they always obtain the best result under their respective settings. Therefore, the result does not change with the number of codebook optimization steps but only serves as a reference value for the SAC method. There are two design parameters in the experiment: the codebook size and the number of codebook optimization steps. Experiments with three different codebook sizes, namely, 4, 8, and 16, are performed. Here, the number of codebook optimization steps of 0 means that the codewords in the codebook are randomly selected.

1) The Number of Tasks is Fixed and Known: Suppose there are two tasks: that is, in each period, there are sensing tasks that can be performed at two of all possible task locations on the map.

Fig. 3(a) shows the effectiveness of task allocation obtained based on varying numbers of codebook optimization steps and

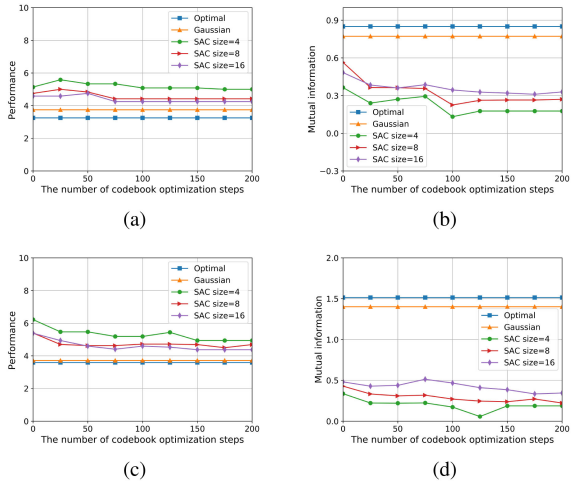


Fig. 3. For indirect attacks and for different numbers of codebook optimization steps, codebook sizes, and numbers of tasks: (a), (c) performance of the task allocation scheme; (b), (d) the level of privacy exposure.

different methods. It should be noted that the optimal method and Gaussian mechanism approach do not involve codebook optimization, and they are compared with the SAC method; therefore, the corresponding results do not change as the number of codebook optimization steps varies. As shown in the figure, for the same number of codebook optimization steps, the larger the codebook size is, the better the final result. Notably, a larger codebook contains more choices than a small codebook. As the number of codebook optimization steps increases, better codewords are retained to generate better codebooks, so the results obtained by the SAC method become increasingly better. The optimal method always obtains the best result, and the results of the Gaussian mechanism approach rank second. Fig. 3(b) shows the result of mutual information, that is, the degree of privacy exposure obtained with different methods. The larger the codebook is, the more information it contains, and the larger the value of mutual information. Because the optimal method and Gaussian mechanism approach choose among all the results of assigned tasks, their mutual information values are large.

2) The Number of Tasks is Unknown: In some cases, the number of tasks is unknown because each of the task locations on a map may or may not have a task, so the exact number of tasks is unknown. Other settings are the same as in the previous section. Similarly, Fig. 3(c) shows that for the same number of codebook optimization steps, the larger the codebook size is, the better the final result will be. As the number of codebook optimization steps increases, the results obtained by the SAC method improve. It can be concluded from Fig. 3(d) that the smaller the codebook is, the smaller the privacy exposure level.

The performance and mutual information performance indexes are comprehensively considered to create Fig. 4(a) and (b). Recall that zero mutual information represents maximum privacy protection and the minimum performance value represents maximum system utility, so operations in the lower-left corner of the figure are optimal. As illustrated, the SAC method achieves a tradeoff between performance and privacy protection, regardless of whether the number of tasks is fixed and known

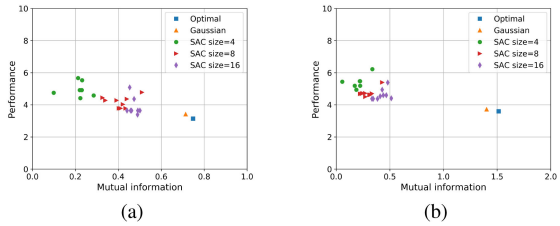


Fig. 4. Task allocation strategy comparison for an indirect attack: (a) the number of tasks is fixed and known and (b) the number of tasks is unknown.

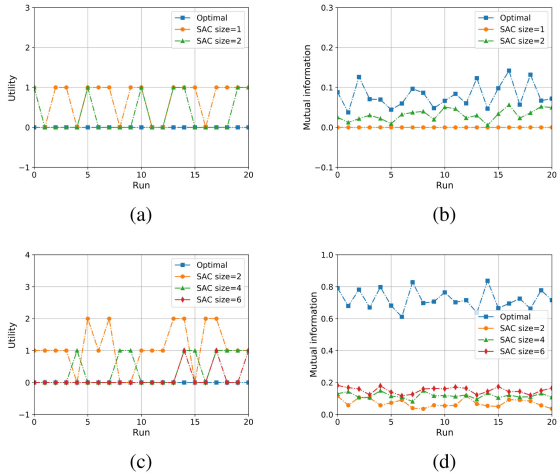


Fig. 5. For intrusive attacks and for different numbers of codebook optimization steps, codebook sizes, and numbers of tasks: (a), (c) performance of the task allocation scheme; (b), (d) the level of privacy exposure.

or unknown. Although the optimal method and Gaussian mechanism approach guarantee utility, privacy exposure is a serious issue. In contrast, the SAC method performs slightly worse in terms of system performance but somewhat better in terms of privacy protection.

C. For Intrusive Attacks

Intrusive attacks are different from indirect attacks in that the attacker in this adversary model is more powerful and can directly invade the platform database. Therefore, the attacker obtains information regarding whether there is a task associated with the corresponding location, i.e., $\mathcal{A} = \hat{\mathcal{L}}_t, \mathcal{L}_t^{\text{esc}}$.

1) The Number of Tasks is Fixed and Known: The attacker in an intrusive attack directly invades the database to directly obtain the information related to task locations. Similarly, if there are two sensing tasks that need to be performed, because the result space is small, the number of codebook optimization steps will be greatly limited. Therefore, the number of codebook optimization steps is not considered here, and the SAC method uses two different codebook sizes: 2 and 3.

Fig. 5(a) shows the utility of the results of the optimal method for different codebooks. It can be concluded that the utility of the result is related to the size of the codebook. The larger the codebook is, the more options there are, and the better the result. Based on the information shown in Fig. 5(b), the larger

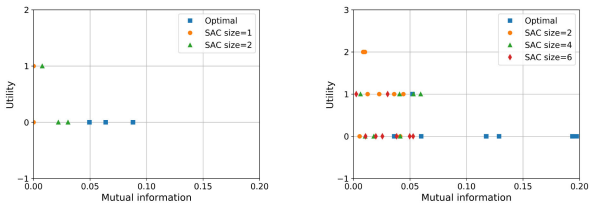


Fig. 6. Task allocation strategy comparison for an intrusive attack: (a) the number of tasks is fixed and known; (b) the number of tasks is unknown.

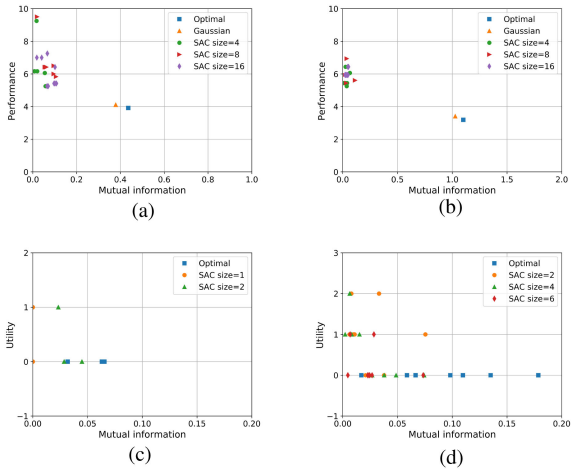


Fig. 7. The false detection rate is 1%: (a), (c) the number of tasks is fixed and known; (b), (d) the number of tasks is unknown. (a) For indirect attacks. (b) For indirect attacks. (c) For intrusive attacks. (d) For intrusive attacks.

the codebook is, the more information it can contain, suggesting that the value of mutual information is high and the effectiveness of privacy protection is poor.

2) The Number of Tasks is Unknown: Since the number of tasks is unknown, there are eight possible results. Similarly, due to the small result space, the number of codebook optimization steps is limited and not considered here. Compared with that in the case in which the number of tasks is fixed and known, the result space is slightly larger in the case in which the number of tasks is unknown. Therefore, we consider three different codebook sizes for the SAC method: 2, 4, and 6.

The conclusions obtained from Fig. 5(c) and (d) are the same as those described earlier. For a large codebook, the utility of the result selected from the codebook is high, but privacy protection is poor.

Fig. 6(a) and (b) are based on the two experimental indicators. Again, operations in the lower-left corner of the graph are optimal. As shown in the two figures, the optimization method achieves the best utility, but the mutual information value is large and the privacy protection result is poor. Although the SAC method loses some utility value in comparison, privacy protection is considerably improved. Therefore, it is concluded that the SAC method achieves a tradeoff between system performance and privacy protection.

In addition, Fig. 7 shows the results of two adversary models when the ESC false detection rate is 1%. As seen from the figures, the mutual information when the false detection rate is 1% is smaller than that when the false detection rate is 0%, as described above. This is because error detection helps to hide some of the actual information. The performance of the allocation result is slightly worse because the selection of the allocation result based on the error detection information will not produce a correct or better result.

VII. Conclusion

In this article, we first discuss location privacy protection for tasks in MCS systems. The concept of mutual information in information theory was applied to measure the degree of privacy exposure in such systems. Using an obfuscation mechanism, a general MCS model was designed to protect task location privacy. However, to some extent, the obfuscation mechanism reduces the quality of data published by the requester to the platform. Therefore, an optimization problem was proposed, the optimal solution is theoretically derived, and the optimal solution to the task allocation problem was obtained. The SAC method proposed in this article avoids many computational processes and achieves efficient sensing task allocation. The simulation results showed that the proposed approach achieves a tradeoff between task location privacy protection and system utility. Considering that tasks are heterogeneous, building a more efficient task privacy protection model and task assignment method to improve the performance of task assignment results in the MCS system is our future work.

References

[1] K. Abualsaud *et al.*, "A survey on mobile crowd-sensing and its applications in the IoT Era," *IEEE Access*, vol. 7, pp. 3855–3881, 2018.

[2] X. Wang *et al.*, "A city-wide real-time traffic management system: Enabling crowdsensing in social internet of vehicles," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 19–25, Sep. 2018.

[3] X. Li and D. W. Goldberg, "Toward a mobile crowdsensing system for road surface assessment," *Comput., Environ. Urban Syst.*, vol. 69, pp. 51–62, 2018.

[4] A. S. El-Wakeel, J. Li, A. Noureldin, H. S. Hassanein, and N. Zorba, "Towards a practical crowdsensing system for road surface conditions monitoring," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4672–4685, Dec. 2018.

[5] S. H. Marakkalage *et al.*, "Understanding the lifestyle of older population: Mobile crowdsensing approach," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 1, pp. 82–95, Feb. 2018.

[6] L. Liu, W. Liu, Y. Zheng, H. Ma, and C. Zhang, "Third-eye: A mobilephone-enabled crowdsensing system for air quality monitoring," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 1–26, 2018.

[7] 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," *ACM Sigmod Rec.*, vol. 44, no. 4, pp. 23–34, 2016.

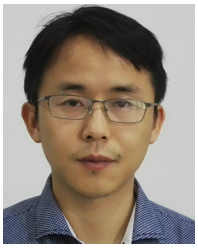
[8] D. Peng, F. Wu, and G. Chen, "Data quality guided incentive mechanism design for crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 307–319, Feb. 2018.

[9] M. Xiao, G. Gao, J. Wu, S. Zhang, and L. Huang, "Privacy-preserving user recruitment protocol for mobile crowdsensing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 519–532, Apr. 2020.

[10] X. Wang, Z. Liu, X. Tian, X. Gan, Y. Guan, and X. Wang, "Incentivizing crowdsensing with location-privacy preserving," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6940–6952, Oct. 2017.

[11] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma, "Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 627–636.

[12] Z. Wang *et al.*, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 6, pp. 1330–1341, Jun. 2019.
[13] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif, "Reliable and privacy-preserving truth discovery for mobile crowdsensing systems," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1245–1260, May/Jun. 2021.
[14] L. Wang, D. Zhang, D. Yang, B. Y. Lim, X. Han, and X. Ma, "Sparse mobile crowdsensing with differential and distortion location privacy," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2735–2749, Feb.24, 2020, to be published, doi: 10.1109/TIFS.2020.2975925.
[15] X. Zhang, J. Ding, X. Li, T. Yang, J. Wang, and M. Pan, "Mobile crowdsensing task allocation optimization with differentially private location privacy," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[16] J. Xiong, M. Zhao, M. Z. A. Bhuiyan, L. Chen, and Y. Tian, "An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 922–933, Feb. 2021.
[17] J. Xiong *et al.*, "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4231–4241, Jun. 2020.
[18] S. Zou, J. Xi, H. Wang, and G. Xu, "CrowdBLPS: A blockchain-based location-privacy-preserving mobile crowdsensing system," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4206–4218, Jun. 2020.
[19] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 901–914.
[20] S. Oya, C. Troncoso, and F. Pérez-González, "Back to the drawing board: Revisiting the design of optimal location privacy-preserving mechanisms," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1959–1972.
[21] W. Zhang, M. Li, R. Tandon, and H. Li, "Online location trace privacy: An information theoretic approach," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 1, pp. 235–250, Jan. 2019.
[22] M. Clark and K. Psounis, "Optimizing primary user privacy in spectrum sharing systems," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 533–546, Apr. 2020.
[23] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2254–2262.
[24] Y. Wu, F. Li, L. Ma, Y. Xie, T. Li, and Y. Wang, "A context-aware multiarmed bandit incentive mechanism for mobile crowd sensing systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7648–7658, Oct. 2019.
[25] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E*, vol. 69, no. 6, 2004, Art. no. 066138.



Xuewen Dong received the B.E., M.S., and Ph.D. degrees in computer science and technology from the Xidian University of China, Xi'an, China, in 2003, 2006, and 2011, respectively.

From 2016 to 2017, he was a Visiting Scholar with the Oklahoma State University, Stillwater, OK, USA. He is currently a Professor with the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include cognitive radio network, wireless network security, and blockchain.

Wen Zhang received the B.E. degree in information and computing science from the Taiyuan University of Technology, Taiyuan, China, in 2019. She is currently working toward the M.S. degree in computer science and technology with Xidian University, Xi'an, China.

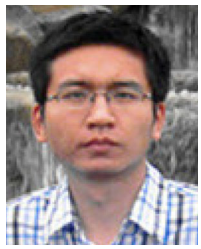
Her research interests include wireless network security and privacy computing.

Yushu Zhang (Member, IEEE) received the Ph.D. degree in computer science and technology from the College of Computer Science, Chongqing University, Chongqing, China, in 2014.

He held various research positions with Southwest University, Chongqing, China, City University of Hong Kong, Hong Kong, China, University of Macau, Macau, China, and Deakin University, Victoria, Australia. He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He is currently an Associate Editor for the *Information Sciences and Signal Processing*. He has authored or coauthored more than 100 refereed journal articles and conference papers in these areas. His current research interests include multimedia security, blockchain, and artificial intelligence.

Zhichao You received the B.E. degree in information and computing science from South China Agricultural University, Guangzhou, China, in 2018, and the M.S. degree in computer science and technology in 2021 from Xidian University, Xi'an, China, where he is currently working toward the Ph.D. degree in the School of Computer Science and Technology, Xidian University, Xi'an, China.

His research interests include federated learning and wireless network security.



Sheng Gao received the B.S. degree in information and computation science from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2009, and the Ph.D. degree in computer science and technology from Xidian University, Xi'an, in 2014.

He is currently an Associate Professor with the School of Information, Central University of Finance and Economics, Beijing, China. He has authored or coauthored more than five books and more than 40 papers in refereed international journals and conferences. His current research interests include blockchain, federated learning, data security, and privacy computing.

Yulong Shen (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2002, 2005, and 2008, respectively.

He is currently a Professor with the School of Computer Science and Technology, Xidian University, and also an Associate Director with the Shaanxi Key Laboratory of Network and System Security. He was also on the technical program committees of several international conferences, including NANA, ICEBE, INCoS, CIS, and SOWN. His research interests include wireless network security and cloud computing security.

Chao Wang received the B.S. and M.S. degrees in mechanical design and theory from the Hefei University of Technology, Hefei, China, in 1999 and 2002, respectively, and the Ph.D. degree in computer application technology from Xidian University, Xi'an, China, in 2006.

He is currently an Associate Professor with Xidian University. His research interests include SNS and information security.