PriTAEC: Privacy-Preserving Task Assignment Based on Oblivious Transfer and Edge Computing in VANET

Zihui Xu, Lei Wu*, Chengyi Qin, Su Li, Songnian Zhang, and Rongxing Lu, *Fellow, IEEE*

***Abstract*—Spatial crowdsourcing, e.g., Vehicular Ad-Hoc Network (VANET)-based spatial crowdsourcing, is a new distributed computing paradigm, in which task assignments highly rely on the wisdom of the crowdsourcing platform. However, with the increase in user data leakage incidents, the existing task assignment methods are no longer sufficient to meet the privacy requirements of users. In the existing VANET-based spatial crowdsourcing, task assignments are usually performed by a trusted third party based on the real locations of tasks and drivers (task performers), which may lead to the leakage of the users' locations. Furthermore, the drivers usually prefer to query the nearest tasks to them in a geometric range, at this point sending query requests to remote crowdsourcing servers increases unnecessary response delays. To assign tasks securely and efficiently, we propose a privacy-preserving task assignment scheme based on OT and edge computing (PriTAEC), which is the first to apply Oblivious Transfer (OT) and edge computing to preserve the location privacy of VANET-based spatial crowdsourcing. In the scheme, we first utilize Hilbert Curve and Bloom Filter to implement location range queries. Then, we use geohash location encoding and Oblivious Transfer to achieve fine-grained location matching. In particular, we design a task assignment algorithm with an offline-online phase to improve the efficiency of task assignments. Finally, we prove the security of the scheme and evaluate its performance, which shows our scheme is secure and efficient.**

***Index Terms*—Location privacy, edge computing, task assignment, privacy-preserving, range query, oblivious transfer.**

I. Introduction

Spatial crowdsourcing [1] is a spatio-temporal based distributed computing paradigm, in which task assignments highly rely on the intelligence of the crowdsourcing platform. Currently, there are many platforms that support spatial crowdsourcing services, such as MTurk, Upwork and CrowdFlower. In VANET, spatial crowdsourcing can be applied to many data collection and integration scenarios, e.g., traffic flow monitoring, road condition monitoring, and online car-hailing [2], [3].

Lei Wu is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250307, China, with Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China, and also with Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, Jinan 250307, China (E-mail: wulei@sdnu.edu.cn).

Zihui Xu, Chengyi Qin, and Su Li are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250307, China (E-mail: xuzihui994@163.com; QCY521111@163.com; 13864106238@163.com).

Songnian Zhang and Rongxing Lu are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: szhang17@unb.ca; rlu1@unb.ca).

* Corresponding author.

In VANET, spatial crowdsourcing is characterized by drivers who need to perform tasks at a specific time and location. In particular, drivers usually prefer to query the tasks nearest to them to reduce travel overhead, while requesters (task publishers) usually need drivers who meet the task attributes to improve the quality of task results. Therefore, in the existing VANET-based spatial crowdsourcing, the crowdsourcing service provider (CSP) performs task assignments based on the plaintext information about the locations and attributes of the drivers and tasks. Though the approach can accurately assign tasks, it may lead to the leakage of sensitive information of users, e.g., locations and interests. This is because, in the real world, the CSP is not completely trustworthy, and it may intentionally disclose sensitive user information to other third parties for personal benefit, which can be used to infer a person's interests or daily trajectory. Furthermore, in the existing VANET-based spatial crowdsourcing, the drivers and requesters send requests to the remote CSPs, which increases unnecessary communication delays. Therefore, a secure and efficient task assignment scheme is needed for the VANET-based crowdsourcing platforms.

In this paper, we mainly focus on the study of location privacy issues. Hu and Yuan et al. [6], [7] divided geographic space into grids to implement range queries. In [6], each grid is labeled and users encrypt their locations using inner product encryption. The CSP utilizes ciphertexts to calculate the inner product to judge the location relationship, in which original data is not disclosed to the CSP. The scheme of Yuan et al. [7] requires users to submit HMAC codes of their grid locations and then implements location matching by whether the location codes are the same, which reduces the computing cost of the CSP, but sacrifices the accuracy of location matching. Han et al. [8] proposed spatial crowdsourcing privacy-preserving location distance computation, which uses bilinear pairs and Chebyshev chaotic mapping to accurately compute the location distance. However, the CSP needs to calculate the distance between each task and driver, which is computationally expensive.

In this paper, we propose a privacy-preserving task assignment scheme (PriTAEC) based on OT and edge computing, which can assign tasks securely and efficiently. In PriTAEC, we divide the task assignment into two phases. In the first phase, we utilize Hilbert Curve [9] to map two-dimensional location coordinates to one-dimensional Hilbert curve values and then use Bloom Filter [10] to implement the grid range query, in which the edge nodes can find out tasks within the query range of the drivers. In the second phase, we utilize Geohash location encoding [11] and Oblivious Transfer [12] to implement the fine-grained location matching, in which the edge nodes can assign the nearest tasks to the drivers. To help the requesters to find the drivers who satisfy the task attributes, we use the attribute-based encryption scheme [13] to ensure that only the drivers who satisfy the task attributes can decrypt the content of the tasks. In particular, to improve the efficiency of task assignments, we design a task assignment algorithm with an offline-online phase in which edge nodes can receive tasks for moving drivers, and then drivers can assist edge nodes to implement task assignments with simple online computing. The offline-online algorithm mainly plays the role of pre-calculation. It not only saves clients' computation and storage costs but also improves the efficiency of computation, which is very friendly to the mobile side with limited computation and storage. Moreover, when users are busy and want to initiate the query in advance, the offline-online algorithm can meet the users' pre-calculation requirements, which also improves the users' time utilization. In particular, the offline-online algorithm can also be convenient for clients when their signal is weak.

The main contributions of our work can be summarized as follows:

1) We propose a privacy-preserving task assignment scheme based on OT and edge computing, which utilizes edge computing to reduce response latency, uses Hilbert curves and Bloom filters to achieve range queries and uses Oblivious transfers and Geohash to achieve fine-grained location matching.

2) We use a hybrid encryption (attribute-based encryption and symmetric encryption) to preserve the confidentiality of the task content, which only allows the drivers who satisfy the task attributes to decrypt the task.

3) We design a task assignment algorithm with an offline-online phase to improve the efficiency of the task assignment, in which the edge node can receive tasks for the drivers in the offline phase, and then the drivers who want to query tasks only need to perform simple calculations in the online phase to assist the edge node to implement task assignments.

4) We analyze the security of the PriTAEC scheme and evaluate its performance, which shows that the PriTAEC scheme can achieve task assignments securely and efficiently.

This paper is organized as follows. In Section II we discuss the related work on the topics covered in this paper. In Section III we provide the necessary background knowledge of the techniques used in this paper. In Section IV we formulate the issues related to the PriTAEC scheme, including the system model, threat model, and design goals. In Section V we describe the detailed construction of the PriTAEC scheme. In Sections VI and VII we provide security analysis and performance evaluation, respectively. In Section VIII we describe the conclusions we have drawn.

II. RELATED WORK

Our research is related to three topics: VANET-based spatial crowdsourcing task assignments, Location privacy preservation, and Edge computing. Those topics are discussed separately as follows.

A. VANET-based Spatial Crowdsourcing Task Assignments

VANET-based spatial crowdsourcing is dedicated to assigning tasks to appropriate workers (drivers) based on their spatio-temporal properties [14]. The existing task assignment problem of VANET-based spatial crowdsourcing is generally divided into two scenarios to study [1]: dynamic scenario (offline scenario) and static scenario (online scenario). In the static scenario, the problem that the crowdsourcing server has to solve is how to assign tasks so that the workers have minimal travel overhead and the requesters receive high quality results, given that the spatio-temporal information of the tasks and workers is known in advance. In the dynamic scenario, the problem that the crowdsourcing server has to solve is how to respond in real time to the task query requests sent by workers. Obviously, in the dynamic scenario, the crowdsourcing server is aware of the worker's requirements only after the worker submits relevant information, which is more in line with real world, but it needs to face more challenges such as timeliness, privacy, etc.

There are two existing models of VANET-based spatial crowdsourcing task assignments [1]: Worker Request Model (WRM) and Server Assignment Model (SAM). In WRM, workers can choose the appropriate tasks according to their preferences. In SAM, workers need to upload thier spatio-temporal information to the crowdsourcing platform, and then the crowdsourcing server assigns tasks to the workers.

B. Location Privacy Preservation

The main privacy problem studied in this paper is location privacy. K-anonymity [15]–[17] enables to hide the real location in a set of location data or cloaked area so that the success probability of an attacker to infer the user's location is $\frac{1}{K}$. The pseudonym technique obscures the real identity of users and it can achieve a certain degree of privacy preservation. Li et al. [19] proposed a location filtering protocol using K-anonymity and cloaked area which can securely fulfill location-based service queries under three-party interaction, but the setting of the cloaked area may lead to lower query accuracy. Lien et al. [20] proposed circular shift queries using Hilbert curve and homomorphic encryption, which preserves the privacy of location and supports private index retrieval, but it is not applicable to nearest neighbor queries. Kong et al. [37] proposed location privacy preserving range query in vehicular sensing, which can retrieve vehicle sensing data without revealing location privacy. Wang and Zheng et al. [22]–[24] proposed the geometric range query using inner product encryption, which can effectively preserve the location while judging the position relationship between location points and geometric ranges. Shu et al. [38] proposed a proxy-free task matching scheme, which implements task matching by searchable encryption and enables user revocation with minimal system overhead. Zhang et al. [25] proposed a blockchain-based crowdsourcing scheme, which effectively realizes fine-grained privacy level for different users. This may increase the computing burden on workers as they have to determine the access rights for each task.

C. Edge Computing

Edge computing can collect and analyze sensory data from edge devices [26]. Compared to cloud computing, edge computing can improve communication and computation latency. Edge computing has been widely used in Internet of Vehicles (IoV) for spatial crowdsourcing, which allows drivers to send query requests to the nearest edge node. Basudan et al. [2] proposed crowdsourcing-based road condition monitoring, which can collect and integrate drivers' road data. Bonomi et al. [27] proposed hierarchical distribution architecture and defined the role of edge computing in the Internet of Things (IoT). Liu et al. [28] and Li et al. [29] proposed secure edge computing for EV charging, where RSUs act as edge nodes to match charging piles and EVs, and the location and state of vehicles are not revealed to edge nodes. Cui et al. [30] proposed IoV computing platform which provides low latency computing services to implement resource management and task loading. Li et al. [31] proposed carpooling edge computing for IoV, which implements user's pick-up and drop-off location matching and anonymous verification.

III. PRELIMINARIES

In this section, we provide the fundamentals of the techniques used in the PriTAEC scheme, which include Hilbert curve, Geohash, Bloom filter, Oblivious transfer, Private equality test and Ciphertext-policy attribute-based encryption.

A. Hilbert Curve and Geohash

Hilbert curves are continuous fractal space filling curves which were first introduced by Hilbert [9]. A Hilbert curve in a multidimensional space traverses every point in the space once and only once in a particular order. Specifically, the order-n Hilbert curve divides the region into $2^n * 2^n$ subregions, and Fig.1 and Fig.2 depict the order-1 and order-2 Hilbert curves, respectively, where each grid has a Hilbert value. The important reason for using the Hilbert curve is that it is a very suitable tool [21] for our proposed scheme. First, the Hilbert curve can map two-dimensional location coordinates to one-dimensional Hilbert values using a spatial transformation function. Second, the Hilbert curve can traverse every subgrid of the grid space and a Hilbert value can represent a rectangular range. Third, given the curve parameters, users can easily construct Hilbert curves and determine the range of the query.

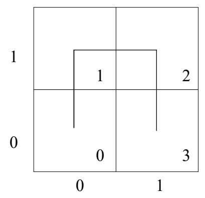

Fig. 1: order-1 curve

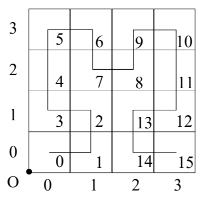

Fig. 2: order-2 curve

Geohash is a hierarchical, grid-based index in which locations can be mapped to strings [11]. A geohash string (or encoding) is obtained from the latitude and longitude pairs of cross bits. Geohash code has two properties, 1) the longer the geohash code the more accurate the position is, and 2) If two geohash codes with the same prefix are longer then they are closer together.

B. Bloom Filter

Bloom filter is proposed by Bloom [10] to solve the approximate set membership problem, which can quickly determine whether an element is an element of a set. Bloom filter is subject to false positives, which can be improved by setting the relevant parameters [32].

Specifically, Bloom Filter maps each element of a set to a binary string by a number of hash functions and the detected element is mapped to a binary string by a number of the same hash functions. If the position of the element binary string is 1, the set binary string is also 1, then the element is in this set.

C. Oblivious Transfer and Private Equality Test

Oblivious Transfer (OT) is an important cryptographic primitive in secure multi-party computing, which was introduced by Rabin [12]. In OT, the sender has a pair of input strings (x_0, x_1) interacts with the receiver who has an input selection bit $b \in \{0, 1\}$. The sender sends a message pair (x_0, x_1) to the receiver who only learns x_b but not any information about x_{1-b}. This protocol is known as 1-out-of-2 OT.

Private Equality Test (PEQT) is a two-party protocol introduced by Beaber et al. [33] in which the sender has the input string x_0 interacts with the receiver has the input string x_1. If $x_0 = x_1$, the receiver receives bit 1, otherwise it receives bit 0. In this process, the sender does not get any information about x_1.

D. Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

CP-ABE [13] is an encryption algorithm that implements data access control, the private key of ABE is associated with a set of attributes, and the ciphertext is encrypted by a specific access structure. Only if the key-related attributes satisfy the access structure, the ciphertext can be decrypted, usually CP-ABE consists of four basic algorithms: $ABE.Setup$, $ABE.Enc$, $ABE.KeyGen$ and $ABE.Dec$, as follows.

•$\boldsymbol{ABE.Setup(1^{\aleph}) \rightarrow (PK, MK).}$ The data owner invokes the initialization algorithm to generate the necessary system parameters, which takes a security parameter $\aleph$ as input and outputs a public key PK and master key MK.

•$\boldsymbol{ABE.Enc(PK, M, L) \rightarrow c_t.}$ The data owner invokes the encryption algorithm to encrypt a message M, which takes the public key PK, message M and a access structure L as input and outputs the ciphertext c_t.

•$\boldsymbol{ABE.KeyGen(MK, P) \rightarrow sk_L.}$ The data user invokes the key generation algorithm to generate the attribute key, which takes the master key MK and a set of attributes P as input and outputs the decryption key sk_L.

•$\boldsymbol{ABE.Dec(PK, c_t, sk_L) \rightarrow M.}$ The data user invokes the decryption algorithm to decrypt the ciphertext c_t, which takes the public key PK, ciphertext c_t and key sk_L as input and outputs the message M.

IV. PROBLEM FORMULATION

In this section, we describe the system model, threat model, design goals and main ideas of the PriTAEC scheme. We summarize the notations used by the PriTAEC scheme in Table 1.

TABLE I: PriTAEC Notations

Notation	Definition
(x, y)	Location coordinates
$< x, y >$	Grid coordinates
$\bar{h}$	Spatial transformation funtion
h_i	Hash functions in Bloom filters
m	Length of Bloom Filter
σ_{sign}	Signature of edge nodes
t	Timestamp of edge nodes
id	Identifier of the user or task
pub	Public parameters
BF_R, BF_D	Binary arrays of requesters, drivers
P_i	Ciphertext policies
C_K, C_M	Ciphertext of the key, task
$M_{n\times n}$	Permutation matrix
$b_{index}, \tilde{b}_{index}$	Index vectors, transpose index vectors
P, X	Location code of the driver, requester

A. System Model

There are four entities involved in the system model: drivers, requesters, edge nodes, and the crowdsourcing platform. In Fig.3, a general spatial crowdsourcing model is depicted, and in Fig.4, an offline-online spatial crowdsourcing model is depicted. Both models are dedicated to assigning the nearest task to the driver. The role of each entity is defined as follows:

- Requesters can release tasks on edge nodes.
- Drivers can query tasks and accept them from edge nodes, and submit the results to the requester.
- The edge nodes are responsible for assigning tasks and managing the local Hilbert curve parameters, in this paper roadside units (RSU) or base stations can act as edge nodes.
- The crowdsourcing platform is responsible for user registration and issuing signature keys and parameters of attribute-based encryption.

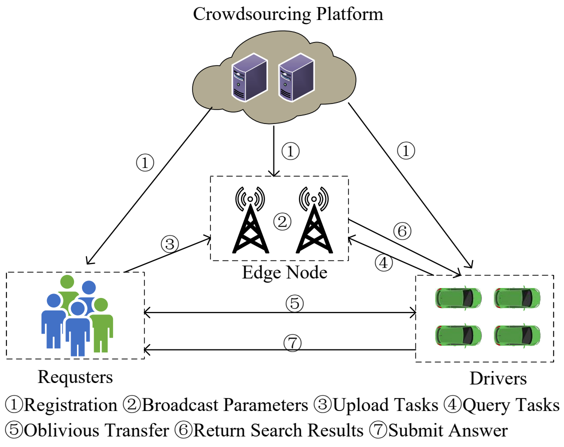
Crowdsourcing Platform
①
①
①
②
③
④
⑤
⑥
⑦
Edge Node
Requsters
Drivers
①Registration ②Broadcast Parameters ③Upload Tasks ④Query Tasks
⑤Oblivious Transfer ⑥Return Search Results ⑦Submit Answer

Fig. 3: General Spatial Crowdsourcing Model

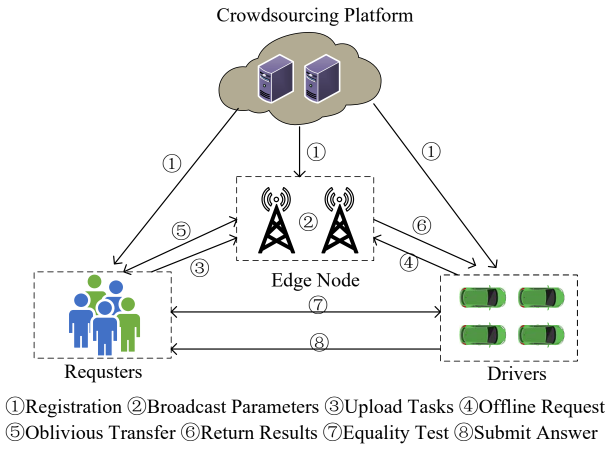
Crowdsourcing Platform
①
①
①
②
③
④
⑤
⑥
⑦
⑧
Edge Node
Requsters
Drivers
①Registration ②Broadcast Parameters ③Upload Tasks ④Offline Request
⑤Oblivious Transfer ⑥Return Results ⑦Equality Test ⑧Submit Answer

Fig. 4: Offline-Online Spatial Crowdsourcing Model

B. Threat Model

Each entity is defined in the threat model as follows:

Users (requesters and drivers) and edge nodes are considered as honest-but-curious entities (semi-honest) [34], which means that they follow protocols but desire to infer sensitive information about other entities.

The crowdsourcing platform is considered to be an honest entity, which is honestly responsible for user registration and issuing signature keys and parameters of attribute-based encryption.

Note that in this paper, we also assume no collusion between users and the edge nodes [35]. Since spatial crowdsourcing is a practical application model, the crowdsourcing platform would damage its reputation and interests by revealing the sensitive information of the users, so this assumption is reasonable in real world.

C. Design Goals

The PriTAEC scheme supports multi-driver and multi-task scenarios, which will meet the following utility and security goals.

Utility Goals. The PriTAEC scheme meets the following utility goals:

- **Grid Range Query.** The driver can send a task query request to the edge node to obtain the tasks within the grid range.
- **Nearest Distance Task Assignment.** The driver can send a task query request to the edge node to obtain the nearest task.
- **Data Access Control.** The requester can set task attributes, such as locations or interests, and only drivers who satisfy the task attributes are allowed to access the task.
- **Offline-Online Task Assignment.** When the driver does not know his/her final location during the moving process, the edge node can receive tasks within a certain time period instead of the driver. Once the driver has determined the location, he/she can assist the edge node to assign tasks through simple online operations.

Security Goals. The PriTAEC scheme meets the following security goals:

• **Location Privacy.** Users (requesters and drivers) are not willing to disclose their location to edge nodes, so users do not upload their real location information in plaintext but in ciphertext or generalized form.

• **Confidentiality of Task Content.** The requesters do not want to reveal the specific task content to the edge nodes, so the requesters utilize cryptography to preserve the task content.

D. Main Ideas

The PriTAEC scheme divides the task assignment process into two phases. In the first phase, the edge node implements the range query for the driver. In the second phase, the edge node assigns the nearest task to the driver.

First Stage. The edge node divides the local spatial region into grids and generates a Hilbert curve in the grid. The driver and requester can map their real location to the Hilbert curves to obtain the Hilbert values. When a requester releases a task, the requester needs to process the Hilbert value by Bloom filter and then upload the filtered value to the edge node. When a driver queries a task, the driver selects the Hilbert values of the neighboring grids according to his/her location in the grid, then filter these Hilbert values and upload the filtered values to the edge node. After receiving the location filter values from the driver and the requester, the edge node utilizes the properties of Bloom filter to determine whether the task is within the query range of the driver.

Second Stage. To get the nearest task, the requester and driver use the characteristics of geohash to sense the location distance. To preserve location privacy, the requester constructs the geohash prefix encoding family and perturbs the order of prefix encoding in the prefix encoding family. The requester and driver then perceive the length of the same prefix encoding in thier geohash encoding by performing the oblivious transfer and private equality test protocols. After the above protocols, the driver uploads the matching index to the edge node. Then, the edge node can recover the encoding order and sense the distance between the driver and the requester, and finally the edge node can find the nearest task and assign it to the driver.

V. DETAILED CONSTRUCTION OF PRITAEC

To clearly express the details of the scheme PriTAEC, this section describes the crowdsourcing process between a driver and multiple requesters. Specifically, we describe the scheme PriTAEC in seven phases, which includes: (1) system initialization; (2) user registration; (3) task release; (4) task query; (5) range query; (6) finding the nearest task; (7) decrypting the task.

A. System Initialization

During the system initialization phase, the edge node generates the necessary system parameters to implement task assignments, as shown in the following steps:

Step 1 (Constructing a Hillbert curve): The edge node divides the geographical area under its jurisdiction into a grid G and constructs a Hilbert curve within it, which is used to convert the two-dimensional location coordinates into one-dimensional Hilbert curve values.

As an example, Fig. 2 depicts the grid G, which contains the grid coordinates and the Hilbert values for each subgrid. In particular, users can convert thier location coordinates to Hilbert values using equation $H(s) = \bar{h}(\langle x, y\rangle)$, where $\bar{h}$ is the spatial transformation function and $\langle x, y\rangle$ is the grid coordinate. More specifically, the users first convert the location coordinates to grid coordinates and then to Hilbert values, which can be referenced in reference [21].

The edge node discloses the construction parameter $STP = \{(X_l, Y_l), N, \sigma, \Theta\}$ of the Hilbert curve to the authorized users, including the curve start point (X_l, Y_l), the curve order N, the curve direction σ, and the curve scale factor Θ. Note that we assume that the authorized users are all located under the same edge node.

Step 2 (Broadcasting parameters): The edge node broadcasts the public parameter $pub = \{STP, G, t\}$ and the signature σ_{sign} for pub, where t denotes the current timestamp of the edge node.

B. User Registration

During the user registration phase, the crowdsourcing platform issues the relevant secret keys for the authenticated users, as shown in the following steps:

Step 1 (Generating attribute keys): Before user registration, the crowdsourcing platform calls the attribute-based encryption algorithm $ABE.setup(1^{\aleph})$ to generate (PK, MSK), where PK is the public key and MSK is the master key.

Step 2 (Constructing a Bloom filter): The crowdsourcing platform constructs a Bloom filter of length m. Specifically, it selects k mutually independent hash functions $h_i : \{0,1\}^* \rightarrow \{1, ..., m\}$, where $i \in (1, ..., k)$.

Step 3 (User registration): When a user registers, he/she selects a role according to his/her needs, such as a driver or requester. In this process, the user needs to provide the necessary identity authentication information to the crowdsourcing platform. After successful verification by the crowdsourcing platform, the user will receive the hash functions $h_1, h_2, ..., h_k$ and an identifier id from the crowdsourcing platform. In particular, if the user is registered as a driver, the crowdsourcing platform forwards the MSK to the user and invokes the AES key generation algorithm $AES.KeyGen(1^{\aleph})$ to generate a symmetric key K for the driver, otherwise the crowdsourcing platform forwards the PK to the user.

Step 4 (Issuing signature keys): The crowdsourcing platform also issues signature keys and certificates to the edge nodes and registered users, respectively, which ensures data integrity and signature forgery during the communication between entities.

C. Task Release

During the task release phase, requesters encrypt locations and task contents and upload the ciphertexts to the edge node. Taking one requester as an example, the steps are as follows:

Step 1 (Verifying messages): After the requester receives the public parameter pub and the signature σ_{sign} broadcasted by the edge node, the requester verifies the integrity of the parameter pub by signature σ_{sign}, and then verifies the timeliness of the parameter pub by timestamp t. If both verification processes are passed, the requester can publish tasks with the public parameter pub.

Step 2 (Filtering Hilbert values): Specifically, the requester maps the location coordinates (x_r, y_r) into the Hilbert curve and obtains the Hilbert curve value $H-index$. Then, the requester initializes a binary string BF_R of length m and uses $h_1, ..., h_k$ to calculate the filter values i on the $H-index$, respectively, and sets the corresponding position in BF_R to 1 according to the filter value i. This process is equivalent to encrypting the location grid, where $i \in \{1, 2, ..., m\}$.

Step 3 (Encrypting the task content): Then, the requester encrypts the task content M. The requester sets the ciphertext policy $L = [L_1, ..., L_n]$, then invokes the AES encryption algorithm $AES.Enc(K, M)$ and attribute-based encryption algorithm $ABE.Enc(PK, K, L)$ to output C_M and C_K, respectively.

Step 4 (Constructing a transposition matrix): The requester also generates a random transposition matrix $M_{n\times n}$ to perturb the order of geohash code to preserve location privacy, noting that the $M_{n\times n}$ used in each task release is different and $n > l + 1$, where l is the length of the geohash code.

Step 5 (Submitting task requests): Finally, the requester uploads the task request $Req = \{id_R, id_{task}, C_K, C_M, M_{n\times n}, BF_R, t_R\}$ and σ_{Req} to the edge node, where id_R is the requester identity, id_{task} is the task identity, t_R is the current timestamp of the requester, and σ_{Req} is the signature of the Req.

D. Task Query

During the task query phase, the driver encrypts the location and query range and uploads the ciphertext to the edge node, as shown in the following steps:

Step 1 (Verifying messages): Similarly, after the driver receives the public parameter pub and signature σ_{sign}, the driver verifies the integrity of the parameter pub by signature σ_{sign} and then verifies the timeliness of the parameter pub by timestamp t. If both verification processes pass, the driver can use parameter pub to query tasks.

Step 2 (Filtering Hilbert values): Specifically, the driver maps the location coordinates (x_d, y_d) into the Hilbert curve, obtains the Hilbert value $H-index$, and then selects several neighboring $H-indexes$, which are used as the driver's grid query range. Note that since the driver knows the grid G and the Hilbert curve, the driver can locate his location in the grid after converting his real location coordinates to Hilbert values. Then the Hilbert values of the neighboring grids can be selected as the grid range according to his/her needs.

Next, the driver initializes a binary string BF_D of length m and then uses $h_1, ..., h_k$ to compute the filter values i for each of these $H-indexes$ and sets the value of the corresponding position in BF_D to 1 according to the filter value i. This process is equivalent to encrypting the location range grid.

Step 3 (Submitting task query requests): Finally, the driver sends a query request $Que = \{id_D, BF_D, t_D\}$ and a signature σ_{Que} to the edge node, where id_D is the driver's identity, t_D is the driver's current timestamp, and σ_{Que} is the signature of the Que.

E. Range Query

In the range query phase, the edge node implements range queries for the driver, as shown in the following steps:

Step 1 (Verifying messages): In this phase, the edge node receives Req, σ_{Req}, Que, and σ_{Que}. Similarly, the edge node first verifies the integrity and timeliness of Req and Que.

Step 2 (Matching tasks): If both verification processes pass, the edge node will compare BF_R and BF_D. For $1 \le i \le m$, if $BF_R[i]$ is 1, $BF_D[i]$ is also 1, then the task is within the driver's grid query range.

Step 3 (Returning matching results): Finally, the edge node forwards the matching result $match = \{id_R, id_{task}, C_K, C_M, t_{edge}\}$ and signature σ_{match} to the driver, where t_{edge} is the current timestamp of the edge node and σ_{match} is the signature of the $match$. Note that the scenario we describe here is a multi-requester and single-worker scenario, so the matching result $match$ obtained by the edge node is not just one.

F. Finding the Nearest Task

When the driver receives the query results, the driver will query the nearest task from the results.

Step 1 (Verifying messages): The driver receives the message $match$ and verifies the integrity and timeliness of the message $match$, if the verification is passed, the driver will search for the nearest task from the matched results.

Step 2 (Generating geohash codes): Specifically, the driver calculates the location code $geohash(x, y)$ to obtain a geohash bit string of length l. The characteristic of geohash is that, given two geohash bit strings, if their same prefix is longer, then they are located closer to each other. Next, we call the geohash bit string as the location code.

To enable the driver to perceive the distance from the task without the location code being compromised, the PriTAEC scheme uses oblivious transfer protocol and private equality test protocol to solve the above problem.

In the plaintext state, the driver and requester hold their own location codes, and the requester computes the prefix family based on the location codes it holds. For example, if the requester's location code is $b_1b_2...b_l$, the prefix family is $b = \{b_1b_2...b_{l-1}b_l, b_1b_2...b_{l-1}*, ..., b_1 * ... **, ** ...*\}$, where $*$ is a wildcard, and $*$ can match either 0 or 1. Let the driver's location code be $a_1a_2...a_l$, and the driver will match $a_1a_2...a_l$ with every string of length l in the prefix family, and the worst case is $l + 1$ times. If the index of the first successful match is smaller, then their same prefix is longer, so the closer their locations are.

Step 3 (Geohash families transposition): To prevent the location code of the requester from being leaked to the driver, the requester builds $l + 1$ indexes for $l + 1$ strings in the order of b. For example, the index values of $b_1b_2...b_{l-1}b_l$ and $b_1b_2...b_{l-1}*$ are 1 and 2, respectively. The requester adds $n - l - 1$ random codes of length l to the prefix family and constructs the initial index vector $b_{index} = \{1, 2, ..., l +$

$1, ..., n-1, n\}$, where n is the security parameter and is greater than $l+1$. Then the requester calculates the transpose index vector $\tilde{b}_{index} = b_{index} \cdot M_{n\times n}$, where $M_{n\times n}$ is a random permutation matrix whose elements are 0 or 1, which is used to change the order of the elements in the vector b_{index}. Then, the requester reorders the strings in the prefix family according to the transpose index vector $\tilde{b}_{index}$. Finally, the requester uses the reordered prefix family to execute the oblivious transfer protocol with the driver. Note that the original prefix family of all requesters is set in the form of b, so the edge nodes also know the initial index vector b_{index}.

Step 4 (Oblivious transfer): In the scheme PriTAEC, oblivious transfer protocol is used to secretly perform wildcard string matching. Specifically, let the bit strings of the driver and the requester be X and P, respectively, and other expressions of P are shown in Fig.5.

P	$\tilde{P}$	$\bar{P}$
$*$	1	0
1	0	1
0	0	0

Fig. 5: Location code

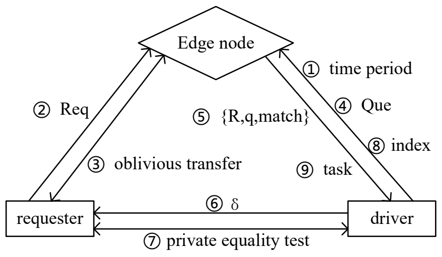

Fig. 6: Offline-online model parameter delivery

If $\bar{P} = \tilde{P} \cdot X \oplus X$, then the bit X matches the bit P. When $\tilde{P} = 1$, the equation holds, and when $\tilde{P} = 0$, if the equation holds then $\bar{P} = X$, we can see from the table that at this time $\bar{P} = P$, so $P = X$. To perform the string matching operation while protecting the location data from being leaked, the above equation is modified to $k \oplus \bar{P} = k \oplus \tilde{P} \cdot X \oplus X$. The matching process of the bit string is shown in **Algorithm 1**. After the execution of this algorithm, the driver gets the matching index.

Correctness: After the execution of OT protocol between the requester and the driver, the driver receives the output q_i. The true value of q_i is shown below:

$$q_i = \begin{cases} k_i & X_i = 0 \\ k_i \oplus \tilde{P}_i \tilde{P}_{i+l} ... \tilde{P}_{i+(n-1)l} & X_i = 1 \end{cases}$$

From the above equation, we can obtain the general form of q_i as $q_i = k_i \oplus (\tilde{P}_i \tilde{P}_{i+l} ... \tilde{P}_{i+(n-1)l}) \cdot X_i$, so $u_i = k_i \oplus (\tilde{P}_i \tilde{P}_{i+l} ... \tilde{P}_{i+(n-1)l}) \cdot X_i \oplus C(X_i)$. Let $\bar{K}$ be the matrix form of the key k_i, that is, $\bar{K} = (k_1, k_2, ..., k_l)^T$, so $u^i = \bar{K}^i \oplus \tilde{P}_{[i,i+l-1]} \cdot X \oplus X$, where $\tilde{P}_{[i,i+l-1]}$ is the i-th bit string of length l of the prefix family, and $t^i = \bar{K}^i \oplus \bar{P}_{[i,i+l-1]}$. If $P = X$, then $u^i = t^i$.

Step 5 (Submitting matching indexs): After the driver and the requester perform the **Algorithm 1**, the driver can get multiple indexes j of successful matches. The driver uploads these indexes and the corresponding requester identity id to the edge node.

Step 6 (Returning the nearest task): Since the edge node holds the transposition matrix $M_{n\times n}$, the index vector b_{index}, and the matching index j, the edge node can use index j to find the corresponding original index in b_{index}, and then the edge node records the smallest original index in b_{index}. Finally, the edge node compares the smallest original index values of multiple tasks to find the nearest task to the driver and forwards the task identifier to the driver.

```
Algorithm 1 Geohash Encoding Matching Protocol
Parameters:
  1. Two parties: the requester and the driver.
  2. The length of the requester's bit string is $nl$, the length of the driver's bit string is $l$, and the number of matches is defined as $n$.
  3. A repeated encoding function $C : \{0,1\} \rightarrow \{0,1\}^n$, for example, $C(a) = aa..aa$, the length of $C(a)$ is $n$.
Requester input: geohash bit string $P \in \{0,1,*\}^{nl}$.
Driver input: geohash bit string $X \in \{0,1\}^l$.
Protocol:
  1. [Random Keys] The requester randomly selects $\{k_i\} \leftarrow \{0,1\}^n$, where $1 \le i \le l$.
  2. [OT] For each $i$, the driver and the requester execute the OT protocol.
      a. The driver inputs a bit $X_i$
      b. The requester inputs a pair of strings $\{k_i, k_i \oplus \tilde{P}_i \tilde{P}_{i+l} ... \tilde{P}_{i+(n-1)l}\}$ of length $n$, where $\tilde{P}_i$ denotes the $i$-th bit character
      c. The driver receives the output $q_i$
  3. [Matrix Form]
      a. The requester constructs the matrix $T_{l\times n}$, where the $i$-th row is the vector $t_i = k_i \oplus \bar{P}_i \bar{P}_{i+l} ... \bar{P}_{i+(n-1)l}$
      b. The driver constructs the matrix $U_{l\times n}$, where the $i$-th row is the vector $u_i = q_i \oplus C(X_i)$
  4. [Private Equality Test]
      a. For each $j$, where $1 \le j \le n$. the driver and the requester invoke the private equality test function
      b. The requester as the sender inputs each column $t^j$ of the matrix $T_{l\times n}$
      c. The driver as the receiver inputs each column $u^j$ of the matrix $U_{l\times n}$
      d. If $t^j = u^j$, the driver receives output 1
```

For example, let the location code of the requester be 101, its transposition index vector $\tilde{b}_{index} = \{1, 4, 3, 2\}$, the transposition prefix family be $\{101, ***, 1**, 10*\}$, and the location code of the driver be 100, so after the oblivious transfer protocol, the driver gets the indexs j as 2, 3, 4. Then the driver uploads the indexs j to the edge node, and the edge node calculates the original indexes as 4, 3, 2 in order. The edge node saves the smallest index 2, and finally the edge node compares the smallest indexes of multiple tasks and forwards the task identifier id of the task with the smallest index to the driver. Note that here there may be the case of multiple identical smallest indexes, and we randomly choose one as the nearest task, at which point our scheme is an approximate nearest task assignment.

G. Decrypting Task

When the driver receives the nearest task, he or she will decrypt the task and submit the result.

Step 1 (Generating the decryption key): After the driver receives the nearest task, the driver invokes the key generation algorithm $ABE.KeyGen(MK, P)$ to generate the decryption key sk_P.

Step 2 (Task decryption): If P satisfies L, the driver invokes the decryption algorithm $ABE.Dec(C_K, sk_P)$ to get the symmetric key K, and finally the driver invokes the symmetric decryption algorithm $AES.Dec(K, C_M)$ to get the task content.

Step 3 (Submitting task results): When the driver finishes the task and gets the task result , the driver invokes the symmetric encryption algorithm $AES.Enc(K, result)$ to encrypt the task result and finally submits the ciphertext result to the requester.

H. Optimization

In order to improve the efficiency of crowdsourcing, our scheme supports offline operations. During this process, edge nodes can perform complex OT operations for drivers without revealing the privacy of drivers and requesters. We focus on the case where drivers do not know their final locations during the move and want to collect tasks in advance. When the driver stops moving and gets the exact location, the driver can find the nearest task through simple online calculations, as shown in the following steps. Fig.6 depicts the parameter interactions between entities in phase H.

Step 1 (Constructing the time period): Specifically, the driver selects a certain time interval (t_s, t_e) and forwards it to the edge node. Then, the edge node will receive the tasks for the driver in time period (t_s, t_e).

Step 2 (Verifying messages): After receiving the requester's task request Req, the edge node verifies the integrity and timeliness of the message Req.

Step 3 (Oblivious transfer): If the timestamp of the task request Req is within the time period (t_s, t_e), the edge node and the requester perform OT protocol. **Algorithm 2** describes the OT protocol between the edge node and the requester.

Step 4 (Submitting query requests): After the time period (t_s, t_e) and the driver get the position (x_d, y_d), the driver uploads the query request Que to the edge node.

Step 5 (Verifying messages): The edge node first verifies the integrity and validity of the message Que.

Step 6 (Range query): Then comparing the BF_D uploaded by the driver with the BF_R received in the time period (t_s, t_e). This process is the range query stage, which can find tasks within the range.

Step 7 (Returning query results): The edge node returns the random number R, outputs q_i and matching information $match$ to the driver, note that each task here corresponds to a random number R and the length is l. Note that R is $R_1R_2...R_l$.

Step 8: (Private equality test) Then the driver converts the real location (x_d, y_d) to the geohash code X and calculates $\delta = X \oplus R$. After that, the driver forwards δ to the requester that matchs the range to perform private equality test. **Algorithm 3** describes the private equality test algorithm.

Algorithm 2 Oblivious Transfer

```
Parameters:
  The functions of the symbols used in this algorithm are the same as in Algorithm 1
Protocol:
  1. [Verification] The edge node checks the integrity and timeliness of the message $Req$, and determines whether the timestamp of the task request $Req$ is within the time period $(t_s, t_e)$. If the check passes, the edge node and the requester perform OT protocol.
  2. [Random key] The requester chooses $\{k_i\} \leftarrow \{0,1\}^n$ at random, where $1 \leq i \leq l$
  3. [OT] For each $i$, the requester and the edge node execute the OT protocol
      a. The edge node inputs a bit $R_i$, note that $R_i$ is randomly selected by the edge node.
      b. The requester inputs a pair of strings $\{k_i, k_i \oplus \tilde{P}_i\tilde{P}_{i+l}...\tilde{P}_{i+(n-1)l}\}$ of length $n$, where $\tilde{P}_i$ represents the bit character of the ith bit
      c. The edge node receives the output $q_i$
```

Algorithm 3 Private Equality Test

```
  3. [Matrix Form]
    1. The requester receives $\delta$ and constructs a matrix $T_{l\times n}$ whose $i$-th row is the vector $t_i = k_i \oplus \bar{P}_i\bar{P}_{i+l}...\bar{P}_{i+(n-1)l} \oplus (\tilde{P}_i\tilde{P}_{i+l}...\tilde{P}_{i+(n-1)l}) \cdot \delta_i$
    2. The driver receives $q_i$ and constructs a matrix $U_{l\times n}$ whose ith row is the vector $u_i = q_i \oplus C(X_i)$
  4. [Private Equality Test]
    a. For each $j$, where $1 \leq j \leq n$, the driver and requester call the equality test function
    b. The requester enters each column $t^j$ of matrix $T_{l\times n}$ as the sender
    c. The driver enters each column $u^j$ of matrix $U_{l\times n}$ as receiver d. If $t^j = u^j$, the driver receives output 1
```

Step 9 (Decrypting and submit results): Next, the driver finds the nearest task, and the process of decrypting the task and submitting the result is the same as before.

VI. Security Analysis and Proofs

In this section, we provide security analysis and proofs for the PriTAEC scheme under semi-honest scenarios from both drivers' and requesters' perspectives.

A. Security Analysis

Our scheme is designed to preserve location privacy and task content privacy. On the functional side, our scheme can implement range query, fine-grained location matching, and task encryption and decryption. Therefore, we analyze the security of our scheme in terms of each of these three functions. Note that we analyze the security of our scheme under the semi-honest adversary model. The collusion problem and malicious users are out of the scope of this paper.

Range query. In this phase, requesters and drivers use Hilbert curves to map their two-dimensional locations to one-dimensional curves, which can map the user's real location into the grid cloaked area and play a certain role in location obfuscation. When users launch a request, their locations need to be bloom filtered before they are submitted to the fog node. The security of the grid range query depends on the one-way nature of the hash functions used by the Bloom filter. Since the fog node does not know the hash key of the Bloom filter used by the user, it cannot infer the real location of the user. To achieve semantic security, our scheme requires the requester to submit only one location grid, while the driver must submit at least one grid. This avoids two users uploading an identical location so that the fog node gets more information.

Fine-grained location matching. In this phase, users implement fine-grained location matching using grid location encoding, in which the security relies on the security nature of the oblivious transfer and private equality protocols. In terms of the oblivious transfer, the driver can only learn the message x_b when it inputs the selection bit b, and will not get any information about x_{1-b}. The requester only needs to input the string pair (x_0, x_1) and will not get any output. In terms of the privacy equality test, the driver and requester input x_0 and x_1 respectively, and if $x_0 = x_1$, the driver receives the bit 1, otherwise the driver receives the bit 0, while the requester does not get any information about x_0. In particular, the prefix family of the requester is perturbed using the permutation key $M_{n\times n}$. Since the driver does not know the permutation key $M_{n\times n}$, the driver cannot infer the location of the requester during the execution of the oblivious transfer and private equality protocols. Note that only the requester and the crowdsourcing server hold the permutation key $M_{n\times n}$. Since the oblivious transfer and private equality protocols are performed between the requester and the driver, the crowdsourcing server can only sense the distance between users using the permutation key $M_{n\times n}$, and cannot infer the location of any user.

Task encryption and decryption. In this phase, the security of the task content of our scheme relies on the security of hybrid encryption, which includes symmetric encryption (AES) and attribute-based encryption (CP-ABE) [13]. The requester uses the symmetric key K to encrypt the task content, and the symmetric key K is encrypted by the ciphertext policy L set by the requester. The symmetric key K cannot be broken if the fog node does not have matching attributes.

B. Security Proofs

In this section, we focus on proving the security of the task assignment, which involves the oblivious transfer and the private equality test cryptographic primitives. Specifically, we prove the security of the task assignment under a semi-honest adversary model, where an adversary can corrupt a requester or driver, noting that we assume that the edge node does not collude with any of the entity users. We prove that the task assignment process is secure, since edge nodes only know index vectors and do not know real locations of drivers and requesters, so edge nodes do not reveal the sensitive information of users. If the task assignment process is secure, we only need to show that views of corrupted parties can be simulated during protocol execution. We use the following theory to prove the security of the task assignment phase.

Theory 1 If a protocol is fully simulable, then its subprotocols are also fully simulable.

Theory 2 If the oblivious transfer and private equality test protocols satisfy security under semi-honest adversaries, then the task assignment can implement functional functions securely under semi-honest adversaries.

Proof: We give the proof under a hybrid model, where the OT protocol and the private equality test are computable under the ideal functional functions. Specifically, we prove that the adversary's view is simulable from two perspectives: the requester is corrupted and the driver is corrupted.

The requester is corrupted: let the adversary A controls the requester in the real world, we construct a simulator S, which invokes the A on input, and then the simulator S performs the specific protocol with the adversary A as an honest driver.

1. The input of the simulator S: the location code P and the matrix $T_{l\times n}$ of the adversary A.

2. The S simulates the oblivious transfer function: one side of the input is the value pair constructed by the adversary A according to the location code P. Since the S does not know the input of the honest driver, the S randomly selects the location code X as the input of the other side of the oblivious transfer function.

3. The S simulates the private equality test function: It inputs the columns u^j, where the u^j is generated by the S calls the oblivious transfer function. In this case, the adversary A does not get any output.

Since the requester does not get any output in the real protocol, we only need to show that the distribution of the ideal execution (the simulator S and the adversary A) and the distribution of the real execution (the adversary A and the honest driver) are computationally indistinguishable, as follows:

$$\{S(P, S^{OT}(P, T_{l\times n}), S^{PEQT}(t^j))\} \\ \overset{c}{\equiv}\{View(P, R^{OT}(P, T_{l\times n}), R^{PEQT}(t^j))\}$$

where the $S^{OT}(P, T_{l\times n})$ is the function used by the simulator S to get the requester's view in the OT protocol, the $S^{PEQT}(t^j)$ is the function used by the simulator S to get the requester's view in the private equality test protocol, and the View() represents the view of the real execution of the protocol. The $R^{OT}(P, T_{l\times n})$ denotes the income message of the requester in the real oblivious transfer execution, the $R^{PEQT}(t^j)$ denotes the income message of the requester in the real private equality test execution, and $1 \le j \le n$.

The only difference between the above two distributions is that the S randomly chooses the X instead of the driver's real input, and the simulator S and the adversary A perform the oblivious transfer protocol in which the S can generate the view of the adversary A in the real OT without knowing the driver's real input. Assuming that there is a probability polynomial adversary that can distinguish the two distributions, this implies that the adversary can learn the driver's bits, but this contradicts the security of the OT protocol. Similarly, it is clear from the properties of the private equality test that the requester's view can be generated without knowing the driver's input, and that the two distributions are computationally indistinguishable even if the P matches with the X.

The driver is corrupted: Let the adversary A controls the driver in the real world. We construct a simulator S whose input invokes the adversary A. The simulator S interacts with the adversary A as an honest requester.

1. The input of the simulator S: the location code X and the matrix $U_{l\times n}$ of the adversary A.

2. The S sends the X to the trusted third party, which uses the X to perform the task assignment. Then, the trusted third party returns the matching result to the simulator S. We assume that the matching results are $j_1, ..., j_b$, where $1 \leq j_1 \leq ... \leq j_b \leq n$, which means the j_i-th string matches with the X.

Since the S knows the location code X, for $j \in \{j_1, ..., j_b\}$, it can randomly select b strings from the prefix family of the X as the matching strings for the adversary A. However, for $j' \in \{1, ..., n\} - \{j_1, ..., j_b\}$, the S randomly selects the strings that are independent of the matching results.

3. The S simulates the oblivious transfer function, For $j \in \{j_1, ..., j_b\}$, S generates the value pairs honestly like the requester. For $j' \in \{1, ..., n\} - \{j_1, ..., j_b\}$, the S generates the value pairs based on the randomly chosen strings. Finally, the S sends the corresponding values to the adversary A.

4. The S simulates the private equality test function with the input t^j, where the t^j is generated by the oblivious transfer protocol. The final outputs are $j_1, ..., j_b$.

We claim that the distribution of the ideal model (the simulator S and the adversary A) and the distribution of the real execution (the adversary A and the honest requester) are computationally indistinguishable as follows:

$$\begin{aligned}&\{S(X, S^{OT}(X, U_{l\times n}), S^{PEQT}(u^j))\}\\ &\qquad\overset{c}{\equiv}\{View(X, R^{OT}(X, U_{l\times n}), R^{PEQT}(u^j))\}\end{aligned}$$

The only difference between the two distributions is that for $j' \in \{1, ..., n\} - \{j_1, ..., j_b\}$, the simulator S randomly selects the string, but this does not affect the results returned to the adversary A in the private equality test phase. Therefore, the adversary A is computationally indistinguishable between the two distributions.

In the optimization algorithm, the security of the task assignment of the PriTAEC scheme relies on the precomputing oblivious transfer, whose security proof can be referenced to [36].

VII. Performance Evaluation

In this section, we analyze the performance of schemes PriTAEC, LPRQ [37], and pMatch [38] and evaluate their performance experimentally. Table II describes the differences among our scheme, the pMatch scheme, and the LPRQ scheme.

A. Performance Analysis

Our scheme consists of four phases: grid range query, nearest task assignment, task encryption and decryption, and result submission.

Grid range query phase. In the grid range query phase, requesters and drivers need to map Hilbert values to Bloom filter values, and the computational cost of this process is $O(k\cdot m)$. The computational cost of implementing range queries at edge nodes is $O(m)$, where k is the number of hash functions invoked in Bloom filter, and m is the length of the binary string used by the Bloom filter.

Nearest task assignment. In the nearest task assignment phase, the requester and driver perform oblivious transfer and private equality test protocols. Since our scheme uses OT extensions, only some initial base OT instances are required, where the communication overhead is $O(\kappa^2)$ and the computation overhead is $O(\kappa)$. Note that κ is a security parameter for OT extensions. Any number of OTs can be constructed using these base OTs, the communication and computation overheads are only proportional to the total input size of the participants, and the computation only includes symmetric key operations.

In our scheme, l OT instances are involved, where each string is of length n, so the total overhead is $O(l \times n)$. In the private equality test phase, we set the security statistics parameter of the private equality test protocol to be λ and the probability of false positives to be $2^{-\lambda}$. This protocol also uses OT extensions which require constant symmetric key operations and $488 + \lambda$-bit communication overhead, note that the base OT required for the private equality test protocol can invoke the initial base OT constructed above. After the users perform the oblivious transfer and the private equality test, the edge node search for the nearest task for the driver, and the computational and communication overheads of this process are proportional to the number of tasks within the grid.

Task encryption and decryption. The communication and computation overheads of task encryption and task result submission depend on the communication and computation overheads of the attribute-based encryption [13] and the AES symmetric encryption.

LPRQ scheme and pMatch scheme. However, in the LPRQ scheme, the main time consumption for the task assignment comes from the exponential operation where the computational overheads of the requester and driver are $4C_e$ and $6C_e$, respectively. Note that C_e is a single exponential operation in $Z^*_{n^2}(|n^2| = 2048)$.

In the pMatch scheme, the main time consumption for the task assignment comes from the bilinear pairing operation E and hash operation H where the computational overheads of the requester and driver are $5E + H$ and $4E + H$, respectively, where we choose a symmetric elliptic curve $SS512$ with a 160-bit prime p.

B. Evaluation Results

Experimental Settings. We tested the performance of the task assignment phase in a virtual machine with Ubuntu 21.10 and C++, where the operating system memory is 2GB and

TABLE II: Comparison with related work

Scheme	Users' privacy	Range query	Edge computing	Access Control	Offline - Online Operation
pMatch	✓	—	×	—	×
LPRQ	✓	✓	✓	—	×
our scheme	✓	✓	✓	✓	✓

"✓" means satisfied, "×" means not satisfied and "—" means not involved .

the linux kernel is 5.13.0-22-generic. The main time overhead of the task assignment phase of our scheme comes from the oblivious transfer and private equality test protocols performed between the requester and the driver or the private equality test protocol performed between the requester and the driver in the optimization algorithm. Therefore, we tested the time overhead of the oblivious transfer and privte equality test using the code of emp-ot [39] and KKRT16 [40].

Experimental Results. We set the length of the location code to l. To protect the location code from leakage, the requester adds the disturbing code to the prefix family of the location codes and sets the number of elements of the prefix family to n. Note that n is greater than $l + 1$.

We first measure the impact of the location code length l on the time overhead of task assignments. As shown in Fig.7, we set n to 25 and 30, respectively, and then measure the time consumed for the task assignment when l varies from 10 to 20, where n determines the number of private equality test and l determines the number of oblivious transfers. It can be observed that the time overhead of the task assignment process is $53ms$ when $n = 25$ and $l = 10$, and $63ms$ when $n = 30$ and $l = 10$. We measure the effect of the number of elements n of the prefix family on the time overhead of task assignments. As shown in Fig.8, we set l to 10 and observe the time consumed for the task assignment as the number of elements n of the prefix family changes from 10 to 20. It can be observed that when $l = 10$ and $n = 20$, the time overhead of the task assignment process is $51ms$.

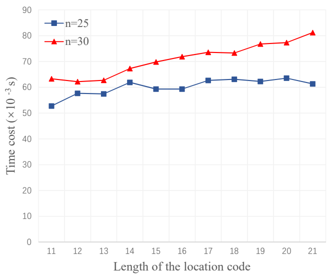

Fig. 7: Time cost of task assignment

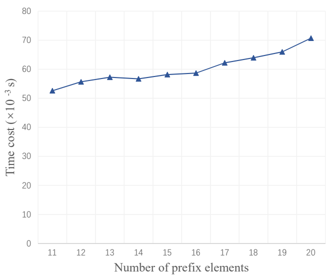

Fig. 8: Time cost of task assignment

In the optimization algorithm, we measure the time overhead of executing the oblivious transfer protocol between the driver and the requester as shown in Fig.9. It can be observed that the time overhead of the oblivious transfer protocol is $12ms$ when $n = 25$ and $l = 10$, and $13ms$ when $n = 30$ and $l = 10$. We measure the time overhead of executing the private equality protocol between the requester and the driver as shown in Fig.10, where n is set to 25 and 30, respectively, and the independent variable l varies from 10 to 20. It can be observed that the time overhead of the private equality protocol is $40ms$ when $n = 25$ and $l = 10$, and $50ms$ when $n = 30$ and $l = 10$.

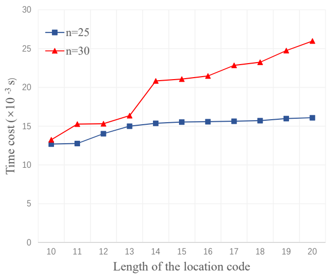

Fig. 9: Time cost of oblivious transfer

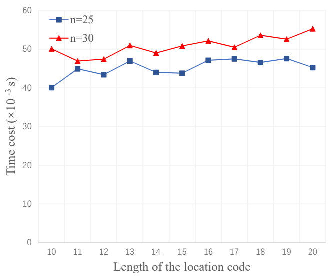

Fig. 10: Time cost of private equality test

In the optimization algorithm, we measure the time overhead of executing the oblivious transfer protocol between the edge node and the requester when l equals 10 as shown in Fig.11. It can be observed that when $l = 10$ and $n = 11$, the time overhead of the oblivious transfer protocol is $12ms$. We measure the time overhead of executing the private equality protocol between the driver and the requester as shown in Fig.12, where the independent variable n varies from 10 to 20. It can be observed that when $l = 10$ and $n = 11$, the time overhead of the private equality protocol is $40ms$.

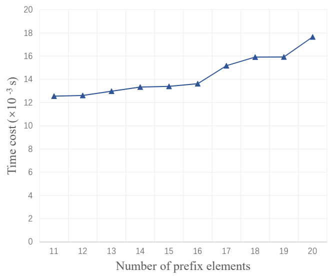

Fig. 11: Time cost of oblivious transfer

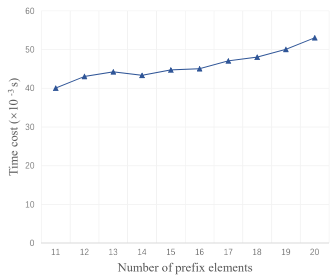

Fig. 12: Time cost of private equality test

We also compared the time overhead of the PriTAEC scheme, LPRQ scheme, and pMatch scheme in the task assignment phase. Fig.13 depicts the time consumption of the requester in the task assignment phase of the PriTAEC, LPRQ, pMatch scheme, where l=10, n=25, the number of keywords of pMatch are 6, and the independent variable is the number of requests varying from 10 to 100. Fig.14 depicts the time consumption of the driver in the task assignment phase of the PriTAEC, LPRQ, pMatch scheme, where l=10, n=25, the number of keywords of pMatch are 6, and the independent variable is the number of queries varying from 10 to 100. It can be observed that when the number of requests is 100, the time overheads of the task assignments of schemes LPRQ, pMatch, and PriTAEC are $9s$, $7s$, and $6s$, respectively.

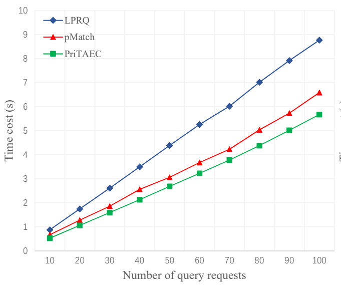

Fig. 13: Time cost of the requester

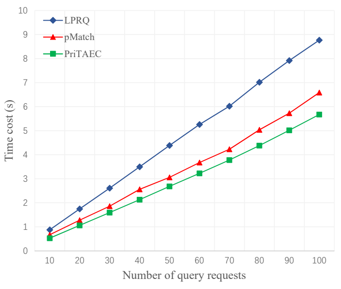

Fig. 14: Time cost of the driver

In summary, in our scheme, the time overhead of the task assignment grows with the values of l and n, where the value of l affects the number of the oblivious transfer, and the value of n affects the number of the private equality test. Through the above experimental analysis and comparison, we can conclude that our scheme can implement task assignments efficiently.

VIII. Conclusion

In this paper, we describe the privacy issues that exist in spatial crowdsourcing and design the PriTAEC scheme which can assign tasks efficiently and securely. Our scheme meets the designed goals of security, practicality, and timeliness. Considering security, we implement the grid range query using Bloom filters and then fine-grained nearest distance task assignments using oblivious transfer. Considering practicality, we combine symmetric encryption and attribute-based encryption to implement access rights setting. Considering timeliness, we use the paradigm of edge computing to reduce the communication latency and design the offline-online phase for task assignment to reduce the computational overhead of drivers. Finally, through theoretical and experimental, we conclude that the PriTAEC scheme can implement task assignments efficiently and securely.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Shandong Province under Grant ZR2020MF056 and Grant ZR2020KF011, in part by the National Natural Science Foundation of China under Grant 62071280, in part by the Major Scientific and Technological Innovation Project of Shandong Province under Grant 2020CXGC010115, and in part by the Henan Key Laboratory of Network Cryptography Technology LNCT2021-A12.

References

[1] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal*, vol. 29, no. 1, pp. 217–250, 2020.
[2] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 772–782, 2017.
[3] X. Shen, L. Wang, Q. Pei, Y. Liu, and M. Li, "Location privacy-preserving in online taxi-hailing services," *Peer-to-Peer Networking and Applications*, vol. 14, no. 1, pp. 69–81, 2021.
[4] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *Proceedings of the VLDB Endowment*, vol. 7, no. 10, pp. 919–930, 2014.
[5] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1298–1309.
[6] P. Hu, Y. Wang, Q. Li, Y. Wang, Y. Li, R. Zhao, and H. Li, "Efficient location privacy-preserving range query scheme for vehicle sensing systems," *Journal of Systems Architecture*, vol. 106, p. 101714, 2020.
[7] D. Yuan, Q. Li, G. Li, Q. Wang, and K. Ren, "Priradar: A privacy-preserving framework for spatial crowdsourcing," *IEEE transactions on information forensics and security*, vol. 15, pp. 299–314, 2019.
[8] S. Han, J. Lin, S. Zhao, G. Xu, S. Ren, D. He, L. Wang, and L. Shi, "Location privacy-preserving distance computation for spatial crowdsourcing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7550–7563, 2020.
[9] D. Hilbert, "Über die stetige abbildung einer linie auf ein flächenstück," in *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*. Springer, 1935, pp. 1–2.
[10] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[11] A. Liu, Z.-X. Li, G.-F. Liu, K. Zheng, M. Zhang, Q. Li, and X. Zhang, "Privacy-preserving task assignment in spatial crowdsourcing," 2017.
[12] V. Kolesnikov, M. Rosulek, and N. Trieu, "Swim: Secure wildcard pattern matching from ot extension," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 222–240.
[13] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.
[14] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gmission: A general spatial crowdsourcing platform," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1629–1632, 2014.
[15] X. Liu, K. Liu, L. Guo, X. Li, and Y. Fang, "A game-theoretic approach for achieving k-anonymity in location based services," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 2985–2993.
[16] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Achieving k-anonymity in privacy-aware location-based services," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 754–762.
[17] D. Yang, X. Fang, and G. Xue, "Truthful incentive mechanisms for k-anonymity location privacy," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 2994–3002.
[18] S. Zhang, X. Mao, K.-K. R. Choo, T. Peng, and G. Wang, "A trajectory privacy-preserving scheme based on a dual-k mechanism for continuous location-based services," *Information Sciences*, vol. 527, pp. 406–419, 2020.
[19] Z. Li, W. Li, F. Gao, P. Yu, H. Zhang, Z. Jin, and Q. Wen, "New blind filter protocol: An improved privacy-preserving scheme for location-based services," *The Computer Journal*, vol. 63, no. 12, pp. 1886–1903, 2020.
[20] I.-T. Lien, Y.-H. Lin, J.-R. Shieh, and J.-L. Wu, "A novel privacy preserving location-based service protocol with secret circular shift for k-nn search," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 863–873, 2013.
[21] T. Peng, Q. Liu, and G. Wang, "Enhanced location privacy preserving scheme in location-based services," *IEEE Systems Journal*, vol. 11, no. 1, pp. 219–230, 2014.
[22] B. Wang, M. Li, and L. Xiong, "Fastgeo: Efficient geometric range queries on encrypted spatial data," *IEEE transactions on dependable and secure computing*, vol. 16, no. 2, pp. 245–258, 2017.
[23] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 704–719, 2015.
[24] Z. Zheng, Z. Cao, and J. Shen, "Practical and secure circular range search on private spatial data," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 639–645.
[25] J. Zhang, F. Yang, Z. Ma, Z. Wang, X. Liu, and J. Ma, "A decentralized location privacy-preserving spatial crowdsourcing for internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2299–2313, 2020.

[26] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2020.

[27] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.

[28] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Network*, vol. 32, no. 3, pp. 78–83, 2018.

[29] H. Li, D. Han, and M. Tang, "A privacy-preserving charging scheme for electric vehicles using blockchain and fog computing," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3189–3200, 2020.

[30] L. Cui, Z. Chen, S. Yang, Z. Ming, Q. Li, Y. Zhou, S. Chen, and Q. Lu, "A blockchain-based containerized edge computing platform for the internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2395–2408, 2020.

[31] M. Li, L. Zhu, and X. Lin, "Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4573–4584, 2018.

[32] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 131–155, 2011.

[33] R. Fagin, M. Naor, and P. Winkler, "Comparing information without leaking it," *Communications of the ACM*, vol. 39, no. 5, pp. 77–85, 1996.

[34] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, "Secure multi-party computation: theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019.

[35] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Annual international cryptology conference*. Springer, 2005, pp. 258–275.

[36] D. Beaver, "Precomputing oblivious transfer," in *Annual International Cryptology Conference*. Springer, 1995, pp. 97–109.

[37] Q. Kong, R. Lu, M. Ma, and H. Bao, "Achieve location privacy-preserving range query in vehicular sensing," *Sensors*, vol. 17, no. 8, p. 1829, 2017.

[38] J. Shu, K. Yang, X. Jia, X. Liu, C. Wang, and R. H. Deng, "Proxy-free privacy-preserving task matching with efficient revocation in crowdsourcing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 117–130, 2018.

[39] X. Wang, A. J. Malozemoff, and J. Katz, "Emp-toolkit: Efficient multiparty computation toolkit," 2016.

[40] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious prf with applications to private set intersection," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 818–829.

Zihui Xu born in 1997, he is currently a graduate student of Information Science and Engineering of Shandong Normal University. His research interest includes privacy preservation and fog computing.

Lei Wu received his Ph.D. degree in applied mathematics from School of Mathematics, Shandong University in 2009. He is currently an associate professor with the School of Information Science and Engineering, Shandong Normal University, China. His research interests include applied cryptography, privacy preservation, and cloud computing security.

Chengyi Qin born in 1996, she is currently a graduate student of Information Science and Engineering of Shandong Normal University. Her research interest includes privacy preservation and cloud computing security.

Su Li born in 1997, she is currently a graduate student of Information Science and Engineering of Shandong Normal University. Her research interest includes privacy preservation and picture encryption.

Songnian Zhang received his M.S. degree from Xidian University, China, in 2016 and he is currently pursuing his Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. His research interest includes cloud computing security, big data query and query privacy.

Rongxing Lu (Fellow, IEEE) is a University Research Scholar, an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise (with H-index 79 from Google Scholar as of March 2022), and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.