A Novel Location Privacy-Preserving Task Allocation Scheme for Spatial Crowdsourcing

Xuelun Huang[1], Shaojing Fu[1,2(✉)], Yuchuan Luo[1], and Liu Lin[1]

[1] College of Computer, National University of Defense Technology, Changsha, China
[2] Sate Key Laboratory of Cryptology, Beijing, China

Abstract. With the increasing popularity of big data and sharing economics, spatial crowdsourcing as a new computing paradigm has attracted the attention of both academia and industry. Task allocation is one of the indispensable processes in spatial crowdsourcing, but how to allocate tasks efficiently while protecting location privacy of tasks and workers is a tough problem. Most of the existing works focus on the selection of the workers privately. Few of them present solutions for secure problems in task delivery. To address this problem, we propose a novel privacy protection scheme that not only protects the location privacy of workers and tasks but also enables secure delivery of tasks with very little overhead. We first use the paillier homomorphic cryptosystem to protect the privacy of workers and tasks, then calculate travel information securely. Finally, let workers restore the tasks' location. In our scheme, only workers who meet the requirements can get the exact location of tasks. In addition, we prove the security of our method under the semi-honest model. Extensive experiments on real-world data sets demonstrate that our scheme achieves practical performance in terms of computational overhead and travel cost.

Keywords: Spatial crowdsourcing · Privacy-preserving · Homomorphic cryptosystem · Task allocation

1 Introduction

Crowdsourcing has gradually attracted the attention of all walks of life since Jeff Howe, a reporter for the Wired magazine, proposed it in 2006 [5]. Jeff Howe defines crowdsourcing as a company or organization posting problems on the network to collect better solutions. Nowadays, many crowdsourcing platforms (e.g., Amazon Mechanical MTurk1, TaskRabbit2) have been established to provide various kinds of crowdsourcing services. Mobile crowdsensing and spatial crowdsourcing also emerge as the times. Both of them require the participation of a large number of users, reduces costs, and leverages the advantages of the network to accumulate the resource of the public under different knowledge backgrounds. However, mobile crowdsensing focuses more on the use of mobile

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 304–322, 2021.
https://doi.org/10.1007/978-3-030-89814-4_23

devices for data perception, collection, and analysis, which does not pay attention to how tasks are allocated. Many mobile crowdsensing tasks are distributed and utilized through crowdsourcing. Therefore, solving task allocation problem in spatial crowdsourcing is also helpful to mobile crowdsensing.

With the rapid development of mobile internet technology, spatial crowdsourcing has also become popular. Spatial crowdsourcing plays a critical role in various fields, such as news, tourism, intelligence, disaster response, and urban planning [17]. Take real-time traffic condition monitoring as an example, it affects people's daily travel and lifestyle at all times. By obtaining the spatial distribution information of users at different times and corresponding various sensor data, such software can analyze and speculate real-time traffic conditions. When users use this software, they passively become crowdsourcing workers and share their spatiotemporal information and sensor data. In contrast to general crowdsourcing, spatial crowdsourcing adds more location requirements that need workers to reach the designated location to complete the task.

At the same time, people have paid attention to information security, hoping not only to enjoy the convenience of emerging techniques but also to protect their private information. When using crowdsourcing, the crowdsourcing platform needs the information of tasks and workers to perform task allocation, which usually contains a lot of private information. If it leaks out private information, disastrous consequences will spring out. For example, when users are enjoying the taxi service, they need to tell drivers where they are and where they want to go, but users do not want the platform to know. Because location data of users may indicate their home addresses, lifestyle, and other sensitive information. Attackers would know the real-time location of users once they grasped these privacies. These security risks depress the availability of crowdsourcing and may let some people refuse to use crowdsourcing. Thus, it's significant to allocate tasks efficiently at the premise of protecting privacy. To solve this problem, many feasible solutions have been proposed. [13] used homomorphic encryption and Yao's garbled circuits to achieve a secure task distribution. But it only protects the privacy of workers, without considering the privacy of tasks. However, task information will indirectly reveal the workers' information. [21] designed a grid-based position protection method for task distribution. But the distribution process involves heavy encryption and decryption operations, which is not efficient for practice. [23] proposed a novel spatial crowdsourcing framework without trusted third parties but providing differential privacy guarantees. But workers need to set up an acceptable location in advance, which requires a relatively sizeable storage space. [22] designed a data aggregation protocol based on k-anonymity, but it cannot well resist malicious deception of workers.

Most of the works focus on how to allocate tasks more securely and efficiently but ignore the next step after task allocation: how to deliver tasks to workers securely and efficiently. The major contributions of this article are as follows:

1. We propose a scheme for spatial crowdsourcing task allocation. Based on the two-server model and an additively homomorphic cryptographic cryptosystem, the proposed scheme protects the location privacy of workers and tasks without involving any online trusted third party (TTP).

2. We put forward a novel method to deliver tasks and calculate the travel information securely. This method ensures that only workers who meet the requirements can get the location information of tasks and also allows workers to reconstruct tasks' location with practical efficiency.
3. Security analysis in our paper indicates that our scheme can protect the location information about the works and tasks, as well as the data access patterns. Experimental results demonstrate that our scheme achieves practical performance in terms of computational overhead and travel cost.

We organize the rest of the paper as follows. Section 2 presents preliminaries, and Sect. 3 gives the problem formulation. Section 4 gives the proposed scheme. We present the security and performance analyses in Sect. 5 and introduce related work in Sect. 6. In Sect. 7, we conclude our work.

2 Preliminaries

In this section, we review the concepts and general procedures of the Paillier cryptosystem and spatial crowdsourcing, then introduce notations of this article.

2.1 Paillier Cryptosystem

The Paillier cryptosystem is a probabilistic public key encryption system invented by Paillier in 1999 [11]. The encryption algorithm is a homomorphic public key encryption system that satisfies addition and number multiplication homomorphism. Firstly, randomly select two large prime numbers p and q, which satisfy $gcd(pq,\ (p-1)(q-1)) = 1$. Then calculate $n = pq$ and $\lambda = lcm(p-1,\ q-1)$. And randomly select an positive integer g ($g \in Z^*_{n^2}$), which is less than n^2. Define $L(x) = \frac{(x-1)}{n}$, and there exists $\mu = (L(\ g^\lambda\ mod\ n^2))^{-1} mod\ n$. The public key PK is (n, g), the private key SK is (λ, μ). In encryption, randomly select a number $r \in Z_n$ and calculate the ciphertext $c = g^m \cdot r^n\ mod\ n^2$, where m is the original message. In decryption, calculate the plaintext $m = \frac{L(c^\lambda\ mod n^2)}{L(g^\lambda\ mod n^2)}\ mod\ n$.

We can express the homomorphic properties as : $c_1 \cdot c_2 = E[m_1, r_1] \cdot E[m_2, r_2] = g^{m_1+m_2}(r_1 \cdot r_2)^n mod\ n^2$, $D[c_1 \cdot c_2] = D[E[m_1, r_1]E[m_2, r_2] mod\ n^2] = m_1 + m_2\ mod\ n$. Here, $r_1, r_2 \in Z_n$ is a random number; $m_1, m_2 \in Z_n$ is plaintext; c_1, c_2 is the ciphertext of m_1, m_2.

2.2 Spatial Crowdsourcing

Spatial crowdsourcing applications are already very common in our daily life, such as Gigwalk, Easyshift, and Fieldagent, *etc.* It has a wide range of applications, but when people enjoy the happiness and convenience brought by these software, people inadvertently reveal a lot of their private information.

Spatial crowdsourcing often includes three parties, requesters, workers, and crowdsourcing platforms. First, requesters publish tasks on the crowdsourcing platform, then the crowdsourcing platform allocates tasks according to tasks'

locations and workers' locations. And then, workers accept and complete tasks. There are two modes of spatial crowdsourcing task allocation [18]. One of them is the Server Assigned Tasks (SAT), which is a platform server-centric model. In this mode, the platform server assigns a nearby task to each worker after receiving the locations of all workers. Therefore, it is possible to assign nearby tasks to each worker when maximizing the overall number of tasks assigned. However, sending workers' location to the server may cause privacy threats. The other one is Worker Selected Tasks (WST), which is a user-centric model. Platform servers often issue space-aware tasks. Online workers can choose any spatial task without consulting with the server. Users submit less personal information, which can increase the participation of mobile users. However, some spatial tasks may never be assigned, and other tasks are assigned redundantly. And may not form a global optimal allocation (Table 1).

2.3 Notations

Table 2 presents several symbols for better readability.

Table 1. Notations

Notation	Meaning
w_i, t_i	Worker i, task i
(PK, SK)	Public key, Secret key
x_{w_i}, y_{w_i}	Coordinates of w_i, $0 < \mathrm{i} \leq \mathrm{m}$
x_{t_i}, y_{t_i}	Coordinates of t_i, $0 < \mathrm{i} \leq \mathrm{n}$
$D[\cdot]$	Decryption operation
$E[x_{w_i}], E[y_{w_i}]$	Encrypted coordinates of w_i
$E[dx_{w_i,t_i}], E[dy_{w_i,t_i}]$	The encrypted difference between w_i and t_i in the x, y direction
Dis_{w_i,t_i}	Distance between w_i and t_i
C_t, C_w	Task set, Worker set
sort	A function implements sorting from small to large
available	A function filtering out workers who are not currently available

3 Problem Formulation

3.1 System Model

Our system model is shown in Fig. 1. In the setting of our privacy-preserving scheme, we have four entities:

– **Requester :** A task requester is a user who first publishes tasks to CSP. The requester then waits for CSP to assign workers. Then wait for the workers to finish the task. For instance, in the taxi service, the taxi passenger is the requester, and he waits for the platform to assign him a driver.

- **Worker :** A task worker is a user who receives tasks. He decides whether to accept tasks according to his own will. When he accepts a task, he will finish the task quickly and efficiently. For instance, in the taxi service, the taxi driver is the worker, and he waits for the platform to allocate passengers to him.
- **CSP (Crowdsourcing Service Provider) :** Upon receiving a task, CSP cooperates with S to calculate the travel information. Also responsible for generating lists of candidate workers and interacting with workers.
- **S (Server) :** S is responsible for key generation and distribution. Assist CSP to complete travel information calculation. Also responsible for interacting with workers.

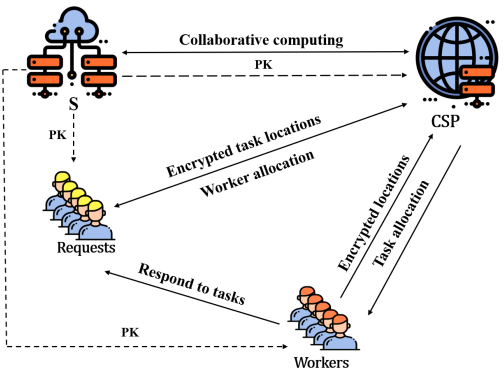

Fig. 1. System model

3.2 Threat Model

The threat model for each entity is set as follows:

- **Users:** (requesters and workers) are considered as fully trusted in the scheme where they could execute the operations properly and protect their locations and secret keys. Users wouldn't leak their locations actively or passively to the other entity, *e.g.* CSP or S.
- **CSP & S** are considered as $honest - but - curious$ in the sense that they could execute the designed operations honestly. We also assume that they cannot collude with each other and wouldn't launch active attacks, such as collusion with users, pretending to be a requester (or a worker).

The assumption for CSP and S is reasonable (e.g. [1]), because most of the cloud service providers in the market are well-established IT companies and they understand the importance of reputation [8,9]. Active attacks are straightforward to detect and may damage their reputation once caught. Collusion between them is highly unlikely as it may damage their reputation and affect their revenues.

4 The Proposed Scheme

4.1 Overview

Our system model is shown in Fig. 1. The scheme consists of four parts, namely **KeyGen**, **DataEnc**, **TaskAllocation**, **TaskDelivery**.

- **KeyGen** $\rightarrow (PK, SK)$: S generate PK and SK for the Paillier Cryptosystem. Sending PK to CSP, requests, and workers, SK to CSP.
- **DataEnc** $\rightarrow (E[x_i], E[y_i])$: Workers encrypt their locations, and requesters encrypt the locations of tasks. Then they all send the encrypted locations to CSP.
- **TaskAllocation** $\rightarrow List(t, w)$: CSP receives requesters' tasks, then cooperates with S to calculate the travel message between tasks and workers, generating candidate worker lists based on this.
- **TaskDelivery** $\rightarrow (tan_{w_i,t_i}, signx_{w_i,t_i}, signy_{w_i,t_i})$: CSP notifies workers in the candidate worker list in turn until the task is assigned. Workers receive travel information about tasks after they accept tasks.

4.2 Scheme Details

4.2.1 Task Allocation

1) Distance Calculation. After receiving the encrypted tasks' locations and workers' locations, the CSP cooperates with S to calculate the distance between tasks and workers. The calculation procedures are shown in algorithm1.

First, CSP uses the homomorphic encryption property of the Paillier cryptosystem to calculate the coordinate distance values. Through step 1 completed by CSP, we can easily see that $E[dx_{w_i,t_i}] = E[x_{w_i} - x_{t_i}], E[dy_{w_i,t_i}] = E[y_{w_i} - y_{t_i}]$.

In order not to let S get any information about the location, then choose two random numbers $r_1, r_2 \in Z_N$, adding them to the result and send the result to S. Upon receiving a, b, c, d from CSP, because S has the decryption key SK, it can decrypt and calculate the corresponding squared distance, that is $ans1 = E[(dx_{w_i,t_i} + r_1) \cdot (dy_{w_i,t_i} + r_2)]$, $ans2 = E[(dy_{w_i,t_i} + r_1) \cdot (dy_{w_i,t_i} + r_2)]$.

So S cannot get the real distance value. CSP can use the properties of homomorphic encryption to remove the influence of $r1$ and $r2$ in the results. $Ans3$ and $ans4$ are the squared distances in the x and y directions of the real distance between workers and tasks. The decryption result of $ans3 \cdot ans4$ is the distance squared value. Same as before, CSP selects a random number to prevent S speculation and finally can get Dis_{w_i,t_i}.

2) Candidates List Generation.
Each worker and task can use Algorithm 1 to calculate the squared distance between the encrypted locations. Each requester submits the maximum acceptable distance D when submitting the task location. Only when the distance between workers and tasks is within the range of the maximum acceptable distance D, workers are eligible for the task. Therefore, by ranking the distance

Algorithm 1: Distance Calculation

Input: $E[x_{w_i}], E[y_{w_i}], E[x_{t_i}], E[y_{t_i}]$
Output: Dis_{w_i,t_i}
1. CSP:
   $E[dx_{w_i,t_i}] \leftarrow E[x_{w_i}] \cdot E[x_{t_i}]^{-1}$, $E[dy_{w_i,t_i}] \leftarrow E[y_{w_i}] \cdot E[y_{t_i}]^{-1}$;
   Pick two random numbers r_1, r_2;
   $a \leftarrow E[dx_{w_i,t_i}] \cdot E[r_1]$; $b \leftarrow E[dx_{w_i,t_i}] \cdot E[r_2]$;
   $c \leftarrow E[dy_{w_i,t_i}] \cdot E[r_1]$; $d \leftarrow E[dy_{w_i,t_i}] \cdot E[r_2]$;
   send a, b, c, d to S;
2. S:
   $a' \leftarrow D[a]$; $b' \leftarrow D[b]$; $c' \leftarrow D[c]$; $d' \leftarrow D[d]$;
   $ans1 \leftarrow E[a' \cdot b']$; $ans2 \leftarrow E[c' \cdot d']$, send $ans1, ans2$ to CSP;
3. CSP:
   Pick one random number r_3;
   $ans3 \leftarrow ans1 \cdot E[r_1 \times r_2]^{-1} \cdot E[dx_{w_i,t_i}]^{-r_1} \cdot E[dx_{w_i,t_i}]^{-r_2}$;
   $ans4 \leftarrow ans2 \cdot E[r_1 \times r_2]^{-1} \cdot E[dy_{w_i,t_i}]^{-r_1} \cdot E[dy_{w_i,t_i}]^{-r_2}$;
   $ans5 \leftarrow ans3 \cdot ans4 \cdot E[r_3]$, send $ans5$ to S;
4. S:
   Receive $ans5$ from CSP;
   $dis \leftarrow D[ans5]$, send dis to CSP;
5. CSP:
   $Dis_{w_i,t_i} \leftarrow dis - r_3$;

between tasks and workers, CSP can generate a list of candidate workers. To facilitate subsequent calculations, CSP records the distance values between workers and tasks in workers' database and tasks' database, respectively. We should note it as it randomly generates the random numbers mentioned in the algorithm in each cycle.

3) Task Allocation.
We devise efficient task allocation algorithms for both SAT and WTS. After the previous algorithm process, each task has a corresponding list of candidate workers. Therefore, the server allocation can directly notify workers in order according to the candidate worker list until the task is assigned. The procedures are shown in Algorithm 2.

In the SAT, the maximum number of candidate workers $maxNt$ is set to prevent too many workers who meet the requirements from affecting efficiency. Use counter to mark the number of selected workers and $t.d_{task}$ is the maximum distance D that the task t can accept. After the processing of **sort** and **available**, CSP can get an optional and orderly set of workers SC_w. After many cycles, each task can get a list of optional workers. It is worth noting that a worker may exist in multiple candidate worker lists at the same time. When a worker is invited to a task, the worker will decide whether to accept the task. And we assume that the workers in this article can only accept one task at a time, and each worker is busy and idle. When the worker accepts the task, his status will become busy, and the worker in the other candidate list will be invalid.

Algorithm 2: Task Allocation - SAT

```
Input: C_t, C_w, maxNt
Output: List_w = (t, w)
for each task t in C_t do
    counter = 0;
    SC_w = available ( sort ( C_w ) );
    for each worker w in SC_w do
        if w.d[t] > t.d_task then
            Break;
        else if counter == maxNt then
            Break;
        else
            List ← (t, w);
            ++counter;
    end
end
return List;
```

The worker mode is that the worker receives all the distance of tasks he satisfies, then chooses the task subjectively. But what criteria the worker uses to select the task based on the candidate task list is not the focus of this article, and this affects the subsequent implementation of the algorithm, so we just briefly introduce this algorithm.

In the WTS, $maxNw$ is the maximum number of tasks that the worker can select, and $w.d_{worker}$ is the maximum distance that the worker w can accept. After going through the previous algorithm, some workers may have some optional tasks in the database. Then these workers can choose tasks, which means that they must decide according to their preferences. For workers with optional tasks, after implementing Algorithm 3, they can get a list of candidate tasks ordered from near to far.

4.2.2 Task Delivery

1) Pre-processing.

The Paillier cryptosystem is carried out on positive integers, but the latitude and longitude coordinates are mostly floating-point. It also involves positive and negative numbers. Therefore, we need to convert the latitude and longitude to positive integers before encryption, then convert it to the original latitude and longitude after decrypting. We exploit the modular arithmetic properties of the Paillier scheme. We represent only integers between $(-n/3, n/3)$. Since n is a

Algorithm 3: Task Allocation - WTS

```
Input: C_w, maxNw
Output: List_t = (t, w)
for each worker w in C_w do
    counter = 0;
    SC_t = available ( sort ( C_{w.t} ) );
    for each task t in SCt do
        if t.d[w] > w.d_worker then
            Break;
        else if counter == maxNw then
            Break;
        else
            List ← (t, w);
            ++counter ;
    end
end
return List;
```

very large number, the longitude range is 0–180° and the latitude range is 0–90°, they are included in the range. Paillier homomorphic arithmetic works modulo n. We take the convention that a number $x < n/3$ is positive and that a number $x > 2n/3$ is negative. The range $n/3 < x < 2n/3$ allows for overflow detection. Representing floating-point numbers as integers is a harder task. Here we use a variant of fixed-precision arithmetic. In fixed precision, we encode by multiplying every float by a large number (*e.g.* 1*e*6) and rounding the resulting product. We decode by dividing by that number. There are many other conversions, the specific details can be found in Sect. 5.

2) Travel Angle Calculation.

Before CSP notifies workers according to the list of candidate workers, it also needs to work with S to calculate the travel angle of workers. In this way, workers can combine their locations and travel angle to restore the locations of tasks. After pre-processing, the Paillier cryptosystem can encrypt and decrypt any real numbers in the range. Algorithm 4 shows the calculation process of workers' travel angle.

Algorithm 4: Travel Angle Calculation

Input: $E[x_{w_i}], E[y_{w_i}], E[x_{t_i}], E[y_{t_i}]$
Output: $tan_{w_i,t_i}, sign_{w_i,t_i}$
1. CSP:
$E[dx_{w_i,t_i}] \leftarrow E[x_{w_i}] * E[x_{t_i}]^{-1}$; $E[dy_{w_i,t_i}] \leftarrow E[y_{w_i}] * E[y_{t_i}]^{-1}$;
Pick random numbers r_4, r_5;
$e \leftarrow E[dx_{w_i,t_i}]^{r_4}$; $f \leftarrow E[dy_{w_i,t_i}]^{r_5}$;
send e, f to S;
2. S:
$e' \leftarrow D[e]$; $f' \leftarrow D[f]$;
$tan_{w_i,t_i} \leftarrow f'/e'$;
if $e' > 0, signx_{w_i,t_i} = 1$, else $signx_{w_i,t_i} = -1$;
if $f' > 0, signy_{w_i,t_i} = 1$, else $signy_{w_i,t_i} = -1$;
Send $tan_{w_i,t_i}, signx_{w_i,t_i}$ to CSP;
3. CSP:
if $r_4 < 0$, change $signx_{w_i,t_i}(2 \leftarrow 1, 1 \leftarrow 2)$;
$tanr_{w_i,t_i} = tan_{w_i,t_i} * r_4/r_5$;
if $tanr_{w_i,t_i} > 0$, $signy_{w_i,t_i} = signx_{w_i,t_i}$, else
$signy_{w_i,t_i} = -signx_{w_i,t_i}$;

Similar to Algorithm 1 at the beginning, we need to get the distance between the worker and the task first. The distance values calculated in Algorithm 1 have been recorded in the worker database, so we can directly call the distance result value according to the worker ID in the candidate worker list. The same is to prevent S from getting any valid information from the intermediate results, and the effect of the random number r_4, r_5 needs to be added, so e, f is calculated and sent to S. S decrypts the received information, calculates the division value tan, and sets the sign value according to the rules. It is not difficult to know that the result corresponding to tan_{w_i,t_i} is $dy_{w_i,t_i}/dx_{w_i,t_i}$, which is mathematically explained by the tan value of the trigonometric function. It should be noted that when the longitudes of the two points are the same, the denominator is 0. Although this situation does not occur frequently, to implement the scheme smoothly, we use a very small value instead of 0 in the code.

Finally, the CSP modifies the $signx_{w_i,t_i}, signy_{w_i,t_i}$ value according to the positive and negative values of r_4, r_5. And it is the next information to be sent to the workers, which is also stored in the workers' database.

3) Task Location Calculation.
After receiving messages $Dis_{w_i,t_i}, tan_{w_i,t_i}, signx_{w_i,t_i}, signy_{w_i,t_i}$, workers can estimate the location of the task.

According to $tan_{w_i,t_i} = \frac{dy_{w_i,t_i}}{dx_{w_i,t_i}} = \frac{y_{w_i} - y_{t_i}}{x_{w_i} - x_{t_i}}$.

We can get

$$\begin{aligned} dx_{w_i,t_i} = x_{w_i} - x_{t_i} = \pm\sqrt{\frac{Dis_{w_i,t_i}}{1+tan^2_{w_i,t_i}}} \\ x_{w_i} = x_{t_i} \pm \sqrt{\frac{Dis_{w_i,t_i}}{1+tan^2_{w_i,t_i}}} \end{aligned} \tag{1}$$

$$\begin{aligned} dy_{w_i,t_i} = y_{w_i} - y_{t_i} = \pm tan_{w_i,t_i} \cdot \sqrt{\frac{Dis_{w_i,t_i}}{1+tan^2_{w_i,t_i}}} \\ y_{w_i} = y_{t_i} \pm tan_{w_i,t_i} \cdot \sqrt{\frac{Dis_{w_i,t_i}}{1+tan^2_{w_i,t_i}}} \end{aligned} \tag{2}$$

The sign in the Eq. (1), (2) is determined by $signy_{w_i,t_i}, signx_{w_i,t_i}$ respectively. When $signx_{w_i,t_i} = 1$, the worker takes the positive sign, and when $sign_{w_i,t_i} = -1$, the worker takes the negative sign. So does $signy_{w_i,t_i}$.

5 Security and Performance Analysis

5.1 Security Analysis

We can summarize the security goal of our scheme as Theorem 1, 2, 3, followed by the proofs. Before providing the rigorous proofs to the privacy, we introduce the semantic security in Paillier homomorphic cryptosystems [3] and the security definition of the protocol under the semi-honest model in advance.

Definition 1 (semantic security in Paillier cryptosystem).

$$Pr\{c \leftarrow [m_1]\} - Pr\{c \leftarrow [m_2]\} \leq negl(\lambda) \tag{3}$$

In Eq. (3), m_1 and m_2 represent two plaintexts, c is the ciphertext of m_1 encrypted by paillier cryptosystem. $Pr\{c \leftarrow [m_1]\}$ is the probability that an attacker judges the message is m_1 after he observes c. $Pr\{c \leftarrow [m_2]\}$ is the probability that an attacker judges the message is m_2 after he observes c. $negl(\lambda)$ is a negligible polynomial. It means that an attacker can not distinguish m_1 from m_1 and m_2 after he observes c in Paillier cryptosystem.

Definition 2 (security in the semi-honest model [2]). Suppose a_i is the input of party P_i, $\prod_i(\pi)$ is the execution image of P_i, and b_i is the output of P_i computed from protocol π. If $\prod_i(\pi)$ can be simulated from a_i and b_i, then π is secure. In other words, distribution of the simulated image is computationally indistinguishable from $\prod_i(\pi)$.

Theorem 1. The *Distance Calculation* (*DC*) protocol described in Algorithm 1 is secure under the semi-honest model.

Proof. Here, let the execution image of CSP be denoted by $\prod_{CSP}(DC) = \{(ans1, ans2), (dis)\}$, where $dis = Dis_{w_i,t_i} + r_3$. Note that r_3 is a random number in Z_N. We assume $\prod^S_{CSP}(DC)$ means the simulated image of CSP, and $\prod^S_{CSP}(DC) = \{(ans1^s, ans2^s), (dis^s)\}$ where all the elements are randomly generated from Z_N. Since paillier cryptosystem is semantically secure, $(ans1, ans2)$ are computationally indistinguishable from $(ans1^s, ans2^s)$. Meanwhile, dis^s is randomly chosen from Z_N, dis is computationally indistinguishable from dis^s. Based on the above, we can draw a conclusion that $\prod_{CSP}(DC)$ is computationally indistinguishable from $\prod^S_{CSP}(DC)$.

Similarly, we can prove $\prod_S(DC)$ is computationally indistinguishable from $\prod^S_S(DC)$. Thus, combining the above analysis, we can confirm that DC protocol is sure under the semi-honest model.

Theorem 2. The *Travel Angle Calculation* (TAC) protocol described in Algorithm 4 is secure under the semi-honest model.

Proof. Here, let the execution image of S be denoted by $\prod_S(TAC) = \{e, f\}$. We assume $\prod^S_S(TAC)$ means the simulated image of S, and $\prod^S_S(TAC) = \{e^s, f^s\}$ where r_4, r_5 are randomly generated from Z_N. Since paillier cryptosystem is semantically secure, (e, f) are computationally indistinguishable from (e^s, f^s). Based on the above, we can draw a conclusion that $\prod_S(TAC)$ is computationally indistinguishable from $\prod^S_S(TAC)$.

Follow that familiar way, the execution image of CSP in TAC protocol is $\prod_{CSP}(TAC) = \{tan_{w_i,t_i}, signx_{w_i,t_i}, signy_{w_i,t_i}\}$. Where $signx_{w_i,t_i}, signy_{w_i,t_i}$ can regard as a random number in $\{1, -1\}$. tan_{w_i,t_i} is plain text with r_5/r_4, r_4 and r_5 are randomly generated from Z_N. So we can draw a conclusion that $\prod_{CSP}(TAC)$ is computationally indistinguishable from $\prod^S_{CSP}(TAC)$. Thus, combining the above analysis, we can confirm that TAC protocol is sure under the semi-honest model.

Theorem 3. The location privacy and the data access patterns are not be disclosed to CSP and S in our scheme. That is, CSP and S cannot infer the real location of workers or tasks from the historical records.

Proof. For location privacy, CSP gets $\{E[x_{w_i}], E[y_{w_i}]\}_{w_i \in C_w}$, $\{E[x_{t_i}], E[y_{t_i}]\}_{t_i \in C_t}$, $\{Dis_{w_i,t_i}\}_{w_i \in List_w}$, $\{signx_{w_i,t_i}\}_{w_i \in List_w}$, and $\{signy_{w_i,t_i}\}_{w_i \in List_w}$ in the whole process. S gets $\{tan_{w_i,t_i}\}_{w_i \in List_w}$ in the entire process. If CSP accidentally knows the real location of a worker, CSP can only infer the task location to which the worker is assigned. And can only guess that workers in the same candidate list is closer to the worker. If S accidentally learns the real location of a worker, S could hardly guess anything. If CSP or S know the location of a task, the situation is similar. So our scheme does not disclose location privacy to CSP and S.

From the perspective of S, S is semi-honest, and he calculates the received data according to the rules. All the data S obtains are added with random number $\{r_i\}_{1\leq i\leq 5}$ as a mask. S has no information to speculate on random numbers, so he cannot know any location information $\{(x_{w_i}, y_{w_i})\}_{w_i\in C_w}$ and $\{(x_{t_i}, y_{t_i})\}_{t_i\in C_t}$.

From the perspective of CSP, each task can get a list of candidate workers. Each worker will get $tan_{w_i,t_i}, sign_{w_i,t_i}$ of the corresponding task. According to the analysis of Theorem 1 & 2, CSP can only speculate that these candidate workers $\langle w_1, w_2, \cdots, w_k\rangle$ are located closer. But it is not possible to know the real range of its location $\{x_{w_i}, y_{w_i}\}_{1\leq i\leq k}$ and $\{x_{t_i}, y_{t_i}\}_{1\leq i\leq k}$. Therefore, the data access mode is not exposed to S and CSP.

5.2 Performance Analysis

In this section, we evaluate the proposed scheme and compare it with existing schemes.

5.2.1 Experimental Setup

In practice, take mobile taxi service as an example, the requester is a user who needs a taxi, and the worker corresponds to a taxi driver, each task usually needs only one worker. The characteristic of this application is that a worker needs to respond in time after each task is released, so it adopts the server distribution model.

Dataset: We conduct experiments on real datasets from the New York taxi website[1]. The data set is from New York taxis, and each month contains about one million pieces of information with real geographic location (using the 2015 data given on the website, the data after 2015 is not marked with latitude and longitude). First, remove duplicated coordinates, then randomly divide the data set into two parts, namely the worker and task locations. Each task requires the worker (representing the taxi driver) to arrive at the location of the user (passenger) to pick up the passenger and deliver it to the destination.

Baseline Approaches: We compare with 3 baseline approaches: (1) the method of our scheme without encryption protection. (2) PriRadar [21] designs a location protection method that maps the locations of workers and tasks to the grid. (3) our scheme.

Evaluation Metrics: We evaluate the effect of our scheme on running time, calculation error, and travel cost. The error is measured by the difference between the task coordinates restored by the worker and the real coordinates. We measure travel cost based on the distance between the last worker in the candidate worker list and the task location (represents the maximum travel cost). We should note in advance that each experimental result is an average result repeated at least 20 times.

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

Setting: All the algorithms were implemented in python 3.4, including the implementation of Paillier cryptosystem[2]. We evaluated all the experiments on window 10, with Intel Core i7 at 2.8 GHz and 16 GB RAM.

5.2.2 Theoretical Analysis

As seen from Table 2, different features realized by different schemes, only our solution can satisfy all features.

Table 2. Notations

Features	DPSC [15]	DPGSC [14]	EDSC [7]	HEEDP [4]	ours
Location privacy-preserving	✓	✓	✓	✓	✓
Proctect workers and requesters	✗	✗	✓	N/A	✓
Server is untrusted	✓	✓	✗	N/A	✓
Don't need a trusted third party	✗	✗	✓	✓	✓
Using cryptographic approach	✗	✗	✓	✓	✓
Don't fake worker locations	✗	✗	✓	✓	✓

5.2.3 Experimental Results

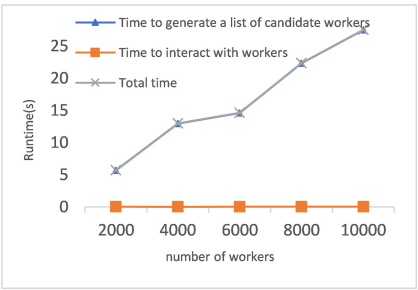

Fig. 2. Runtime

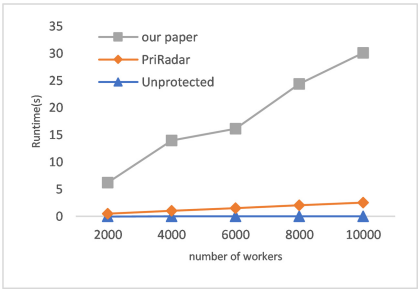

Fig. 3. Time to generate candidate workers under different tasks

Runtime. We evaluate the runtime under different numbers of workers and different schemes. Figure 2 indicates the time spent on the program under different numbers of workers. We can see that the time to generate the list of candidate workers increases continuously with the increase in the number of workers. Because each worker needs to calculate the distance from tasks' locations, and it is based on encrypted data. When interacting with workers, the time for workers to calculate the task location is not affected by the number of workers. On the one hand, because the calculations required by workers locally are very easy.

[2] https://python-paillier.readthedocs.io/en/develop/index.html.

On the other hand, because the candidate worker list sets a maximum number of workers to improve efficiency, So even if the amount of data is large, the calculation time is short. It can be found that we leave all or most of the calculations to the cloud platform, and local workers only need to perform simple calculations to get the real location of the task. Figure 3 shows the time for the two schemes to generate a list of candidate workers under different numbers of workers. We can see that the time for generating the candidate worker list in this paper is much higher. PriRadar didn't describe how to project the location on the grid in detail, so we can not know the runtime. But in the previous article, we theoretically analyzed the computational overhead. Our scheme requires fewer exponentiation operations than PriRadar [21] and has more advantages. Compared with the scheme without protection, it can be seen that the inevitable disadvantage of encryption technology is that it consumes a lot of running time. However, when the number of workers is 2,000, the average time is 6 s. When the number of workers is 10,000, the average time is almost half a minute, acceptable for a certain area. It is more reasonable to have hundreds of taxi drivers at the same time, but 10,000 drivers to be in a small area at the same time are not realistic. In reality, 10,000 drivers can be divided into small areas according to their locations. Then assign it again. It is very easy to complete.

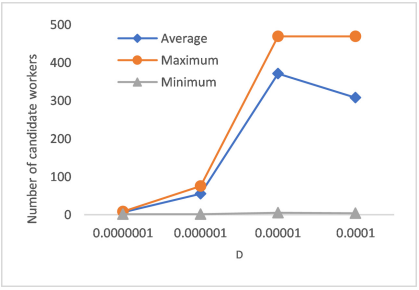

Fig. 4. Number of candidate workers under different D

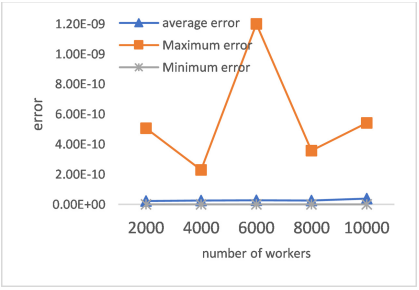

Fig. 5. Longitude error

Number of Candidate Workers. Figure 4 shows the number of candidate workers generated under different maximum acceptable distance D when the maximum number of candidate workers is not set. We can find that as D increases, the number of candidate workers also gradually increases. When $D = 0.00001$, the average number of workers is close to 400. When $D = 0.0001$, the average number of workers is higher than 300. We can see that for a task when D gradually increases, the number of qualified workers also increases. But only one worker is enough to complete a task, too many candidate workers may increase the running time. So set the maximum number of candidate workers is necessary.

Error Calculation. Figure 5 and Fig. 6 show the error value between the task location and the real task location recovered by the worker according to the information sent by CSP under different numbers of workers, Fig. 5 is longitude error

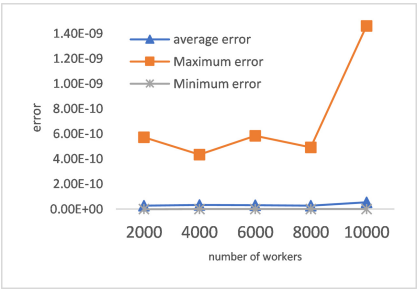

Fig. 6. Latitude error

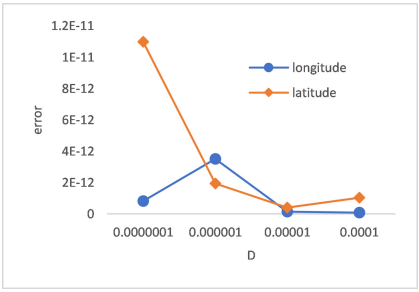

Fig. 7. Error under different D

and Fig. 6 is latitude error. It is found that the error is not significantly related to the number of workers, the minimum error is very close to 0. The average error is basically in the 10^{-10}, and the maximum error reaches the 10^{-9}. The amount of error is already tiny. Fig. 7 indicates the latitude and longitude error under different maximum acceptable distances D. We can find that there is no particularly obvious rule. The value is basically in the order of 10^{-11} to 10^{-12}. We can see that the change of D does not cause a large error change. The error is tiny. The calculations in the workers' local are easy, fast, and highly accurate.

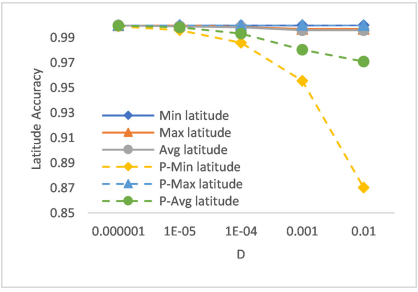

Fig. 8. Latitude travel cost of task allocation under different D

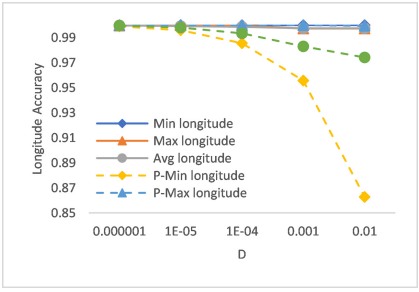

Fig. 9. Longitude travel cost of task allocation under different D

Travel Cost. Figure 8 and Fig. 9 indicate the latitude and longitude travel cost of the two schemes at different maximum acceptable distances D, calculated by the last worker in the list of candidate workers. It can be seen that with the continuous increase of the maximum acceptable distance D, the travel cost is increased, especially the PriRadar. However, our scheme has no particularly obvious impact, and the travel cost is lower. Figure 10 and Fig. 11 indicate the latitude and longitude travel distances under different workers. It can be seen that our travel distance is much smaller than in PriRadar [21]. The distance between the worker and the task is shorter, then when the worker receives the task, he is more willing to receive the task, and can quickly respond and complete the task. It also indicates that the travel cost of the task allocation of our scheme is less.

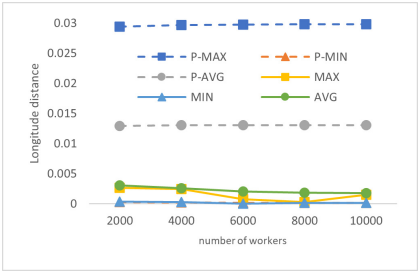

Fig. 10. Latitude travel distance under different tasks

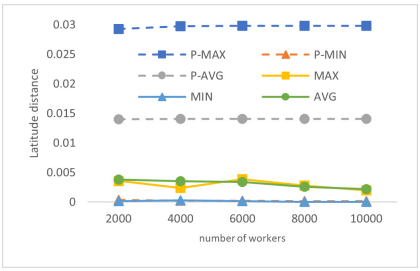

Fig. 11. Longitude travel distance under different tasks

In summary, our overall running time is acceptable and requires fewer exponentiation operations than PriRadar [21]. The maximum number of candidate workers set has well suppressed the increase in the amount of calculation. The calculation required locally by the worker is easy and the calculation error is negligible. The travel cost of our scheme is significantly less than [21], which indicates the superiority of our scheme.

6 Related Work

The privacy-preserving task allocation in spatial crowdsourcing has been an active area of research in recent years. Early schemes usually need a trusted third party [15], or a trusted data processing agency [6], but often cannot be realized in reality. At the same time, it may cause unnecessary delays in task allocation. When the staff wants to update their location, online TTP needs to publish new statistical location data, which may result in higher communication overhead [19]. And they usually only focus on protecting the location privacy of workers [12,15], and assume that the task location is public. However, the task location should also be protected, because workers who receive the task are often near the task location, and the requester often releases the task near its location [16].

The more mainstream solutions also have their advantages and disadvantages. In [10,20,22], k-anonymity and dummy users are leveraged to hide the location in a cloak region or a group of location data to achieve location privacy. But spatial anonymity cannot resist background knowledge attacks. Differential privacy protection and encryption technology can resist background knowledge attacks and have a top-level of privacy protection. [23] proposed a novel spatial crowdsourcing framework without using a trusted third party but providing a DP guarantee. Workers no longer submit their locations but choose their acceptable task set from public task location space, realizing the protection of workers' locations. But workers need to set up an acceptable location in advance, which requires a relatively sizeable storage space. The amount of noise is difficult to control, which can easily lead to reduced data availability. For example, [13] proposed an efficient protocol to securely compute the worker travel cost and

select minimum cost worker in the encrypted domain, which reveals nothing about location privacy. But encryption technology usually has a relatively large operating overhead. [21] devised a grid-based location protection method, which can protect the locations of workers and tasks while keeping the distance-aware information on the protected locations. And they leveraged both attribute-based encryption and symmetric-key encryption to establish secure channels through servers, which ensures that the task is delivered securely and accurately by any untrusted server. But the delivery process involves numerous encryption and decryption operations, which was not efficient enough.

7 Conclusion

Aiming at the privacy protection of task allocation in spatial crowdsourcing, we proposed a task allocation scheme based on encryption protection. We use a two-server model and adopting paillier homomorphic cryptosystem to protect the location privacy of workers and tasks. And our scheme doesn't need any online TTP to involve. We especially realize the delivery of tasks' locations, which allows workers to achieve effectively privacy protection with a tiny computational overhead locally. At the same time, its advantages are proved by experiments on real data. Our scheme achieves practical performance in terms of computational overhead and travel cost and the running time is acceptable. For further work, we are going to focus on how to reduce the computing time of encryption protection technology, and how to further combine different kinds of protecting the task to improve efficiency.

Acknowledgment. This work is supported by the National Key Research and Development Program of China (No. 2018YFB0204301), National Nature Science Foundation of China (No. 62072466, No. U1811462), and the NUDT Grants (No. ZK19-38).

References

1. Bugiel, S., Nurnberger, S., Sadeghi, A.R., Schneider, T.: Twin clouds: an architecture for secure cloud computing (2011)
2. Goldreich, O.: Foundations of cryptography. II: basic applications 2 (2004). https://doi.org/10.1017/CBO9780511721656
3. Goldwasser, S.: The knowledge complexity of interactive proof system. SIAM J. Comput. **18**(1), 186–208 (1989)
4. Haiping, H., Tianhe, G., Ping, C., Reza, M., Tao, C.: Secure two-party distance computation protocol based on privacy homomorphism and scalar product in wireless sensor networks. Tsinghua Sci. Technol. (2016)
5. Howe, J.: The rise of crowdsourcing. Wired **14**(6), 176–183 (2006)
6. Kazemi, L., Shahabi, C.: A privacy-aware framework for participatory sensing. ACM SIGKDD Explor. Newsl. **13**(1), 43 (2011)
7. Liu, B., Chen, L., Zhu, X., Zhang, Y., Zhang, C., Qiu, W.: Protecting location privacy in spatial crowdsourcing using encrypted data. In: EDBT, pp. 478–481. OpenProceedings.org (2017)

8. Liu, L., Chen, R., Liu, X., Su, J., Qiao, L.: Towards practical privacy-preserving decision tree training and evaluation in the cloud. IEEE Trans. Inf. Forensics Secur. **15**, 2914–2929 (2020)
9. Miao, C., Jiang, W., Su, L., Li, Y., Guo, S.: Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In: ACM Conference on Embedded Networked Sensor Systems (2015)
10. Niu, B., Li, Q., Zhu, X., Cao, G., Hui, L.: Achieving K-anonymity in privacy-aware location-based services. In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications (2014)
11. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on Advances in Cryptology-Eurocrypt (1999)
12. Pournajaf, L., Li, X., Sunderam, V., Goryczka, S.: Spatial task assignment for crowd sensing with cloaked locations. In: IEEE International Conference on Mobile Data Management (2014)
13. Shen, Y., Huang, L., Li, L., Lu, X.: Towards preserving worker location privacy in spatial crowdsourcing. In: IEEE Global Communications Conference (2015)
14. To, H., Ghinita, G., Fan, L., Shahabi, C.: Differentially private location protection for worker datasets in spatial crowdsourcing. IEEE Trans. Mob. Comput. **16**(4), 934–949 (2017)
15. To, H., Ghinita, G., Shahabi, C.: A framework for protecting worker location privacy in spatial crowdsourcing. Proc. VLDB Endow. **7**(10), 919–930 (2014)
16. To, H., Shahabi, C.: Location privacy in spatial crowdsourcing. In: Gkoulalas-Divanis, A., Bettini, C. (eds.) Handbook of Mobile Data Privacy, pp. 167–194. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98161-1_7
17. To, H., Shahabi, C., Kazemi, L.: A server-assigned spatial crowdsourcing framework. ACM Trans. Spat. Algorithms Syst. **1**(1), 1–28 (2015)
18. Wang, Z., Hu, J., Jing, Z., Yang, D., Chen, H., Qian, W.: Pay on-demand: dynamic incentive and task selection for location-dependent mobile crowdsensing systems. In: IEEE International Conference on Distributed Computing Systems (2018)
19. Xiao, Y., Xiong, L.: Protecting locations with differential privacy under temporal correlations, pp. 1298–1309 (2015). https://doi.org/10.1145/2810103.2813640
20. Yang, D., Xi, F., Xue, G.: Truthful incentive mechanisms for K-anonymity location privacy. In: Infocom. IEEE (2013)
21. Yuan, D., Li, Q., Lia, G., Wang, Q., Ren, K.: PriRadar: a privacy-preserving framework for spatial crowdsourcing. IEEE Trans. Inf. Forensics Secur., 1 (2019)
22. Zhai, D., et al.: Towards secure and truthful task assignment in spatial crowdsourcing. World Wide Web **22**(5), 2017–2040 (2018). https://doi.org/10.1007/s11280-018-0638-2
23. Zhang, L., Xiong, P., Ren, W., Zhu, T.: A differentially private method for crowdsourcing data submission. Concurrency Comput. Pract. Exp. 31, **e5100** (2018)