Privacy-Preserving Task Matching With Threshold Similarity Search via Vehicular Crowdsourcing

Fuyuan Song, Zheng Qin, *Member, IEEE*, Dongxiao Liu, *Member, IEEE*, Jixin Zhang, Xiaodong Lin, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In vehicular crowdsourcing, task requesters rely on a server to distribute spatial crowdsourcing tasks to on-road vehicular workers based on interests and locations. To protect the privacy of the interests and locations, both requesters and workers prefer to encrypt the information before uploading them to the server. However, such an encryption-before-outsourcing paradigm makes it a challenging issue to conduct the task matching. In this paper, we propose a Privacy-Preserving Task Matching (PPTM) with threshold similarity search via vehicular crowdsourcing. We first propose an interest-based PPTM by transforming vehicular workers' interests into binary vectors. By using Symmetric-key Threshold Predicate Encryption (STPE) and proxy re-encryption, PPTM achieves privacy-preserving multi-keyword task matching with Jaccard similarity search in multi-worker multi-requester setting. Furthermore, by comparing the Euclidean distances between workers and requesters against a pre-defined threshold, PPTM preserves the location privacy of workers and requesters that only reveals the comparison results to the crowdsourcing server. The security analysis and extensive experiments demonstrate that PPTM protects the confidentiality of locations and interests of requesters and workers while achieving the efficient task matching.**

***Index Terms*—Vehicular crowdsourcing, task matching, similarity search, privacy-preserving.**

I. Introduction

With the proliferation of mobile devices and the ubiquity of wireless communications, vehicular crowdsourcing [1] has emerged as a novel and transformative paradigm that enables task requesters to release crowdsourcing tasks to a vehicular crowdsourcing server. The vehicular crowdsourcing server can distribute spatial crowdsourcing (SC) tasks to nearby vehicular workers of related interests [2]–[4]. To do so, the vehicular workers can search and complete the tasks on the crowdsourcing server (i.e., crowd server) based on their interests and locations. For instance, in the Meituan[1] platform, if a task requester uses his/her mobile device to submit a task of food delivery to the crowd server with rewards, the crowd server allocates the task to a nearby vehicular worker of related interest. Then, the suitable deliveryman (i.e., vehicular worker) accepts the delivery task and performs the assigned task. Finally, the deliveryman fetches the food from the designated restaurant and delivers it to the task requester.

Task matching plays an important role in vehicular crowdsourcing, which enables the crowd server to allocate SC tasks to vehicular workers based on their interests and locations. Specifically, given two pre-defined thresholds θ and δ, the crowd server performs threshold similarity search for spatial interest task matching. Namely, the Jaccard similarities between the interests of vehicular workers and the task requester should be greater than or equal to θ, and the Euclidean distances between the locations of vehicular workers and the task requester should be less than δ. By using the threshold similarity search, the crowd server can successfully check whether the interests and locations of the vehicular workers meet the requirements of the requesters. In [5], [6], according to the task requirements of the requesters and the locations and interests of the vehicular workers, the crowd server can accurately perform task-worker matching operations. However, such a matching mechanism inevitably discloses the private information of both task requesters and vehicular workers to the crowd server, such as home address, health status, vehicle trajectories, and habits. Specifically, the untrusted crowd server may sell vehicular workers' interests and locations to other entities for monetary reasons. Hence, the task requirements and vehicular workers' interests and locations should be protected. Several privacy-preserving task matching mechanisms [7], [8] have been proposed to achieve spatial crowdsourcing, and they mainly focus on worker privacy protection. Nevertheless, the crowd server can obtain the sensitive information of vehicular workers by analyzing the query results and task content. Therefore, it is important to explore a privacy-preserving spatial interest task matching mechanism in vehicular crowdsourcing that can simultaneously protect the privacy of tasks and workers.

Manuscript received December 24, 2020; revised March 10, 2021; accepted May 30, 2021. Date of publication June 14, 2021; date of current version July 20, 2021. This work was supported in part by the National Natural Science Foundation of China under Grants 61772191, 61902123, and 62002112, in part by National Key R&D Project under Grant 2018YFB0704000, in part by the Science and Technology Key Projects of Hunan Province under Grants 2019WK2072 and 2018TP3001, and in part by China Scholarship Council, the NSERC of Canada. The review of this article was coordinated by Dr. Zhipeng Cai. *(Corresponding author: Zheng Qin.)*

Fuyuan Song and Zheng Qin are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: fysong@hnu.edu.cn; zqin@hnu.edu.cn).

Dongxiao Liu and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: dongxiao.liu@uwaterloo.ca; sshen@uwaterloo.ca).

Jixin Zhang is with the School of Computer Science, Hubei University of Technology, Wuhan 430068, China (e-mail: zhangjixin@hnu.edu.cn).

Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N2L 3C5, Canada (e-mail: xlin08@uoguelph.ca).

Digital Object Identifier 10.1109/TVT.2021.3088869

[1][Online]. Available: https://www.meituan.com/

Currently, many efforts have been devoted to protect the dual privacy of both workers and requesters by using encryption-before-outsourcing method [7]–[9], such as searchable encryption [10], [11]. In the traditional searchable encryption schemes [10]–[14], the crowd server mainly performs secure search in the single-worker single-requester setting [10], [12], [13], [15], and excutes the single keyword search [11], [14]. However, different vehicular workers are interested in different tasks in vehicular crowdsourcing, and each worker's interests consist of multiple keywords. To address this issue, proxy re-encryption based task matching mechanisms [8], [9], [16], [17] have been designed, but most of them cannot support both spatial and interest crowdsourcing at the same time. To realize such functionality, some spatial keyword search schemes [18]–[20] have been proposed, which can provide spatial keyword query services over encrypted data. Unfortunately, most existing spatial keyword search schemes mainly obtain the query results based on the relative scores or the geometric range the worker owned, which may leak the sensitive information about the matched number between the workers' interests and the task requirements, or the workers' locations. With threshold similarity search, the crowd server executes compute-then-compare operations with a pre-defined threshold, which guarantees that only the magnitude of the query result and the threshold is revealed and no key information of the vehicular workers and the task requesters can be learned.

In this paper, we propose an efficient and Privacy-Preserving Task Matching (PPTM) scheme via vehicular crowdsourcing, which supports multi-keyword search in the multi-requester multi-worker setting. In vehicular crowdsourcing, if vehicular workers submit their locations and interests to the crowd server, the crowd server can securely perform task matching with threshold similarity search to assign SC tasks to the vehicular workers based on the location constraint and the interest constraint. To achieve multi-keyword search, we employ matrix decomposition to generate encryption keys and re-encryption keys. Moreover, we use proxy re-encryption to converse the encryption keys of both indexes and trapdoors to the symmetric keys of transformed indexes and trapdoors, which supports flexible key distribution in the multi-worker multi-requester setting. Our contributions are summarized as follows:

- We propose an efficient and Privacy-Preserving Task Matching (PPTM) scheme with threshold similarity search using vehicular crowdsourcing that preserves both task privacy and vehicular worker privacy.
- We utilize matrix decomposition and proxy re-encryption to support multi-keyword search in multi-user model. Furthermore, an aggregation solution is integrated into our PPTM to improve the task matching performance.
- We prove the security of PPTM in terms of the confidentiality of task requirements as well as interests and locations of vehicular workers. Moreover, experimental results demonstrate that the computational cost of PPTM is far less than the existing schemes.

The remainder of this paper is organized as follows. In Section II, we discuss the related works. In Section III, we introduce our system model, threat model, problem statement, and design goals. In Section IV, we provide the preliminaries. After that, we present the overview of PPTM in Section V. We give the detailed construction of PPTM in Section VI. We analyze the security of PPTM in Section VII. In Section VIII, we conduct experiments to evaluate the performance of PPTM. Finally, we conclude the paper in Section IX.

II. Related Work

In this section, we briefly introduce some existing works related to privacy-preserving task matching for crowdsourcing and searching over encrypted data.

A. Privacy-Preserving Task Matching for Crowdsourcing

Task matching has been extensively studied with the rapid development of crowdsourcing [5], [9], [16], [24]. In specific, vehicular crowdsourcing based platform [25] usually allocates SC tasks to the vehicular workers according to the worker models, such as worker's performance history [5], social profiles [6], locations [26], or interests [9], [20]. Meanwhile, considering that the crowd server is not fully trusted, privacy issues in crowdsourcing have been widely investigated in recent years, including worker privacy and task privacy. Since vehicular crowdsourcing usually requires locations and interests of vehicular workers to execute task matching, protecting vehicular workers' privacy (e.g., locations and interests) and task privacy is considerable issue. In particular, Gong *et al.* [7] proposed a flexible task recommendation to balance utility, security, and efficiency. Based on the differential privacy and geocasting, To *et al.* [27] proposed a privacy protection mechanism to preserve location privacy in spatial crowdsourcing. However, all these works [7], [27] only aim to protect worker privacy, and rely on a trusted authority to execute task matching over encrypted data. Shu *et al.* [16] employed Elgamal encryption to design a secure and efficient task recommendation method, which can simultaneously protect task privacy and worker privacy. However, it only supports single-keyword search. Furthermore, Shu *et al.* [9] utilized ASPE and proxy re-encryption to achieve multi-keyword search, but they use the matrix to denote the worker's keywords and introduces a key-based hash function to protect the privacy of worker's keywords, which may reduce task matching efficiency.

B. Searching Over Encrypted Data

To protect the privacy of tasks and workers, the requesters and workers usually encrypt their task requirements and workers' locations before outsourcing to the crowd server. Searching over encrypted data has been extensively studied, such as searchable symmetric encryption [9], [10], [12], [13] and public-key based searchable encryption [21]–[23], [28], [29]. Song *et al.* [10] first proposed the practical searchable encryption, and Boneh *et al.* first presented the public-key based keyword searchable encryption [29]. Subsequently, several searchable encryption schemes have been proposed, such as keyword ranked searchable encryption [13], fuzzy keyword searchable encryption [30], semantic search [12], boolean query [18]. However, all of them only support the single-user setting. That is, the

TABLE I
COMPARISON WITH PRIOR RELATED SCHEMES

Scheme	Multi-keyword	Multi-owner	User scalability	Non-interactive	Non-middleware	Security	Performance
MSDE [11]	×	√	√	√	√	COA	Low
MuED [14]	×	√	√	×	√	COA	Low
PPTR [9]	√	√	√	√	√	KPA	High
PRMSM [21]	√	√	√	√	×	COA	Low
KASE [22]	×	×	×	√	√	CPA	Low
MPSE [23]	×	√	×	√	√	CPA	Low
PPTM	√	√	√	√	√	CPA	Very high

search user who has a corresponding search token for each owner's data can be correctly matched. However, the single-user single-keyword search cannot directly apply to multi-keyword search in the multi-worker multi-requester model. To address this issue, in [11], [14], they utilized proxy re-encryption to achieve multi-user searchable encryption. Unfortunately, the server needs to encrypt the owner's symmetric key with the user's public key. By using the secret key, the user can decrypt the owner's symmetric key that can further recover the owner's ciphertexts to obtain the sensitive information. Zhang *et al.* [21] proposed a privacy-preserving ranked multi-keyword search with multi-owner (PRMSM) scheme. Nevertheless, PRMSM relies on an extra middleware to build indexes and generate trapdoors for the data owners and search users, which limits its usage in real vehicular crowdsourcing applications.

Different from the previous works, we aim to design an efficient and privacy-preserving spatial interest task matching scheme via vehicular crowdsourcing, which can achieve multi-keyword search in the multi-worker multi-requester setting. As shown in TABLE I, we compare our PPTM with the prior related schemes in terms of functionality and security.

III. SYSTEM MODEL, THREAT MODEL, PROBLEM STATEMENT, AND DESIGN GOALS

In this section, we formulate the system model, threat model, problem statement, and design goals.

A. System Model

The system model is derived from the task matching model in vehicular crowdsourcing. As shown in Fig. 1, there are four different entities in our system model, namely a key generation center (KGC), a crowd server, task requesters, vehicular workers.

- KGC: KGC is a trusted third party to generate secret keys and re-encryption keys, and distribute them to all participating entities.
- Crowd server: The crowd server has strong storage and computation abilities. On the one hand, the crowd server can store the vehicular workers' locations and interests. On the other hand, the crowd server can provide task matching services for the task requesters based on the interests and locations of the vehicular workers.
- Task requesters: The task requesters submit their SC tasks to the crowd server for finding the vehicular workers who have related interests and nearby positions. To protect the confidentiality of the task requirements, the task requesters generate trapdoors for the task requirements and send them to the crowd server.
- Vehicular workers: The on-road vehicular workers drive vehicles to find and complete the allocated tasks. In addition, the vehicular workers encrypt their locations and interests and upload them to the crowd server. According to the interests and locations of the vehicular workers, the crowd server allocates the tasks to suitable workers.

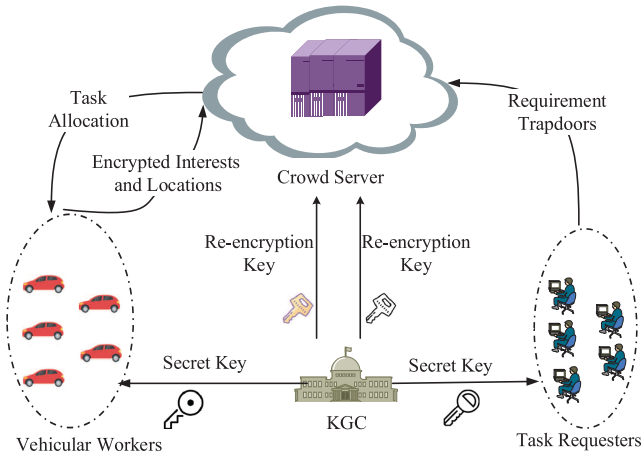


Fig. 1. System model.

B. Threat Model

In our threat model, KGC is fully trusted, and the communication between KGC and other participating entities is secure. The task requesters are fully trusted. The crowd server is semi-honest (i.e., honest-but-curious) [8], [17], [20], [31], which properly follows the designed protocols but may try to learn some sensitive information of the vehicular workers and task requesters (e.g., task requirements, task specification, as well as interests and locations of vehicular workers). What's more, the vehicular workers are semi-honest, and they may try to learn the sensitive information of the task requesters and the other vehicular workers. Moreover, we consider that there is no collusion between the crowd server and vehicular workers. Since the crowd server is usually viewed as a large server that considers the importance of reputation, collusion attacks are easy to detect and may reduce the reputation of the crowd server once caught [9]. Based on the available information that the crowd server may derive, we consider two attack models as follows.

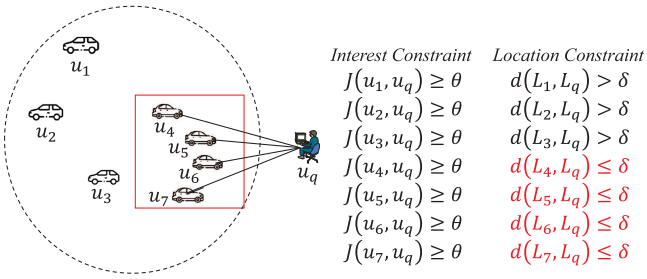


Fig. 2. An example of task matching via vehicular crowdsourcing.

- *Known-Plaintext-Attack (KPA):* The crowd server can obtain several plaintext-ciphertext pairs of objects (e.g., the workers' interests and the corresponding ciphertexts, the requesters' requirement, and the corresponding query trapdoor). Using the information of these plaintext-ciphertext pairs, the crowd server tries to obtain the secret keys for decrypting other ciphertexts.
- *Chosen-Plaintext-Attack (CPA):* The crowd server can obtain more information than what he gains in the KPA model. The crowd server can observe several ciphertexts as well as the corresponding plaintexts.

C. Problem Statement

Let $D = \{O_1, O_2, \ldots, O_m\}$ be a spatial interest database. Each object O_i $(i \in [1, m])$ in database D can be denoted as $\{W_i, L_i\}$, where $W_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,n_i}\}$ is a keyword set, i.e., interest description, and $L_i = (x_i, y_i)$ is a user's location in the database, i.e., coordinate. Each vehicular worker u_i obtains his/her task allocation from the crowd server according to its location and interest. The worker's interest I_i can be defined by a keyword set W_i, which belongs to a collection W of keywords. Before a task requester u_q publishes a new task requirement Q, which also can be denoted as a set of keywords, to the crowd server, the task requester sets two constraints for task matching. Firstly, the requester generates a pre-defined threshold θ, and the keyword-based Jaccard similarity between the worker and the requester should be greater than or equal to θ. Secondly, the requester generates another pre-defined threshold δ, and the Euclidean distance between the locations of the vehicular worker and the requester should be within the pre-defined threshold δ.

For task matching in vehicular crowdsourcing, the crowd server calculates the Jaccard similarity between the interest of the vehicular worker and the task requirement of the requester and selects the suitable worker whose Jaccard similarity is not less than θ. After that, the crowd server compares the Euclidean distances between these vehicular workers and task requester with δ, and the crowd server sends the task to the vehicular workers whose Euclidean distances to the requester are less than or equal to δ. As shown in Fig. 2, for the interest constraint, there exists 7 vehicular workers who satisfy the constraint of the interest-based Jaccard similarity search, i.e., $J(u_i, u_q) \geq \theta$ for $i \in [1, 7]$, where $J(\cdot, \cdot)$ denotes the Jaccard similarity. Then, the crowd server compares the Euclidean distances between 7 vehicular workers and task requester, and the crowd server chooses the vehicular workers whose Euclidean distances to the task requester are within δ. Here, $d(L_i, L_q) \leq \delta$ for $i \in [4, 7]$, where $d(\cdot, \cdot)$ denotes the Euclidean distance between two location points, and L_i, L_q are the vehicular worker's location and the requester's location, respectively. Thus, u_4, u_5, u_6, u_7 are the suitable vehicular workers for the given requirement. Finally, the crowd server allocates the task to these four workers to complete it. We formally define our spatial interest task matching in vehicular crowdsourcing and present an example as follows.

TABLE II
INTERESTS AND LOCATIONS OF VEHICULAR WORKERS

Vehicular worker	Keyword set	Location coordinate
u_1	$W_1 = \{w_1, w_2, w_3, w_4, w_6, w_7\}$	$L_1 = (2, 4)$
u_2	$W_2 = \{w_1, w_2, w_5, w_8\}$	$L_2 = (5, 1)$
u_3	$W_3 = \{w_1, w_2, w_3, w_6, w_7, w_8\}$	$L_3 = (5, 8)$
u_4	$W_4 = \{w_1, w_2, w_3, w_8\}$	$L_4 = (9, 12)$
u_5	$W_5 = \{w_1, w_2, w_3, w_6\}$	$L_5 = (6, 7)$

Definition 1 (Task Matching): Given a task requirement with keyword set W_q, a Jaccard similarity threshold θ, and a Euclidean distance similarity threshold δ, the spatial interest task matching via vehicular crowdsourcing can be determined by two constraints: (i) Interest constraint. That is, the keyword set $W_i \in W$ of the vehicular worker u_i, whose Jaccard similarity with the keyword set W_q of the requirement is not less than θ, i.e., $\{W_i \in W | J(W_i, W_q) \geq \theta\}$; (ii) Location constraint. The Euclidean distance between the worker's location and the requester's location should be within δ, i.e., $d(L_i, L_q) \leq \delta$.

Example 1: Table II shows a collection of vehicular workers $\{u_1, u_2, u_3, u_4, u_5\}$, and each vehicular worker has its keyword set (i.e., interests) and location coordinate. Suppose the keyword set of task requirement $W_q = (w_1, w_2, w_3, w_6, w_8)$, the location coordinate of task requester $L_q = (4, 8)$, the Jaccard similarity threshold θ is 0.8, and the Euclidean distance threshold δ is 2. Based on the Definition 1, the crowd server first chooses $\{u_3, u_4, u_5\}$ as the preselected vehicular workers according to the Jaccard similarity search, because the Jaccard similarity between W_q and W_3, W_4, W_5 satisfy $J(W_3, W_q) = \frac{5}{6} \geq \theta$, $J(W_4, W_q) = \frac{4}{5} \geq \theta$, and $J(W_5, W_q) = \frac{4}{5} \geq \theta$. After obtaining the vehicular workers who satisfy Jaccard similarity search, the crowd server calculates Euclidean distances over the locations L_3, L_4, L_5 and L_q, respectively. Finally, the crowd server acquires the suitable vehicular worker u_3, and allocates the task to u_3, because $d(L_3, L_q) = 1 < \delta$, $d(L_4, L_q) = 6.4 > \delta$, and $d(L_5, L_q) = 2.24 > \delta$.

Before outsourcing the database D to the crowd server, the vehicular worker needs to build and encrypt an index $\mathcal{I}$, represented as $Enc(\mathcal{I}) = \{(C_{B_1}, C_{L_1}), (C_{B_2}, C_{L_2}), \ldots, (C_{B_m}, C_{L_m})\}$, where C_{B_i} and C_{L_i} are the encrypted binary vector corresponding to the keyword set and the encrypted location, and encrypt the spatial interest database D as $Enc(D) = \{Enc(O_1), Enc(O_2), \ldots, Enc(O_m)\}$. Besides, the requester needs to encrypt his/her requirement $\mathcal{Q}$, denoted as $Enc(\mathcal{Q}) = (T_Q, C_{L_q})$, where T_Q is the encrypted task keywords, and C_{L_q} is the encrypted location. Specifically, we define our privacy preserving task matching as follows.

Definition 2 (Privacy Preserving Task Matching, PPTM): Given an encrypted spatial interest database $Enc(D)$, an encrypted index $Enc(\mathcal{I})$, and a trapdoor $Enc(\mathcal{Q})$, a PPTM is to retrieve a subset $\Gamma = \{Enc(O_1), Enc(O_2), \ldots, Enc(O_{m'})\}$ from $Enc(D)$ by using $Enc(\mathcal{I})$ and $Enc(\mathcal{Q})$ without decrypting on the crowd server such that for any $Enc(O_i) \in \Gamma$ $(i \in [1, m'])$, both $C_{B_i} \circ T_Q \geq 0$ and $C_{L_i} \circ C_{L_q} \geq 0$ hold, where $C_{L_i} \circ C_{L_q}$ denotes the inner product of C_{L_i} and C_{L_q}.

Remark: To execute privacy-preserving task matching based on the interest constraint and the location constraint, we transform Jaccard similarity and Euclidean distance similarity in the plaintext into the inner product in the ciphertext.

D. Design Goals

The aim of PPTM is to achieve efficient spatial interest task matching using vehicular crowdsourcing while preserving the confidentiality of requesters' requirements as well as vehicular workers' locations and interests under the above threat model. In particular, the following goals should be achieved in our scheme.

- *Multi-keyword search:* PPTM should support multi-keyword search in the multi-worker multi-requester setting for spatial interest task matching.
- *Index confidentiality and trapdoor confidentiality:* PPTM should guarantee that the crowd server cannot derive the sensitive information of indexes (i.e., vehicular workers' interests and locations) and trapdoors (i.e., task requirements).
- *Index unlinkability and trapdoor unlinkability:* In the task matching process, indexes and trapdoors are exposed to the crowd server. To protect privacy, the indexes and trapdoors should be randomized such that the crowd server cannot deduce the relationship between these indexes and trapdoors from their corresponding ciphertexts. That is, the crowd server cannot infer whether two workers have the same interests or keywords. Moreover, the ciphertexts of the same requirement are different in different task matching processes.
- *Efficiency:* PPTM should achieve an efficient task matching process. Namely, PPTM should minimize the computational cost of task matching in vehicular crowdsourcing.

IV. Preliminaries

In this section, we introduce some building blocks that are used in our proposed scheme, including similarity search [32], asymmetric scalar-product-preserving encryption (ASPE) [33], and symmetric-key threshold predicate encryption (STPE) [34].

A. Similarity Search

Suppose that there are m vehicular workers $\{u_1, u_2, \ldots, u_m\}$ in the vehicular crowdsourcing, each worker has its interest I_i and location L_i. Meanwhile, the worker's interest can be denoted as a set of keywords. Then, we assume that $W = \{W_1, W_2, \ldots, W_m\}$ is a collection of keyword sets, and each keyword set $W_i \in W$ is a subset of $\mathcal{W}$, i.e., $W_i \subseteq \mathcal{W}$, where $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ is the whole collection of keywords for all of the vehicular workers, i.e., keyword dictionary.

According to the vehicular workers' interests and locations, the crowd server executes the similarity search to perform task matching for a given task requirement. Generally speaking, there are two manners to define the similarity search based on the interests and locations [34]:

- Jaccard similarity search: For the vehicular workers' interests, it returns the keyword sets whose Jaccard similarity with the keyword set of requirement is not less than a pre-defined threshold θ.
- Euclidean distance similarity search: For the vehicular workers' locations, it returns the vehicular workers' locations whose Euclidean distances to the requester's location is within a pre-defined threshold δ.

Considering that the task requirement consists of two parts, namely keyword set and location, we use Jaccard similarity search to obtain the vehicular workers who have the matched interests. Subsequently, we utilize Euclidean distance similarity search to find the suitable workers whose Euclidean distances to the task requester are less than or equal to δ, i.e., $d(L_i, L_q) \leq \delta$.

To measure the similarity of keyword sets, we utilize Jaccard similarity as the similarity metric, which can be defined as follows.

- For any two keyword sets W_i and W_j, the Jaccard similarity $J(W_i, W_j)$ can be represented by the division between the size of $W_i \cap W_j$ and the size of $W_i \cup W_j$:

$$J(W_i, W_j) = \frac{|W_i \cap W_j|}{|W_i \cup W_j|}.$$

- For any two n-dimensional binary vectors B_i and B_j, the Jaccard similarity $J(B_i, B_j)$ can be denoted as

$$J(B_i, B_j) = \frac{B_i \circ B_j}{|B_i|^2 + |B_j|^2 - B_i \circ B_j},$$

where $|B_i|^2 = B_i \circ B_i$ and $|B_j|^2 = B_j \circ B_j$. The range of $J(B_i, B_j)$ is [0,1].

Each keyword set W_i $(i \in [1, m])$ can be encoded by using n-dimensional binary vector, with "1" denotes its existence in the keyword set and "0" otherwise. Therefore, each keyword set $W_i \in W$ can be equivalent converted to be a binary vector $B_i = (b_1, b_2, \ldots, b_n)$, where $b_i = 1 (i \in [1, n])$ if $w_i \in W_i$; otherwise, $b_i = 0$. By doing so, the Jaccard similarity between two keyword sets can be transformed to the Jaccard similarity between two corresponding binary vectors, i.e., $J(W_i, W_j) = J(B_i, B_j)$.

B. Asymmetric Scalar Product Preserving Encryption

Asymmetric scalar product-preserving encryption (ASPE) is an important technique for similarity search over encrypted data, by which one can obtain k vectors in the database that are closest to a given query vector [33]. In ASPE, the Euclidean distances between two binary vectors B_1, B_2, and a binary vector Q can be measured by the inner product of these corresponding vectors. Specifically, the order of the inner products in the plaintext between the data vectors and the query vector is maintained by that in the ciphertext. That is, if $C_Q \circ C_{B_1} \leq C_Q \circ C_{B_2}$, then

their plaintext form $Q \circ B_1 \leq Q \circ B_2$ holds, where C_{B_1}, C_{B_2} and C_Q are the ciphertexts of B_1, B_2 and Q, respectively.

C. Symmetric-Key Threshold Predicate Encryption

Symmetric-key Threshold Predicate Encryption (STPE) is a functional encryption, where query algorithm outputs a function over the plaintext instead of plaintext itself [35]. Moreover, given a spatial keyword of vehicular worker and a requirement of a requester, the crowd server can determine whether the Jaccard similarity and Euclidean distance similarity between the vehicular worker and the task requester are within the threshold θ and δ, respectively. Hence, STPE can realize task matching while preserving the privacy of both worker and task. Specifically, the STPE consists of five algorithms as follows:

- $\mathsf{STPE.Setup}(1^\lambda) \rightarrow msk$: Given a security parameter λ, the setup algorithm generates a master secret key msk.
- $\mathsf{STPE.KeyGen}(msk) \rightarrow sk$: Given the master secret key msk, the key generation algorithm outputs a secret key sk.
- $\mathsf{STPE.Enc}(sk, B) \rightarrow C_B$: Given the secret key sk and a binary vector B, the encryption algorithm outputs the encrypted binary vector C_B.
- $\mathsf{STPE.TrapGen}(sk, Q) \rightarrow T_Q$: Given the secret key sk and a binary vector Q of task requirement, the trapdoor generation algorithm outputs a search trapdoor T_Q.
- $\mathsf{STPE.Query}(C_B, T_Q) \rightarrow \Lambda = \{0, 1\}$: Given the encrypted vector C_Q and the search trapdoor T_Q, the query algorithm outputs a predicate function Λ, which satisfies the following formula:

$$\Lambda = \begin{cases} 1, & J(B, Q) \geq \theta \\ 0, & otherwise, \end{cases}$$

where θ is a pre-defined threshold.

V. Overview of PPTM

In this section, we first present our main idea of Privacy-Preserving Task Matching (PPTM). Then, we provide a framework of PPTM.

A. Main Idea

The task matching with threshold similarity search via vehicular crowdsourcing can be performed based on the interest constraint and the location constraint. The interest-based task matching over binary vectors needs to execute the Jaccard similarity search, and the location-based task matching requires to implement Euclidean distance similarity search. As shown in Fig. 3, both the searchable index and query trapdoor can be represented as n-dimensional binary vectors. To check whether a binary vector of vehicular worker matches a binary vector of task requester is equivalent to check whether each "1" bit of the index is equal to the corresponding bit of trapdoor; otherwise, the index does not match the corresponding bit of trapdoor. Namely, the task is not suitable for the worker. A naive privacy-preserving task matching method is to encrypt each index and traverse all indexes for a trapdoor to find out the binary vectors of the vehicular workers satisfying the interest constraint of the Jaccard

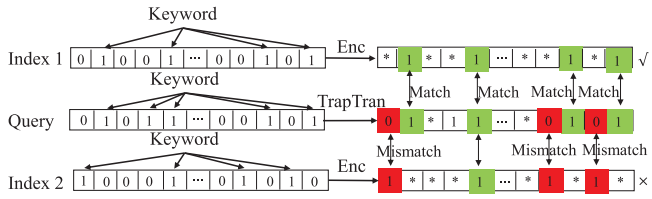


Fig. 3. An example of matched index.

similarity search. However, query processing is time-consuming over a large number of vehicular workers and high-dimensional binary vectors. To improve the search performance, by using the aggregation method, the binary vectors with the same prefix can be aggregated into one vector, and several wildcards "$*$" can be used to represent these different bits to represent the suffix. By doing so, the crowd server only needs to execute a one-time query operation to check whether there is a binary vector that meets the condition of the Jaccard similarity search in the collection. If not, the crowd server prunes the collection of binary vectors; otherwise, repeating the Jaccard similarity search over the subsets of the collection.

As shown in Fig. 4(a), the collection of keyword sets contains three binary vectors $\{(1,1,0,1,1,0,1,0), (1,1,0,1,0,1,0,1), (1,1,0,1,0,0,0,1)\}$ with the same prefix (1,1,0,1), which can be aggregated into one vector $B = (1,1,0,1,*,*,*,*)$. In the aggregated vector B, the wildcard "$*$" represents that the fifth, sixth, seventh, and eighth entries are either 0 or 1. Assume that any aggregated vector can be represented as an n-dimensional vector $B = (B_{n_1}, B_{n_2})$, where B_{n_1} is an n_1-dimensional binary vector, i.e., $B_{n_1} \in \{0,1\}^{n_1}$, $B_{n_2} = \{*\}^{n_2}$, and $n_1 + n_2 = n$. By doing so, we can integrate n-dimensional binary vectors with the same prefix B_{n_1} into one collection, i.e., the collection of binary vectors is $\mathcal{B} = \{\widetilde{B} | \widetilde{B} \in \{0,1\}^n, \widetilde{B}_{n_1} = B_{n_1}\}$. Let Λ_Q denotes the predicate function to evaluate the Jaccard similarity between $\widetilde{B}$ and Q, where $\widetilde{B} \in \mathcal{B}$, $Q \in \{0,1\}^n$, and $\theta \in [0,1]$. Generally speaking, $\Lambda_Q = 1$ holds if and only if there exists at least one binary vector $\widetilde{B} \in \mathcal{B}$ such that $J(\widetilde{B}, Q) \geq \theta$. That is, the maximal Jaccard similarity between Q and the binary vector in $\mathcal{B}$ is not less than θ.

After acquiring the initial workers based on the interest constraint with the Jaccard similarity search, the crowd server utilizes the Euclidean distance similarity search to obtain the suitable workers whose locations satisfy $d(L_i, L_q) \leq \delta$.

B. Framework of PPTM

In this subsection, we present a framework of PPTM. Since the location-based PPTM and interest-based PPTM have the same operations for task matching, we just describe the entire process of interest-based PPTM as follows:

- $msk \leftarrow \mathsf{PPTM.Setup}(1^\lambda)$: Given a security parameter λ, KGC executes the system setup algorithm, and the algorithm outputs a master secret key msk.
- $\{(sk_i, rk_i), (sk_q, rk_q)\} \leftarrow \mathsf{PPTM.KeyGen}(msk, u_i, u_q)$: Given the master secret key msk, a vehicular worker's identity u_i and a task requester's identity u_q, KGC executes

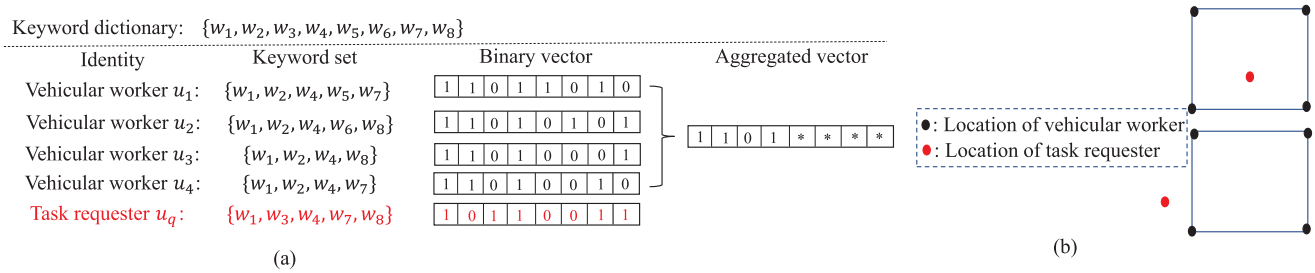


Fig. 4. An example of keyword encoding and all location situations about a task requester and the bounding area of vehicular workers.

the key generation algorithm to generate the secret key sk_i for vehicular worker u_i and the corresponding re-encryption key rk_i for the crowd server, and KGC also generates the secret key sk_q for the requester u_q and the corresponding re-encryption key rk_q for the crowd server.

- $\widetilde{C_{B_i}} \leftarrow \mathsf{PPTM.IndexEnc}(sk_i, B_i)$: Given the secret key sk_i and a binary vector B_i of the vehicular worker u_i, u_i executes the index encryption algorithm, and the algorithm outputs a ciphertext $\widetilde{C_{B_i}}$ for the binary vector B_i.
- $C_{B_i} \leftarrow \mathsf{PPTM.IndexTran}(rk_i, \widetilde{C_{B_i}})$: Given the re-encryption key rk_i of the vehicular worker u_i, the crowd server executes the index transformation algorithm and transforms the ciphertext $\widetilde{C_{B_i}}$ to C_{B_i}.
- $\widetilde{T_Q} \leftarrow \mathsf{PPTM.TrapGen}(sk_q, Q)$: Given a binary vector Q of the task requester u_q and the secret key sk_q, u_q executes the trapdoor generation algorithm, and the algorithm outputs a trapdoor $\widetilde{T_Q}$ for the binary vector Q.
- $T_Q \leftarrow \mathsf{PPTM.TrapTran}(rk_q, \widetilde{T_Q})$: Given the re-encryption key rk_q of the task requester u_q, the crowd server executes the trapdoor transformation algorithm and transforms the trapdoor $\widetilde{T_Q}$ to T_Q.
- $\mathcal{R} \leftarrow \mathsf{PPTM.Query}(\{C_{B_i}\}_{i=1}^m, T_Q)$: Given the transformative ciphertexts $\{C_{B_i}\}_{i=1}^m$ and transformative trapdoor T_Q, the crowd server executes the task matching algorithm for interest crowdsourcing, and outputs $\Lambda_Q = 1$ if $C_{B_i} \circ T_Q \geq 0$. As a result, the crowd server obtains the query result $\mathcal{R} = \{C_{B_i} | C_{B_i} \circ T_Q \geq 0\}$.

Subsequently, for the location constraint, the crowd server similarly executes the task matching mechanism, and obtains the suitable vehicular workers whose locations satisfy $d(L_i, L_q) \leq \delta$. Thus, the final query result is $\mathcal{R}' = \{u_i | C_{B_i} \circ T_Q \geq 0 \tilde{} \& \tilde{} C_{L_i} \circ C_{L_q} \geq 0\}$.

VI. Construction of PPTM

In this section, we give the detailed construction of PPTM. By leveraging the techniques of ASPE, STPE, and proxy re-encryption, PPTM can achieve efficient and privacy-preserving task matching based on the interest constraint and the location constraint in vehicular crowdsourcing.

In PPTM, we first randomly generate a master secret key with two invertible matrices, and we randomly decompose the invertible matrix into two matrices by using matrix decomposition. Then, based on the proxy re-encryption, PPTM realizes task matching by utilizing vehicular crowdsourcing without key sharing over multiple workers and multiple requesters. To improve the query performance and handle the large-scale task queries, we employ the aggregation method to design our PPTM. As shown in Fig. 4(a), to reduce the search complexity, we use the aggregation method to achieve one-time query operation over a collection of binary vectors. In Fig. 4(b), we can observe that there are two location situations between a task requester (i.e., red point) and vehicular workers (i.e., black points). The first situation is that the task requester locates in the bounding area of vehicular workers, and the other situation is that the task requester is outside the bounding area of vehicular workers. To concern the location constraint, we utilize the Euclidean distance similarity search to realize the task-worker matching. Typically, the crowd server determines whether the Euclidean distance between the vehicular worker and the task requester is less than the threshold δ is equivalent to check whether the inner product of the corresponding encrypted vectors is positive.

Next, we first present the detailed construction of interest-based PPTM. Then, after obtaining the query result $\mathcal{R}$ from the interest-based PPTM, we design the location-based PPTM, which can find the suitable vehicular workers who both satisfy the interest constraint and the location constraint. Finally, we give the correctness analysis of both interest-based PPTM and location-based PPTM.

A. Interest-Based PPTM

Let $\mathbb{B} = \{B = (B_{n_1}, B_{n_2}) | B_{n_1} \in \{0,1\}^{n_1}, B_{n_2} \in \{*\}^{n_2}\}$ denotes the class of binary vectors of vehicular workers. Assume that $B = (B_{n_1}, B_{n_2})$ is a binary vector of a vehicular worker in the class $\mathbb{B}$ with the same prefix, we have the following definition.

Definition 3: Given a binary vector B and a collection $\mathcal{B}$ of binary vectors with the same prefix of B, namely $\mathcal{B} = \{\widetilde{B} | \widetilde{B} \in \{0,1\}^n, \widetilde{B}_{n_1} = B_{n_1}, \widetilde{B}_{n_2} \in \{*\}^{n_2}\}$, the maximal Jaccard similarity between the binary vector Q and the binary vectors in $\mathcal{B}$ is $J(\widetilde{B}_{max}, Q)$, where $\widetilde{B}_{max} = (B_{n_1}, Q_{n_2})$. Thus, the predicate function Λ_Q is equal to 1 if and only if the maximal Jaccard similarity is not less than θ, i.e., $J(\widetilde{B}_{max}, Q) \geq \theta$.

Based on the Definition 2 and Definition 3, we present the detailed construction of interest-based PPTM as follows.

1) $\mathsf{PPTM.Setup}(1^\lambda)$. Given a security parameter λ, KGC randomly generates two $(3n+4) \times (3n+4)$ invertible matrices $\{M_1, M_2\}$, a $(3n+4)$-dimensional bit vector S, and a

permutation $\pi : \mathbb{R}^{3n+4} \rightarrow \mathbb{R}^{3n+4}$ as the master secret key, where n denotes the size of keyword dictionary $\mathcal{W}$. The master secret key msk can be denoted as $msk = \{M_1, M_2, S, \pi\}$.

2) PPTM.KeyGen(msk, u_i, u_q). Given a vehicular worker's identity u_i, KGC first randomly selects two $(3n+4) \times (3n+4)$ invertible matrices $\{M_{u_{i,1}}, M_{u_{i,2}}\}$, and then KGC computes $M'_{u_{i,1}} = M^{-1}_{u_{i,1}} M_1$, $M'_{u_{i,2}} = M^{-1}_{u_{i,2}} M_2$. After that, KGC generates the secret key sk_i and the re-encryption key rk_i for the worker u_i and the crowd server as $sk_i = \{M_{u_{i,1}}, M_{u_{i,2}}, S, \pi\}$ and $rk_i = \{M'_{u_{i,1}}, M'_{u_{i,2}}\}$.

For a task requester u_q, KGC first selects two random $(3n+4) \times (3n+4)$ invertible matrices $\{M_{u_{q,1}}, M_{u_{q,2}}\}$, and then KGC computes $M'_{u_{q,1}} = M^{-1}_1 M^{-1}_{u_{q,1}}$, $M'_{u_{q,2}} = M^{-1}_2 M^{-1}_{u_{q,2}}$. After that KGC generates the secret key sk_q and the re-encryption key rk_q as $sk_q = \{M_{u_{q,1}}, M_{u_{q,2}}, S, \pi\}$ and $rk_q = \{M'_{u_{q,1}}, M'_{u_{q,2}}\}$.

Finally, KGC sends the secret keys sk_i, sk_q to u_i, u_q, respectively, and sends the re-encryption keys rk_i, rk_q to the crowd server.

3) PPTM.IndexEnc(sk_i, B_i). Given a binary vector B_i of vehicular worker u_i, u_i performs the following operations to build index for task matching.

- Step 1: Firstly, u_i extends the n-dimensional binary vector $B_i = (B_{i,n_1}, B_{i,n_2}) = (b_{i,1}, b_{i,2}, \ldots, b_{i,n_1}, *, \ldots, *)$ to a $2n$-dimensional binary vector $B'_i = (b'_{i,1}, b'_{i,2}, \ldots, b'_{i,2j-1}, b'_{i,2j}, \ldots, b'_{i,2n})$, which can be set as follows.

$$b'_{i,2j-1} = \begin{cases} b_{i,j}, & j \in [1, n_1] \\ 0, & j \in (n_1, n]. \end{cases} \tag{1}$$

$$b'_{i,2j} = \begin{cases} 0, & j \in [1, n_1] \\ 1, & j \in (n_1, n]. \end{cases} \tag{2}$$

- Step 2: Next, u_i constructs an n-dimensional vector $\overline{B_i}$, whose elements can be set as shown in 3.

$$\overline{b_{i,j}} = \begin{cases} 0, & j \in [1, n_1] \\ q_j, & j \in (n_1, n]. \end{cases} \tag{3}$$

- Step 3: Then, u_i further extends B'_i to a $(3n+4)$-dimensional vector $\widehat{B_i} = (\alpha B'_i, \alpha \overline{B_i}, \alpha, \alpha |B_{i,n_1}|^2, r_b, 0)$, where α, r_b are one-time random positive numbers.
- Step 4: After that, u_i permutes $\widehat{B_i}$ to another $(3n+4)$-dimensional vector $\widetilde{B_i} = \pi(\widehat{B_i})$.
- Step 5: For each vector $\widetilde{B_i}$, u_i splits it into two random vectors as $\{\widetilde{B_i^a}, \widetilde{B_i^b}\}$. For z from 1 to $3n+4$, u_i sets each elements as shown in 4.

$$\begin{cases} \widetilde{B_i^a}[z] = \widetilde{B_i^b}[z] = \widetilde{B_i}[z], & S[z] = 0 \\ \widetilde{B_i^a}[z] + \widetilde{B_i^b}[z] = \widetilde{B_i}[z], & S[z] = 1. \end{cases} \tag{4}$$

- Step 6: Finally, u_i encrypts $\{\widetilde{B_i^a}, \widetilde{B_i^b}\}$ as $\widetilde{C_{B_i}} = \{M^T_{u_{i,1}} \widetilde{B_i^a}, M^T_{u_{i,2}} \widetilde{B_i^b}\}$. Thus, the interests of vehicular workers can be encrypted as $\{\widetilde{C_{B_i}}\}_{i=1}^{m'}$, where m' denotes the number of aggregated collections $\mathcal{B}$, here $m' < m$. Then, the vehicular workers upload the ciphertexts $\{\widetilde{C_{B_i}}\}_{i=1}^{m'}$ to the crowd server along with his/her identity u_i.

4) PPTM.IndexTran$(rk_i, \widetilde{C_{B_i}})$. Once receiving the ciphertext $\widetilde{C_{B_i}}$ from the vehicular worker u_i, the crowd server first finds the corresponding re-encryption key rk_i, and then re-encrypts the ciphertext $\widetilde{C_{B_i}}$ as shown in 5.

$$\begin{aligned} C_{B_i} &= \{M'^T_{u_{i,1}} M^T_{u_{i,1}} \widetilde{B_i^a}, M'^T_{u_{i,2}} M^T_{u_{i,2}} \widetilde{B_i^b}\} \\ &= \{M_1^T \widetilde{B_i^a}, M_2^T \widetilde{B_i^b}\}. \end{aligned} \tag{5}$$

Finally, the ciphertext of u_i's vector $\widetilde{C_{B_i}}$ can be transformed to C_{B_i}. According to the interests of vehicular workers, the crowd server stores the searchable indices $\{C_{B_i}\}_{i=1}^{m'}$ for task matching.

5) PPTM.TrapGen(sk_q, Q). Given a secret key sk_q, a task requester u_q specifies an n-dimensional keyword-based binary vector of requirement as $Q = (q_1, q_2, \ldots, q_n)$, and a Jaccard similarity threshold θ. Then, u_q generates a corresponding trapdoor as follows.

- Step 1: Task requester u_q first extends n-dimensional binary vector Q to a $2n$-dimensional vector $Q' = (q'_1, q'_2 \cdots, q'_{2n-1}, q'_{2n})$, whose entries can be set as follows.

$$\begin{cases} q'_{2i-1} = q_i, & i \in [1, n] \\ q'_{2i} = q_i^2, & i \in [1, n]. \end{cases} \tag{6}$$

- Step 2: Next, u_q further extends Q' to a $(3n+4)$-dimensional vector $\widehat{Q} = ((1+\theta)\beta Q', -\theta\beta Q, -\theta\beta |Q|^2, -\theta\beta, 0, r_q)$, where β, r_q are one-time random positive numbers.
- Step 3: Then, u_q permutes $\widehat{Q}$ to another $(3n+4)$-dimensional vector as $\widetilde{Q} = \pi(\widehat{Q})$ by using the same permutation π.
- Step 4: For the permuted vector $\widetilde{Q}$, u_q splits it into two vectors as $\{\widetilde{Q^a}, \widetilde{Q^b}\}$. For z from 1 to $3n+4$, u_q sets each entries as shown in 7.

$$\begin{cases} \widetilde{Q^a}[z] + \widetilde{Q^b}[z] = \widetilde{Q}[z], & S[z] = 0 \\ \widetilde{Q^a}[z] = \widetilde{Q^b}[z] = \widetilde{Q}[z], & S[z] = 1. \end{cases} \tag{7}$$

- Step 5: Finally, with the secret key sk_q, u_q encrypts $\{\widetilde{Q^a}, \widetilde{Q^b}\}$ as $\{M^{-1}_{u_{q,1}} \widetilde{Q^a}, M^{-1}_{u_{q,2}} \widetilde{Q^b}\}$. In this case, the trapdoor of the keyword-based binary vector can be denoted as $\widetilde{T_Q} = \{M^{-1}_{u_{q,1}} \widetilde{Q^a}, M^{-1}_{u_{q,2}} \widetilde{Q^b}\}$. Subsequently, u_q submits the trapdoor $\widetilde{T_Q}$ to the crowd server along with his/her identity u_q.

6) PPTM.TrapTran$(rk_q, \widetilde{T_Q})$. Once receiving the trapdoor $\widetilde{T_Q}$ from the task requester u_q, the crowd server first finds the requester's re-encryption key rk_q, and then re-encrypts the trapdoor as shown in 8.

$$\begin{aligned} T_Q &= \{M'_{u_{q,1}} M^{-1}_{u_{q,1}} \widetilde{Q^a}, M'_{u_{q,2}} M^{-1}_{u_{q,2}} \widetilde{Q^b}\} \\ &= \{M_1^{-1} \widetilde{Q^a}, M_2^{-1} \widetilde{Q^b}\}. \end{aligned} \tag{8}$$

7) PPTM.Query$(\{C_{B_i}\}_{i=1}^{m'}, T_Q)$. With the transformative trapdoor T_Q, the crowd server executes the task matching operations for each transformative index C_{B_i} and checks whether the keyword-based binary vector matches the task requirement as follows.

- Step 1: The crowd server first calculates the inner product of T_Q and C_{B_i} as shown in 9.

$$\begin{aligned} C_{B_i} \circ T_Q &= \{M_1^T \widetilde{B_i^a}, M_2^T \widetilde{B_i^b}\} \circ \{M_1^{-1} \widetilde{Q^a}, M_2^{-1} \widetilde{Q^b}\} \\ &= \{M_1^T \widetilde{B_i^a}, M_2^T \widetilde{B_i^b}\}^T \{M_1^{-1} \widetilde{Q^a}, M_2^{-1} \widetilde{Q^b}\} \\ &= \widetilde{B_i^a}^T \widetilde{Q}^a + \widetilde{B_i^b}^T \widetilde{Q}^b \\ &= \widetilde{B_i} \circ \widetilde{Q}. \end{aligned} \tag{9}$$

- Step 2: After calculating the value of $C_{B_i} \circ T_Q$, the crowd server outputs $\Lambda_Q = 1$ if $C_{B_i} \circ T_Q \geq 0$, which means the corresponding interest is similar to the task requirement, and the crowd server adds C_{B_i} to the query result in $\mathcal{R}$; otherwise, the crowd server outputs 0. Meanwhile, the crowd server filters the collection $\mathcal{B}$ with the same prefix. Next, the crowd server chooses another transformative index and repeats the task matching process.
- Step 3: Then, the crowd server obtains the query result $\mathcal{R} = \{C_{B_i} | \text{PPTM.Query}(C_{B_i}, T_Q) = 1\}$. The query result $\mathcal{R}$ shows that the Jaccard similarity between the worker's interest and the task requirement is not less than θ. That is, if the inner product $C_{B_i} \circ T_Q \geq 0$, the vehicular worker u_i satisfies the interest constraint.
- Step 4: Finally, the crowd server acquires a series of vehicular workers $\{u_1, u_2, \ldots, u_t\}$ whose encrypted binary vectors satisfy PPTM.Query$(C_{B_i}, T_Q) = 1$ $(i \in [1, t])$, where t is the number of preselected workers based on the interest constraint, and $t < m$.

B. Location-Based PPTM

Although interest-based PPTM can find vehicular workers who have common interests, it still cannot identify vehicular workers in the acceptable distance to complete the task. As shown in Fig. 4(b), the location of the task requester may locate inside or outside the bounding area of vehicular workers. If the location of a task requester is inside the bounding area of vehicular workers, we should compare the Euclidean distances between the bounding points and the requester's location, and check whether the Euclidean distances are less than or equal to the threshold δ. Otherwise, we compare the Euclidean distances between the bounding points on the nearest side and the requester's location.

Obviously, we can utilize the same encryption mechanism proposed in the interest-based PPTM to protect the location privacy and achieve location-based task matching. Therefore, the key idea of location-based PPTM is to check whether the Euclidean distance between the vehicular worker's location and the requester's location is less than or equal to the threshold δ in the ciphertext.

In specific, we assume that the location L_w of a vehicular worker u_i is (x_w, y_w), and the location L_q of the task requester u_q is (x_q, y_q). Then, u_i extends L_w to a $(3n+4)$-dimensional vector $\widehat{L_w} = (r_l x_w, r_l y_w, -r_l(x_w^2 + y_w^2), r_l, r_l, 0, \ldots, 0)$, where r_l is an one-time random positive number. Accordingly, u_q extends L_q to a $(3n+4)$-dimensional vector $\widehat{L_q} = (2r'_l x_q, 2r'_l y_q, r'_l, -r'_l(x_q^2 + y_q^2), r'_l \delta^2, 0, \ldots, 0)$, where r'_l is also an one-time random positive number. Then, by using the same permutation π, the extended vectors $\widehat{L_w}$ and $\widehat{L_q}$ can be permuted as $\widetilde{L_w}$ and $\widetilde{L_q}$, respectively. After that, by executing the same operations of PPTM.IndexEnc, PPTM.IndexTran and PPTM.TrapGen, PPTM.TrapTran, the crowd server obtains the ciphertexts of $\widehat{L_w}$ and $\widehat{L_q}$ as follows.

$$C_{L_w} = \{M_1^T \widetilde{L_w^a}, M_2^T \widetilde{L_w^b}\}, \tag{10}$$

$$C_{L_q} = \{M_1^{-1} \widetilde{L_q^a}, M_2^{-1} \widetilde{L_q^b}\}. \tag{11}$$

Similarly, as shown in 9, we have $C_{L_w} \circ C_{L_q} = \widetilde{L_w} \circ \widetilde{L_q}$. Since $\widetilde{L_w}$ and $\widetilde{L_q}$ are permuted by the same permutation π, we have $\widetilde{L_w} \circ \widetilde{L_q} = \widehat{L_w} \circ \widehat{L_q}$. Then, the equation $C_{L_w} \circ C_{L_q} = \widehat{L_w} \circ \widehat{L_q}$ holds. Therefore, according to the inner product of the encrypted vectors C_{L_w} and C_{L_q}, the crowd server can check whether the Euclidean distance between the locations of u_i and u_q is less than or equal to δ as shown in 12.

$$\begin{aligned} C_{L_w} \circ C_{L_q} &= \widehat{L_w} \circ \widehat{L_q} \\ &= r_l r'_l (2x_w x_q + 2y_w y_q - x_w^2 \\ &\quad - x_q^2 - y_w^2 - y_q^2 + \delta^2) \\ &= r_l r'_l \left(\delta^2 - d^2(L_w, L_q)\right). \end{aligned} \tag{12}$$

As a result, the crowd server can check whether u_i's location is acceptable by calculating $C_{L_w} \circ C_{L_q}$. Due to $r_l r'_l > 0$, if $C_{L_w} \circ C_{L_q} \geq 0$, then $d^2(L_w, L_q) \leq \delta^2$, the crowd server adds the vehicular worker's identity u_i to the final result and allocates the task to u_i who simultaneously satisfies the interest constraint and the location constraint; otherwise, the crowd server chooses another worker in $\mathcal{R}$ and repeats the above processes.

C. Correctness Analysis

Here, we analyze the correctness of both interest-based PPTM and location-based PPTM. Based on the Definition 3, we have the following theorem about the correctness of interest-based PPTM.

Theorem 1 (Correctness of Interest-based PPTM): Assume that $\overline{B_i} \circ Q + |Q|^2 + |B_{i,n_1}|^2 > B'_i \circ Q'$ holds, for the interest-based PPTM, PPTM.Query$(C_{B_i}, T_Q) \rightarrow \Lambda_Q = 1$ is correct if and only if $C_{B_i} \circ T_Q \geq 0$.

Proof: Since both u_i and u_q use the same permutation π to permute $\widehat{B_i}$ and $\widehat{Q}$, we have $\widetilde{B_i} \circ \widetilde{Q} = \widehat{B_i} \circ \widehat{Q}$. Based on the 9,

we have

$$\begin{aligned}
&C_{B_i} \circ T_Q \\
&= \widetilde{B_i} \circ \widetilde{Q} \\
&= \widehat{B_i} \circ \widehat{Q} \\
&= (1+\theta)\alpha\beta B_i' \circ Q' - \theta\alpha\beta\overline{B_i} \circ Q - \theta\alpha\beta|Q|^2 - \theta\alpha\beta|B_{i,n_1}|^2 \\
&= \alpha\beta\left((1+\theta)B_i' \circ Q' - \theta\overline{B_i} \circ Q - \theta|Q|^2 - \theta|B_{i,n_1}|^2\right).
\end{aligned}$$

Due to $\alpha > 0$ and $\beta > 0$, we have the following statement. If $C_{B_i} \circ T_Q \geq 0$, then $(1+\theta)B_i' \circ Q' - \theta\overline{B_i} \circ Q - \theta|Q|^2 - \theta|B_{i,n_1}|^2 \geq 0$. Since the assumption $\overline{B_i} \circ Q + |Q|^2 + |B_{i,n_1}|^2 > B_i' \circ Q'$ holds, we have

$$\frac{B_i' \circ Q'}{\overline{B_i} \circ Q + |Q|^2 + |B_{i,n_1}|^2 - B_i' \circ Q'} \geq \theta. \tag{13}$$

Considering that on the one hand

$$\begin{aligned}
&B_i' \circ Q' \\
&= \sum_{j=1}^{2n} b_{i,j}' q_j' \\
&= \sum_{j=1}^{n_1} (b_{i,2j-1}' q_{2j-1}' + b_{i,2j}' q_{2j}') + \sum_{j=n_1+1}^{n} (b_{i,2j-1}' q_{2j-1}' + b_{i,2j}' q_{2j}') \\
&= \sum_{j=1}^{n_1} b_{i,j} q_j + \sum_{j=n_1+1}^{n} q_j q_j \\
&= (b_{i,1}, \ldots, b_{i,n_1}, q_{n_1+1}, \ldots, q_n) \circ (q_1, \ldots, q_{n_1}, q_{n_1+1}, \ldots, q_n) \\
&= (B_{i,n_1}, Q_{n_2}) \circ Q \\
&= \widetilde{B}_{i,max} \circ Q.
\end{aligned}$$

On the other hand

$$\begin{aligned}
&\overline{B_i} \circ Q + |Q|^2 + |B_{i,n_1}|^2 \\
&= (0, \ldots, 0, q_{n_1+1}, \ldots, q_n) \circ (q_1, q_2, \ldots, q_n) + |Q|^2 + |B_{i,n_1}^2| \\
&= |B_{i,n_1}|^2 + |Q_{n_2}|^2 + |Q|^2 \\
&= |\widetilde{B_{i,max}}|^2 + |Q|^2.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
&\frac{B_i' \circ Q'}{\overline{B_i} \circ Q + |Q|^2 + |B_{i,n_1}|^2 - B_i' \circ Q'} \\
&= \frac{\widetilde{B}_{i,max} \circ Q}{|\widetilde{B_{i,max}}|^2 + |Q|^2 - \widetilde{B}_{i,max} \circ Q} \\
&= J(\widetilde{B}_{i,max}, Q).
\end{aligned} \tag{14}$$

Based on the 13 and 14, we have $J(\widetilde{B}_{i,max}, Q) \geq \theta$. According to the Definition 3, $J(\widetilde{B}_{i,max}, Q)$ is the maximal Jaccard similarity between Q and the binary vectors in the collection $\mathcal{B}$. If $J(\widetilde{B}_{i,max}, Q) \geq \theta$, there exists at least one binary vector $\widetilde{B} \in \mathcal{B}$ such that $J(\widetilde{B}, Q) \geq \theta$, which means $\Lambda_Q = 1$ holds. Therefore, the correctness of interest-based PPTM is demonstrated.

Similarly, we have the following theorem about the correctness of location-based PPTM.

Theorem 2 (Correctness of Location-based PPTM): Given the encrypted locations C_{L_w} and C_{L_q} of the vehicular worker u_i and the task requester u_q, the algorithm $\mathsf{PPTM.Query}(C_{L_w}, C_{L_q}) \to \Lambda_q = 1$ is correct if and only if $C_{L_w} \circ C_{L_q} \geq 0$.

Proof: Note that location-based PPTM also designed by using STPE, we have

$$\Lambda_q = \begin{cases} 1, & d(L_w, L_q) \leq \delta \\ 0, & otherwise. \end{cases} \tag{15}$$

As shown in 12, we have

$$C_{L_w} \circ C_{L_q} = r_l r_l' \left(\delta^2 - d^2(L_w, L_q)\right).$$

Since r_l and r_l' are one-time positive numbers, if $C_{L_w} \circ C_{L_q} \geq 0$, we have $d^2(L_w, L_q) \leq \delta^2$. Therefore, $\mathsf{PPTM.Query}(C_{L_w}, C_{L_q}) \to \Lambda_q = 1$ is correct if and only if $C_{L_w} \circ C_{L_q} \geq 0$.

VII. Security Analysis

In this section, we analyze the security of our PPTM concerning the threat model and security requirements mentioned in Section III.

1) Security of PPTM under the KPA model: In the KPA model, the crowd server observes several plaintext-ciphertext pairs and attempts to obtain the secret keys sk_i and sk_q to decrypt other ciphertexts.

Theorem 3: PPTM can protect index confidentiality and trapdoor confidentiality under the KPA model.

Proof: As we described in the threat model, in the CPA model, the crowd server owns more information than that in the KPA model. That is, if PPTM is secure under the CPA model, it can also defend against KPA. Thus, we skip the security proof under the KPA model and focus on the security proof under the CPA model.

2) Security of PPTM under the CPA model: In the CPA model, the crowd server can select several plaintexts and obtain the corresponding ciphertexts, and has oracle access to the encryption process.

Theorem 4: PPTM can protect index confidentiality and trapdoor confidentiality under the CPA model.

Proof: 1) *Index confidentiality:* Considering that the index confidentiality in the CPA model, the crowd server can derive a number of encrypted values $\{\widetilde{B}_i\}_{i=1}^{m'}$ and the corresponding aggregated binary vector $B_i \in \mathcal{B}$. Without the splitting vector S, permutation π, and the aggregated metric, the crowd server cannot recover the secret key $sk_i = \{M_{u_{i,1}}, M_{u_{i,2}}\}$. On the one hand, in our $\mathsf{PPTM.IndexEnc}$, B_i will be extended to a $(3n+4)$-dimensional vector $\widehat{B_i} = (\alpha B_i', \alpha\overline{B_i}, \alpha, \alpha|B_{i,n_1}|^2, r_b, 0)$, where α, r_b are one-time random positive numbers. The security of one-time randomness is equivalent to the security of the one-time pad encryption, which can perform perfect security and ensure the ciphertext $\widetilde{C_{B_i}}$ be a random vector to the crowd server.

On the other hand, the crowd server cannot obtain the exact solution to address the linear equations created by the vehicular worker as $\widetilde{C^a_{B_i}} = M_1^T \widetilde{B^a_i}$ and $\widetilde{C^b_{B_i}} = M_2^T \widetilde{B^b_i}$, where both M_1 and M_2 are unknown to the crowd server, and the secret key $sk_i = \{M_{u_{i,1}}, M_{u_{i,2}}\}$ is different for each workers. Theoretically speaking, there are $2(3n+4)$ unknowns in $\widetilde{B^a_i}$ and $\widetilde{B^b_i}$, and $2(3n+4)^2$ unknowns in M_1 and M_2. However, it only exists $2(3n+4)$ equations that are far less than the number of unknowns. Hence, the crowd server cannot launch a linear analysis attack with the gained information in the CPA model.

Since the construction of location-based PPTM is similar to the interest-based PPTM, the crowd server only knows whether the inner product of C_{L_w} and C_{L_q} is negative or positive, which indicates that the Euclidean distance between the vehicular worker and the task requester is greater than or within the threshold δ, but has no idea about the key information about L_w, L_q or $L_w \circ L_q$. Therefore, the confidentiality of locations also can be protected.

2) *Trapdoor confidentiality:* Note that the trapdoor generation in the PPTM.TrapGen is similar to the index construction in the PPTM.IndexEnc, it can also protect the confidentiality of the trapdoor with the property of both learning with error and randomness under the CPA model. Since the binary vector Q is extended to $\widehat{Q} = ((1+\theta)\beta Q', -\theta\beta Q, -\theta\beta|Q|^2, -\theta\beta, 0, r_q)$, where β and r_q are one-time random numbers, the ciphertext $\widetilde{T_Q}$ is a random vector to the crowd server. In addition, the trapdoor is concealed by the random permutation π and the splitting vector S. Without S and π, the private content of the trapdoor cannot be disclosed, i.e., the confidentiality of the trapdoor can be guaranteed. □

Theorem 5: PPTM can achieve index unlinkability and trapdoor unlinkability under the CPA model.

Proof: 1) *Index unlinkability:* Suppose binary vectors $B_i, B_j \in \mathcal{B}$ are two interest representations, and B_i and B_j have different prefixes and suffixes, we have $Pr[B_i = B_j] = \frac{1}{C_n^{n_1} C_n^{n_2}}$. Note that the randomized indexes constructed in the PPTM.IndexEnc can be realized in three aspects: (i) One-time randomness; (ii) Random permutation; (iii) Random segmentation.

(i) One-time randomness: Given two binary vector $B_0, B_1 \in \mathcal{B}$, where $\mathcal{B} = \{\widetilde{B} | \widetilde{B} \in \{0,1\}^n, \widetilde{B}_{n_1} = B_{n_1}, \widetilde{B}_{n_2} \in \{*\}^{n_2}\}$. As shown in step 3 in the `PPTM.IndexEnc`, two binary vectors B_0 and B_1 are extended to $\widehat{B_0} = (\alpha_0 B_0', \alpha_0 \overline{B_0}, \alpha_0, \alpha_0 |B_{0,n_1}|^2, r_{b,0}, 0)$ and $\widehat{B_1} = (\alpha_1 B_1', \alpha_1 \overline{B_1}, \alpha_1, \alpha_1 |B_{1,n_1}|^2, r_{b,1}, 0)$, respectively. Since α_0, α_1, $r_{b,0}$, and $r_{b,1}$ are different one-time random numbers, the index encryption function is randomized to the crowd server. That is, Given two binary vectors B_0, B_1 chosen by the crowd server, the ciphertexts of B_0 and B_1 are random to the crowd server.

(ii) Random permutation: As show in step 4 in the PPTM.IndexEnc, by using the permutation π, the extended vectors are permuted as $\widetilde{B_0} = \pi(\widehat{B_0})$ and $\widetilde{B_1} = \pi(\widehat{B_1})$. Loosely speaking, the random permutation π can generate a new arrangement of the elements in $\widetilde{B_0}$ and $\widetilde{B_1}$, which can randomly change the original orders of the extended vectors. Besides, B_0 and B_1 are two aggregated binary vectors. Without knowing the aggregated manner, the crowd server cannot infer whether two workers' interests are same or contain the same keywords.

TABLE III
NOTATIONS FOR COMPARISON ANALYSIS

Notations	Descriptions
n	size of keyword dictionary $\mathcal{W}$
m	number of vehicular workers
n_i	number of keywords in vehicular worker's interests
l	number of keywords in task requirement
d	maximal number of keywords in task requirement, $d \geq l$
T_{h_s}	time cost for a keyword-based hash function h_s
T_{e_z}	time cost for each exponentiation operation on Z_p
T_e	time cost for each exponentiation operation on G
T_p	time cost for each pairing on group G
T_H	time cost for a hash function on G
T_h	time cost for a hash function on Z_k
T_{Enc}	time cost for a symmetric encryption
T_{Dec}	time cost for a symmetric decryption
T_V	time cost for each inner product operation
T_M	time cost for each matrix-matrix multiplication

(iii) Random segmentation: As show in 4, when the value of $S[z]$ is equal to 1, both $\widetilde{B_0}[z]$ and $\widetilde{B_1}[z]$ are randomly split into two values with the sum of these two random values being the original value, i.e., $\widetilde{B^a_0}[z] + \widetilde{B^b_0}[z] = \widetilde{B_0}[z]$ and $\widetilde{B^a_1}[z] + \widetilde{B^b_1}[z] = \widetilde{B_1}[z]$. Suppose $Pr[\{\widetilde{B^a_0}[z], \widetilde{B^b_0}[z]\} = \{\widetilde{B^a_1}[z], \widetilde{B^b_1}[z]\}] = \gamma$ for any dimension z where $S[z] = 1$, we have $Pr[\{\widetilde{B^a_0}, \widetilde{B^b_0}\} = \{\widetilde{B^a_1}, \widetilde{B^b_1}\}] = \gamma^{s_0}$, where s_0 is the number of "1" in the bit vector S. Since $\gamma \to 0$, $s_0 \to 3n+4$, and both C_{B_0} and C_{B_1} are constructed by using the algorithm of PPTM.IndexEnc and PPTM.IndexTran with the same binary vector (i.e., B_0 and B_1 have the same prefix and suffix), then we have

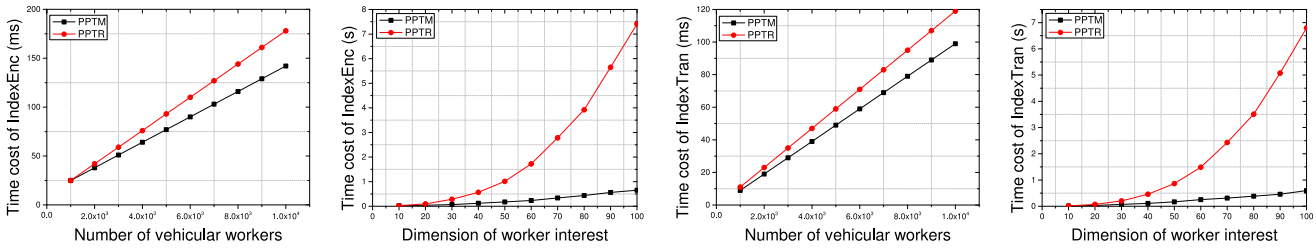
$$Pr[C_{B_0} = C_{B_1}] = \frac{1}{C_n^{n_1} C_n^{n_2}} \cdot \gamma^{s_0} \to 0.$$

Therefore, given any two plaintexts B_0 and B_1 chosen by the crowd server, the ciphertexts of B_0 and B_1 are random to the crowd server. Namely, the crowd server cannot distinguish which binary vector is actually encrypted. Then, the index unlinkability under the CPA model is demonstrated.

2) *Trapdoor unlinkability:* Similarly, the trapdoor unlinkability also guaranteed by three aspects: (i) One-time randomness; (ii) Random permutation; (iii) Random segmentation. Since the effect of one-time randomness and random permutation in PPTM.TrapGen are similar to the effect in PPTM.IndexEnc, we skip the analysis here, and mainly focus on analyzing the random segmentation in PPTM.TrapGen. Given two requirement binary vectors Q_0 and Q_1, as shown in 7, when $S[z]$ is equal to 0, both $\widetilde{Q_0}[z]$ and $\widetilde{Q_1}[z]$ are randomly split into two values, i.e., $\widetilde{Q^a_0}[z] + \widetilde{Q^b_0}[z] = \widetilde{Q_0}[z]$ and $\widetilde{Q^a_1}[z] + \widetilde{Q^b_1}[z] = \widetilde{Q_1}[z]$. Suppose $Pr[\{\widetilde{Q^a_0}[z], \widetilde{Q^b_0}[z]\} = \{\widetilde{Q^a_1}[z], \widetilde{Q^b_1}[z]\}] = \eta$ for any dimension z where $S[z] = 0$, we have $Pr[\{\widetilde{Q^a_0}, \widetilde{Q^b_0}\} = \{\widetilde{Q^a_1}, \widetilde{Q^b_1}\}] = \eta^{s_1}$, where s_1 is the number of "0" in the bit vector S. Since $\eta \to 0$ and $s_1 \to 3n+4$, we have $Pr[T_{Q_0} = T_{Q_1}] \to 0$. Hence, given any

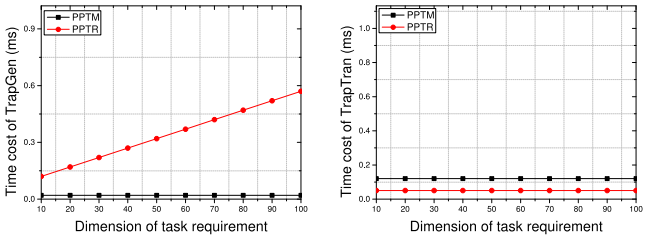
TABLE IV
COMPARISON OF COMPUTATIONAL COMPLEXITY

Scheme	IndexEnc	IndexTran	TrapGen	TrapTran	Query
MSDE [11]	$2(n_i+1)\mathsf{T}_e+n_i\mathsf{T}_{h_s}+\mathsf{T}_h$	$n_i\mathsf{T}_e$	$(3+2l)\mathsf{T}_e+l\mathsf{T}_{h_s}$	$l\mathsf{T}_e$	$l\mathsf{T}_e+n_il\mathsf{T}_{h_s}$
MuED [14]	$2n_i\mathsf{T}_e+n_i\mathsf{T}_H+n_i\mathsf{T}_h+n_i\mathsf{T}_{Enc}$	n_iT_p	$l\mathsf{T}_e+l\mathsf{T}_H$	$l\mathsf{T}_p$	$l\mathsf{T}_h+n_il\mathsf{T}_{Dec}$
PPTR [9]	$n_i\mathsf{T}_{h_s}+n_i(d+1)\mathsf{T}_{e_z}+2n_i\mathsf{T}_M$	$2n_iT_M$	$(d+1)\mathsf{T}_{h_s}+(\sum_{j=1}^{d}C_d^j(j-1)+2(d+1))\mathsf{T}_V$	$2(d+1)\mathsf{T}_V$	$2(d+1)\mathsf{T}_V$
PPTM	$2n\mathsf{T}_V$	$2n\mathsf{T}_V$	$2n\mathsf{T}_V$	$2n\mathsf{T}_V$	$2\mathsf{T}_V$



(a) (b) (c) (d)

Fig. 5. Experimental results of Our PPTM and PPTR [9]. (a), (b) The time cost of `IndexEnc`. (c), (d) The time cost of `IndexTran`. (a) n=10. (b) m=1000. (c) n=10. (d) m=1000.



(a) (b)

Fig. 6. (a) The time cost of `TrapGen`. (b) The time cost of `TrapTran`. (a) m=1000. (b) m=1000.

two requirement vectors Q_0 and Q_1 chosen by the crowd server, the trapdoors of Q_0 and Q_1 are random to the crowd server. That is, the crowd server cannot distinguish which task requirement is actually encrypted. Therefore, the trapdoor unlinkability is demonstrated. □

VIII. PERFORMANCE ANALYSIS AND EXPERIMENTAL EVALUATION

In this section, we analyze the computational complexity of PPTM and conduct experiments to evaluate the performance of PPTM.

A. Performance Analysis

The notations used in performance analysis can be referred in TABLE III. Since the location-based PPTM and the interest-based PPTM have the same algorithms, the computational cost of PPTM is twice as much as interest-based PPTM. Therefore, we only need to focus on the computational cost of interest-based PPTM. In the PPTM.IndexEnc phase, a vehicular worker needs to perform two multiplications between a vector and a matrix (i.e. matrix-vector multiplication), and two matrix-vector multiplications in the PPTM.IndexTran, which for both of them cost $2n_i\mathsf{T}_V$ as show in step 6 of PPTM.IndexEnc and 5, where the time taken to perform one matrix-vector multiplication is equal to nT_V. In the PPTM.TrapGen phase, to generate a search trapdoor, a task requester needs to perform two matrix-vector multiplications, which cost $2n\mathsf{T}_V$. Similarly, in the PPTM.TrapTran phase, the crowd server also needs to perform two matrix-vector multiplications as shown in 8, and the time cost of PPTM.TrapTran is also $2n\mathsf{T}_V$. In the phase of PPTM.Query, the crowd server needs to perform two inner products of the transformative indexes (i.e., $M_1^T\widetilde{B_i^a}$ and $M_2^T\widetilde{B_i^b}$) and the transformative trapdoors (i.e., $M_1^{-1}\widetilde{Q^a}$ and $M_2^{-1}\widetilde{Q^b}$) as shown in 9. Hence, the time cost of PPTM.Query is $2\mathsf{T}_V$.

As shown in TABLE IV, we compare the computational complexity of PPTM with the existing schemes. Note that both MSDE [11] and MuED [14] design their protocols based on the pairing and exponentiation operations, which are time-consuming. Besides, PPTR [9] also performs key-based hash operation to protect the confidentiality of keywords, and maps each worker's keywords into an $n_i \times (d+1)$ matrix, which also increases the computational cost in the index encryption phase and task matching phase. Because it should execute matrix-matrix multiplication for encrypting worker's interests and matrix-vector multiplication for task-worker matching. While our PPTM utilizes the binary vectors to represent workers' interests, which only needs to perform matrix-vector multiplication in the index encryption phase and vector-vector inner product operation in the task matching phase. In addition, we utilize the aggregation solution to reduce the number of binary vectors, which also can improve the query performance. What's more, PPTM performs the whole algorithm without any pairing, exponentiation, or hash operations.

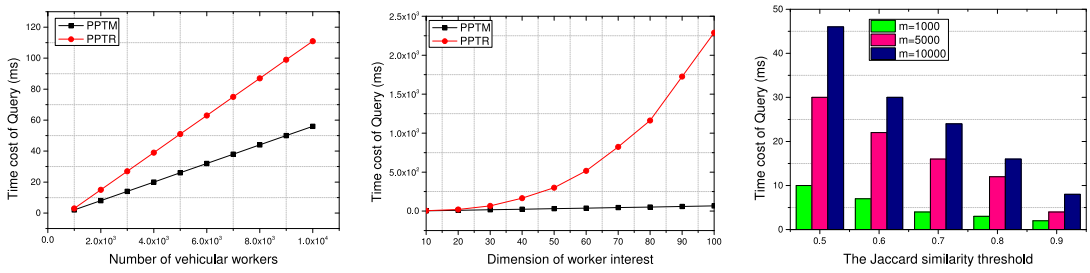


Fig. 7. Experimental results of our PPTM and PPTR [9]. (a), (b) The time cost of `Query` with different number of vehicular workers and different dimension of worker's interest, respectively. (c) The time cost of `Query` in PPTM with different Jaccard similarity threshold. (a) n=10. (b) m=1000. (c) n=10.

B. Experimental Evaluation

To demonstrate the performance of PPTM, we compare it with Privacy-Preserving Task Recommendation (PPTR) [9], which is also a multi-keyword searchable encryption scheme in the multi-worker multi-requester setting.

1) Dataset and Setup: We conduct experimental simulations in a synthetic dataset and performance evaluations in a real-world dataset to test the efficiency of PPTM. The real-world dataset downloads from MTurk Track [36], which contains large scale of Human Intelligence Tasks (HITs). In this dataset, 10000 different HITs are selected to be our tasks to evaluate the performance of PPTM, and the keywords of each HIT can be viewed as task requirements. In the synthetic dataset, there are 10000 vehicular workers, and the size of keyword dictionary is 100, i.e., $m = 10000$ and $n = 100$. The experiments are conducted on a desktop with 32.00 GB of RAM and an Intel Core E3-1225 v5, CPU @ 3.30 GHz. All of the algorithms are implemented in JAVA.

2) Experimental Results: We evaluate the performance of PPTM, and compare it with PPTR [9]. Since both `PPTM.Setup` and `PPTM.KeyGen` are one-time operations during task matching process, we mainly focus on the execution time cost of `PPTM.IndexEnc`, `PPTM.IndexTran`, `PPTM.TrapGen`, `PPTM.TrapTran`, and `PPTM.Query`.

Index encryption. In Fig. 5(a), we plot the time cost of `IndexEnc` of our PPTM in the different numbers of vehicular workers where the dimension of the worker's interest $n = 10$, and compare it with PPTR. The `IndexEnc` of PPTM performs better than that of PPTR, and both of them are linear with the number of vehicular workers because the dimension of worker's interest is 10, which is a quite low-dimensional operation for executing index encryption algorithm. From Fig. 5(b), we can see that the computational cost of encrypting indexes in PPTM is far less than PPTR. The reason is that the vehicular worker in PPTM needs to perform matrix-vector multiplication in `IndexEnc` phase, while each index is encrypted by matrix-matrix multiplication in PPTR. The running time of `IndexEnc` in PPTM increases slowly when the dimension of the worker's interest varies from 10 to 100. Specifically, when the dimension of the worker's interest is 100, the running time of PPTM only needs 0.65 s for index encryption, which is at least 11× faster than that of PPTR.

Index transformation. We also evaluate the performance of index transformation of both PPTM and PPTR. From Fig. 5(c) , similar to the index encryption, the time costs of `IndexTran` in two schemes increase linearly with the number of vehicular workers when the dimension of worker's interest fixes on 10. As shown in Fig. 5(d), index transformation of PPTM is much more efficient than that of PPTR, which takes 6.79 s to transform each index when the dimension of the worker's interest is 100.

Trapdoor generation. Fig. 6(a) plots the time cost of `TrapGen` in different dimensions of task requirement with the number of vehicular workers fixes on 1000. We can observe that the computational cost of `TrapGen` in PPTR increases linearly with the dimension of task requirement, while the time cost of PPTM tends to be stable, because each trapdoor generation process only needs to execute one-time matrix-vector multiplication in PPTM, and PPTR requires additional keyword-based hash operations in PPTR.

Trapdoor transformation. In Fig. 6(b), we also draw the running time of `TrapTran` with the different dimensions of task requirement. It shows that both time costs of PPTM and PPTR are minimally affected by the dimension of task requirement and quite close to each other, because each trapdoor transformation process also performs one-time matrix-vector multiplication in both PPTM and PPTR as shown in 8.

Query. Fig. 7 shows the computational complexity of PPTM and PPTR with two impacts in the query phase, i.e., the number of vehicular workers m and the dimension of the worker's interest n. In addition, we evaluate the query performance of PPTM with different Jaccard similarity threshold θ. As shown in Fig. 7(a), the time cost of `Query` increases tardily with the number of vehicular workers in PPTM, and that of PPTR increases linearly. The running time of `Query` in PPTM over 10000 vehicular workers is 56 ms, and the time cost of PPTR requires 111 ms, which is about 2× larger than that of PPTM. Since the aggregation method can significantly reduce the number of vehicular workers, even the dimension of task interest is lower to 10, the query performance of PPTM is more efficient than that of PPTR. Fig. 7(b) demonstrates that the time cost of `Query` in PPTM increases at a slowing pace with different dimensions of worker's interest when the number of vehicular workers fixes on 1000. The query time of PPTR increases rapidly, which reaches up to 2,289 ms when the dimension of the worker's interest

is 100. The reason is that PPTR executes two matrix-vector multiplications for task matching, while PPTM performs two vector-vector multiplications. When the dimension of worker's interest in PPTR exceeds 80, the computational complexity of PPTR is exponential growth. Meanwhile, Fig. 7(c) plots the time cost of `PPTM.Query` on the MTurk Track dataset varying with m and θ. It shows that the time cost of `PPTM.Query` decreases as the Jaccard similarity threshold increases. For example, the time cost of `PPTM.Query` over the real-world dataset is only 8 ms when $\theta = 0.9$ and $m = 10000$, and the time cost of query is about 46 ms when $\theta = 0.5$ and $m = 10000$. For the Jaccard similarity threshold θ, the number of vehicular workers who are suitable for the task requirement can be pruned when θ increases.

IX. Conclusion

In this paper, we have proposed an efficient and privacy-preserving task matching (PPTM) scheme with threshold similarity search through vehicular crowdsourcing. For the interest matching, we have exploited binary vectors to represent task requirements and vehicular worker's interest and designed an interest-based PPTM with Jaccard similarity search. Meanwhile, to achieve spatial crowdsourcing, we have presented a location-based PPTM with Euclidean distance similarity search. By leveraging the techniques of matrix decomposition and proxy re-encryption, PPTM supports multi-keyword search in the multi-requester multi-worker setting. Extensive experiments show that PPTM achieves efficient task matching in the spatial interest-based vehicular crowdsourcing applications. For the future work, we will design a dynamic task matching scheme with the spatiotemporal information of both workers and tasks by utilizing vehicular crowdsourcing.

References

[1] J. Howe, "The rise of crowdsourcing," *Wired Mag.*, vol. 14, no. 6, pp. 1–4, 2006.

[2] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[3] M. Li, J. Weng, A. Yang, J. Liu, and X. Lin, "Toward blockchain-based fair and anonymous ad dissemination in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 248–11259, Nov. 2019.

[4] J. Ni, X. Lin, K. Zhang, and X. Shen, "Privacy-preserving real-time navigation system using vehicular crowdsourcing," in *Proc. IEEE 84th Veh. Technol. Conf.*, 2016, pp. 1–5.

[5] M. C. Yuen, I. King, and K. S. Leung, "Task recommendation in crowdsourcing systems," in *Proc. Crowdsourcing Data Mining*, 2012, pp. 22–26.

[6] D. E. Difallah, G. Demartini, and P. Cudré-Mauroux, "Pick-a-crowd: Tell me what you like, and i'll tell you what to do," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 367–374.

[7] Y. Gong, Y. Guo, and Y. Fang, "A privacy-preserving task recommendation framework for mobile crowdsourcing," in *Proc. IEEE Glob. Commun. Conf.*, 2014, pp. 588–593.

[8] J. Shu, X. Liu, X. Jia, K. Yang, and R. H. Deng, "Anonymous privacy-preserving task matching in crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3068–3078, Aug. 2018.

[9] J. Shu, X. Jia, K. Yang, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Trans. Serv. Comput.*, vol. 14, no. 1, pp. 235–247, Jan./Feb. 2021.

[10] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.

[11] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Secur.*, vol. 19, no. 3, pp. 367–397, 2011.

[12] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C. N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 813–823, Sep./Oct. 2019.

[13] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distributed Comput. Syst.*, 2010, pp. 253–262.

[14] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Secur. Practice Experience*, 2008, pp. 71–85.

[15] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[16] J. Shu and X. Jia, "Secure task recommendation in crowdsourcing," in *Proc. IEEE Glob. Commun. Conf.*, 2016, pp. 1–6.

[17] J. Shu, K. Yang, X. Jia, X. Liu, C. Wang, and R. Deng, "Proxy-free privacy-preserving task matching with efficient revocation in crowdsourcing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 117–130, Jan./Feb. 2021.

[18] X. Wang *et al.*, "Search me in the dark: Privacy-preserving boolean range query over encrypted spatial data," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2020, pp. 2253–2262.

[19] S. Su, Y. Teng, X. Cheng, K. Xiao, and G. Li, "Privacy-preserving top-k spatial keyword queries in untrusted cloud environments," *IEEE Trans. Serv. Comput.*, vol. 11, no. 5, pp. 796–809, Sep./Oct. 2018.

[20] D. Liu, J. Ni, X. Lin, and X. Shen, "Transparent and accountable vehicular local advertising with practical blockchain designs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15 694–15 705, Dec. 2020.

[21] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016.

[22] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (kase) for group data sharing via cloud storage," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2374–2385, Aug. 2016.

[23] Q. Tang, "Nothing is for free: Security in searching shared and encrypted data," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 11, pp. 1943–1952, Nov. 2014.

[24] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1317–1331, Jun. 2020.

[25] A. Yassine, M. S. Hossain, G. Muhammad, and M. Guizani, "Cloudlet-based intelligent auctioning agents for truthful autonomous electric vehicles energy crowdsourcing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5457–5466, May 2020.

[26] K. Suto, H. Nishiyama, and N. Kato, "Postdisaster user location maneuvering method for improving the qoe guaranteed service time in energy harvesting small cell networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9410–9420, Oct. 2017.

[27] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *Proc. VLDB Endowment*, vol. 7, no. 10, pp. 919–930, 2014.

[28] C. Huang, D. Liu, J. Ni, R. Lu, and X. Shen, "Achieving accountable and efficient data sharing in industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 1416–1427, Feb. 2021.

[29] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 506–522.

[30] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–5.

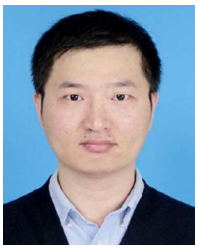
[31] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif, "Reliable and privacy-preserving truth discovery for mobile crowdsensing systems," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1245–1260, May/Jun. 2021.

[32] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan, "An efficient framework for exact set similarity search using tree structure indexes," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 759–770.

[33] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Management Data*, 2009, pp. 139–152.

[34] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving efficient and privacy-preserving exact set similarity search over encrypted data," *IEEE Trans. Dependable Secure Comput.*, to be published, doi:10.1109/TDSC.2020.3004442.

[35] K. Zhou and J. Ren, "Passbio: Privacy-preserving user-centric biometric authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 12, pp. 3050–3063, Dec. 2018.
[36] D. E. Difallah, M. Catasta, G. Demartini, P. G. Ipeirotis, and P. Cudré-Mauroux, "The dynamics of micro-task crowdsourcing: The case of amazon mturk," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 238–247.



Fuyuan Song received the B.S. degree from the College of Mathematics and Information Science, Jiangxi Normal University, Nanchang, China, in 2014. He is currently working toward the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. In 2019, he joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, as a Visiting Scholar. His research interests include data security and applied cryptography.

Zheng Qin (Member, IEEE) received the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in 2001. From 2010 to 2011, he was a Visiting Scholar with the Department of Computer Science, Michigan University, Ann Arbor, MI, USA. He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include machine learning, applied cryptography, and cloud computing.

Dongxiao Liu (Member, IEEE) received the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2020. He is currently a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests include mobile computing and blockchain.

Jixin Zhang received the M.S. degree in computer science and technology from the Wuhan University of Technology, Wuhan, China and the Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2014 and 2019, respectively. He is currently an Assistant Professor of computer science with the Hubei University of Technology, Wuhan, China. His primary research interests include artificial intelligence and security.

Xiaodong Lin (Fellow, IEEE) received the Ph.D. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada. He is currently an Associate Professor with the School of Computer Science, University of Guelph, Guelph, ON, Canada. His research interests include wireless communications and network security, computer forensics, software security, and applied cryptography. He was the recipient of the Outstanding Achievement in Graduate Studies Award for the Ph.D. degree.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He was the recipient of the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He was also the recipient of the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Awardin 2003 from the Province of Ontario, Canada. He was the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the President Elect of the IEEE Communications Society. He was the Vice President for Technical and Educational Activities, the Vice President for Publications, Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a Member of IEEE Fellow Selection Committee of the ComSoc. He was the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*.