Received August 3, 2020, accepted August 18, 2020, date of publication August 25, 2020, date of current version September 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3018940

A Secure and Efficient Task Matching Scheme for Spatial Crowdsourcing

FULIN ZHOU, JUNYI LI, YAPING LIN, (Member, IEEE), JIANHAO WEI, (Graduate Student Member, IEEE), AND VOUNDI KOE ARTHUR SANDOR

College of Computer Science and Electronics Engineering, Hunan University, Changsha 410082, China
Hunan Provincial Key Laboratory of Dependable Systems and Networks, Hunan University, Changsha 410082, China

Corresponding author: Junyi Li (junyilee@hnu.edu.cn)

ABSTRACT The sharing economy has greatly promoted the rapid development and application of spatial crowdsourcing. Although privacy-preserving spatial task matching as an indispensable part has been extensively explored, existing schemes cannot be deployed into the practical environment due to drawbacks in the one-side location protection, the matching efficiency, and the dynamic updates. In this study, we propose a novel Secure and Efficient Spatial Task Matching framework (SESTM) with utilizing multi-user searchable encryption and secure index technique, which enables to preserve the location privacy of requesters and workers while achieving efficient task allocation and good user scalability. Specifically, requesters firstly transform and encrypt their task locations before being outsourced, and we secondly design a secure and dynamic tree-based index SD-Tree for SC-server to merge these uploaded encrypted data without knowing their underlying content. Finally, SESTM provides efficient task matching services for multiple workers based on encrypted queries. Furthermore, SD-Tree also provides fast delete and insert operations under logarithmic time to reduce the dynamic update overhead for real SC services. Extensive theoretical analysis and performance evaluation demonstrate the practicality of our method.

INDEX TERMS Spatial crowdsourcing, task matching, location privacy, matching efficiency, dynamic update, user scalability.

I. INTRODUCTION

The widespread popularization of 4G networks and the rapid deployment of 5G networks have actively promoted the diversified application of spatial crowdsourcing (SC) [1], [2]. Generally, in SC service, the SC-server receives location-based tasks published by requesters and matches appropriate tasks according to workers' work scopes. Then workers travel to the required place to complete the assigned task for monetary or other rewards. Nowadays, many enterprises have established various SC platforms around the world to provide convenient and shared services for people, such as Uber [3], Amazon Mechanical MTurk [4] and Witmart [5].

Despite the various benefits of SC services, outsourcing location information to SC-server has raised concerns about privacy disclosure. To realize accurate and efficient task matching, current solutions need to expose users' specific locations to the SC-server, such as home address in food delivery application, individual health status if requesters locate in hospitals or clinics [6], [7], or daily trajectory. However, SC-server as a third party cannot be completely trustworthy, it possibly sells users' private information to related companies for profit [8], or be compromised by hackers [9]. Besides, Scheck [10] also found that adversaries can monitor users by collecting their locations. Considering their safety, users may be reluctant to use SC services. In addition, task matching efficiency is also a non-neglectful problem since if it is too low, SC-platforms have no ability to handle massive task matching requests, and users may get a poor experience. Therefore, there is an urgent need to design a secure and efficient task matching scheme to address these problems.

Pournajaf *et al.* [11] and To *et al.* [12] proposed a privacy task allocation method with adopting differential privacy technique, whereby worker locations are processed by a trusted third party (TTP) server called cellular service provider (CSP) before being outsourced to SC-server.

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Hui Yeh.

However, the TTP assumption is not compelling since CSP may expose these vital data once it is compromised. Besides, the one-side protection may cause SC-server to easily obtain the exact locations of requesters and workers according to the allocation result. On the contrary, some searchable symmetric encryption (SSE) approaches [13]–[15] based on TPP-free are more practical, which allow a single user holding the key k_s to perform secure query over ciphertext domain. Unfortunately, these conventional solutions cannot be applied to spatial crowdsourcing service (multi-user setting) since there are multiple unassisted requesters and workers in the platform, and any user revocation will cause keys redistribution and data re-encryption if they share a same key [16]. On the other hand, the system is also vulnerable to malicious adversaries whenever the secret key is leaked [17], [18]. Therefore, Shu *et al.* [19] proposed a secure task matching approach with utilizing multi-user searchable encryption and proxy re-encryption technique, which allows each requester and worker holding a unique key to protect their location privacy and meanwhile realize effective user revocation. However, the authors failed to consider the task matching efficiency, thus the query time is increasing linearly with the number of tasks. Moreover in a previous study, Liu *et al.* [20] adopted Paillier Cryptosystem and KD-tree to build a secure index based on the dual-server setting to address the privacy-preserving and efficiency issues. Unfortunately, the huge update overhead makes this scheme unsuitable for the practical SC services since worker locations are dynamically changing rather than being static [21].

The drawbacks of the above research motivate us to design a novel solution that not only protects the location privacy of requesters and workers but also allows SC-server to efficiently execute the task matching services and the dynamic update operations. Inevitably, to achieve the above goals, there are three significant challenges:

1) Multi-user searchable encryption technique is a general and effective approach to achieve privacy task matching over the multi-user setting, but it is difficult for SC-server to build a secure index based on ciphertexts encrypted by different keys.
2) In a practical environment, the number of spatial tasks in the SC platform is dynamically changing, since the SC-server requires to frequently delete accepted tasks and insert newly published tasks. Thus, it is a challenge for the secure index to support dynamic updates.
3) The SC platform should allow users to freely enter or leave the system without causing huge update overhead to the system and affecting other users. Therefore, achieving efficient user enrollment and revocation is not a simple task.

To address the three challenges, we propose a Secure and Efficient Spatial Task Matching scheme (SESTM). We firstly adopt multi-user searchable encryption [17], [22] to preserve the location privacy, whereby the differences from previous works are that we improve query efficiency and achieve user revocation. Then, we design a tree-based index based on segment tree [13], [19], [23] to realize high matching efficiency, and the differences from previous schemes are that we reduce the query time complexity from $O(n)$ to $O(logn)$. Specifically, to address challenge 1), we turn the index construction issue into an index merging issue and design a novel secure and dynamic tree-based index (*SD-Tree*). Requesters and workers firstly utilize the segment tree to transform their respective task coordinates and queries, then encrypt them with their secret keys. Finally, the SC-server merges all transformed tasks into the SD-Tree and provides task matching services for transformed queries. In this way, we not only achieve the protection of location privacy but also improve task allocation efficiency. Moreover, the difference from previous works [13], [19], [23] is that they only utilized the segment tree to change the location information of requesters and workers into a set of labels for range query while failing to consider reducing the matching time. Regarding challenge 2), *SD-Tree* is essentially a binary tree with fixed tree height, whereby each tree branch represents a specific value. The update operation is to insert or delete tree branches from the *SD-Tree*, which is similar to the binary tree search process. Therefore, it is easy to demonstrate the time complexity of the update operations is under logarithmic time. For challenge 3), to realize the user enrollment and revocation, we adopt the proxy re-encryption technique based on Dong *et al.* [24], whereby each location needs to be encrypted by user and SC-server, respectively.

We summarize the contributions of our study in the following.

1) We propose a Secure and Efficient Spatial Task Matching scheme (SESTM) to address the location privacy-preserving issue for requesters and workers over one SC server. Simultaneously, our approach also achieves effective user enrollment and revocation.
2) We design a novel Secure and Dynamic Tree-based index *SD-Tree* to realize high task matching efficiency, which also executes the update operations under logarithmic time.
3) We implement the proposed solution on a real-world dataset. The results show our approach achieves good user scalability and dynamic updates while preserving location privacy. Furthermore, we also compare SESTM with one of the most relevant schemes [19], the results illustrate that our scheme has an apparent advantage over it in the aspect of task matching efficiency.

The rest of this paper is organized as follows. Section II reviews the related works and Section III introduces the problem. The preliminary is formulated in Section IV. Section V details the working mechanism of SESTM. Subsequently, Section VI present the performance analysis. Section VII evaluates our scheme SESTM. Finally, we conclude the paper in section VIII.

II. RELATED WORKS

In this section, we review the related works from two categories: privacy task matching and secure index.

A. PRIVACY TASK MATCHING

There have been extensive research on task matching in recent years, such as low-cost task allocation [25], [26], skills [27], [28] and interests [29] based assignments. These works treated the task matching problem as an optimization problem but ignored the location privacy problem. Considering the concern of privacy disclosure, some approaches were proposed to investigate the privacy task matching issue. To *et al.* [12] and Wang *et al.* [30] utilized a trusted third party (TTP) to obfuscate the worker locations with the adoption of differential privacy (DP) technique. In order to balance the privacy and raw data utility, Gong *et al.* [31] designed a framework with a trusted proxy to optimize the trade-offs. In addition, k-anonymity [32], [33] is also a general method, which utilizes a cloaking area containing at least k users to replace a specific user's location. However, there are two inevitable shortcomings in the above researches. Firstly, a potential hazard is that TTP may expose these vital data if it is compromised by malicious attackers. Secondly, one-side protection is vulnerable since SC-server enables to infer the location information of both requesters and workers according to the matching result. Although some solutions [19], [34]–[36] were proposed to solve the above problems, these studies failed to consider the task matching efficiency.

B. SECURE INDEX

The secure index as a promising technique has received widespread attention. Karras *et al.* [37] and Xu *et al.* [38] built a secure AVL tree by utilizing linear algebra to reduce the range query search time. Liu *et al.* [20] proposed a newly devised SKD-tree based on Paillier Cryptosystem to index worker locations, thereby improving task matching efficiency. Unfortunately, the above schemes are not practical for dynamic environments due to the huge update overhead. Although some studies were proposed to address the index dynamic update issue, those conventional approaches cannot be directly applied in the multi-user setting since user revocation will cause key redistribution and index reconstruction. In addition, attribute-based encryption (ABE) and identity-based encryption (IBE) [39]–[42] have also been extensively studied to protect the privacy in the multi-user setting. However, the major drawback is that the own-forced search is not suitable for SC [19]. Therefore, to address the above drawbacks and challenges, we design a secure and dynamic index SD-Tree, which enables SC-server to quickly process data update operations.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

We consider that our scheme works on Worker Selected Tasks (WST) mode, whereby four entities are involved in the system, as illustrated in Fig. 1, they are requesters, workers, SC-server, and key generation center (KGC).

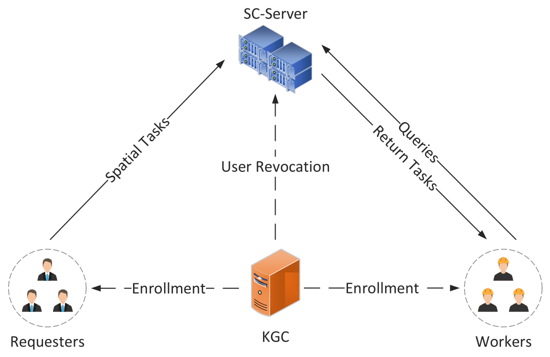

FIGURE 1. System model.

1) Requesters publish their spatial tasks including contents and specific geographic coordinates to the SC-server.
2) Workers submit their range queries to the SC-server.
3) SC-server is a spatial crowdsourcing server for task allocation services to requesters and workers.
4) KGC is a key generation center, which only performs user enrollment and revocation operations. The difference from the two-server setting is that KGC will not participate in the task matching process.

B. THREAT MODEL

KGC is responsible for the generation of the secret keys and saves all users information. It is undeniable that SC companies are reluctant to outsource these crucial data to SC-server, thus we treat KGC as a certificate authority in our threat model. We also consider requesters and workers are reliable since they are the providers of the original data. We assume that SC-server is ''honest but curious'', it follows our designed protocol to provide SC services but intends to snoop users' location privacy. Besides, the encryption of task content is however beyond our research scope. Thus, we assume that users encrypt their task content under a symmetric key K_s, and decrypt it under the same key after receiving the return results. Besides, we also assume the ID of users can be protected by faked identity or other techniques.

C. DESIGN GOALS

To realize the secure and efficient spatial task matching for SC, our scheme aims to reach the following four purposes:

1) **Location privacy**: The SESTM should preserve the location privacy of requesters and workers over one SC server.
2) **Task matching efficiency**: The SESTM should build a tree-based index based on encrypted spatial task locations from multiple requesters to improve the task matching efficiency.
3) **Dynamic update**: The SESTM should allow the index to quickly insert and delete tasks without causing high update overhead. Besides, the size of our index depends

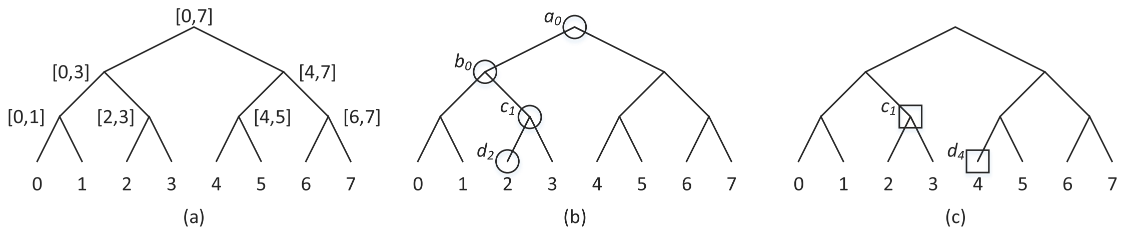

FIGURE 2. Segment tree structure, (a) the $tr(8)$ segment tree, (b) the representation of integer 2, and (c) the representation of range $[2, 4]$.

upon the number of tasks, which means there is no additional storage overhead.

4) **User scalability**: SESTM should realize favorable user scalability as the number of tasks increases. Moreover, user enrollment and revocation should not affect other users and the data in SC-server.

IV. PRELIMINARIES

In this section, we briefly introduce the background of segment tree and bilinear pairing, which will be used in our method.

A. SEGMENT TREE

Segment tree is essentially a binary tree over integers from 0 to $N-1$, denoted as $tr(N)$. The tree structure recursively divides each non-leaf node into two segments until each leaf node only contains one integer. Fig. 2(a) shows a $tr(8)$ segment tree structure, whereby each tree node v will be pre-assigned a unique node label $l(v)$ which is same as its interval. In the following, we introduce how to utilize the segment tree to express integer and range value.

Integer representation: An integer x is defined as a cover path $CP(x)$, which contains a set of nodes from tree root to leaf node. For example, $CP(2)$ is $\{a_0, b_0, c_1, d_2\}$ in Fig. 2(b).

If a $CP(x)$ intersects a tree node v, we say node v covers this integer x, defined as a label cover $LC(v)$, which represents all leaf nodes below it. For example, in Fig. 2(c), $LC(c_1)$ is $\{2, 3\}$.

Range representation: We define a one-dimensional range value $q_x = \left[q_{x_l}, q_{x_r}\right]$ as minimum cover set $MCS\left(q_x\right)$ in a segment tree, which is a set of $LC(v)$ that cover all integers in q. For example, in Fig. 2(c), $MCS\left(2, 4\right)$ is $\{LC\left(c_1\right), LC\left(d_4\right)\}$, whereby $LC\left(c_1\right)$ is $\{2, 3\}$ and $LC\left(d_4\right)$ is $\{4\}$.

And according to Shen *et al.* [43] work we have the following proposition.

Proposition 1: If $x \in q_x$, $CP\left(x\right)$ and $MCS\left(q_x\right)$ intersect at only one node.

For example in Fig. 2(b) and Fig. 2(c), $CP\left(2\right) \cap MCS\left(2, 4\right) = c_1$, thus $2 \in [2, 4]$. However, building a complete segment tree will cause huge time and space overhead, users only need to know the tree size N and then calculate the $CP\left(x\right)$ or $MCS\left(q_x\right)$ by themselves.

In addition, we can get the maximum value of $|MCS\left(q_x\right)|$ using the Theorem1 proved by Lu *et al.* [23].

Theorem 1: $\forall q_x \in [0, N-1]$, the largest $|MCS\left(q_x\right)|$ is $2 \times (logN - 1)$ if $N \geq 4$.

B. BILINEAR MAP

G and G_1 are two cyclic groups with a prime order p, whereby G denotes an additive group of a generator g and G_1 denotes a multiplicative group of a generator g_1. A bilinear map $e : G \times G \rightarrow G_1$ satisfies three properties:

Bilinearity: $\forall x, y \in Z_p^*, e\left(g^x, g^y\right) = e\left(g, g\right)^{xy}$.

Degeneracy: $e\left(g, g\right) \neq 1$

Efficient computability: e will not cause high time overhead.

V. SECURE AND EFFICIENT SPATIAL TASK MATCHING (SESTM)

To meet the secure and efficient requirements of spatial task matching over the multi-user setting, we propose a novel task matching mechanism: SESTM. In the following section, we first introduce the overview of SESTM, then detail our scheme in four aspects. Finally, we present how to dynamically update the index and the secure analysis.

A. OVERVIEW

There are four function modules in SESTM: **System Initialization, Location Transformation, Multi-User Searchable Encryption, Merge and Match**. Table.1 summarizes the main notations and Fig. 3 shows the overview of SESTM.

Definition 1: The SESTM is involved in ten algorithms (Setup, Enroll, Revoke, Geo-Trans, Ran-Trans, Index-Enc, Trap-Enc, Re-Enc, Index-Merge, Task-Match), defined below.

TABLE 1. Notations of SESTM.

Notation	Description
PK, MSK	public key and master secret key
u_i	user identity
sk_i, rk_i	secret key and re-key for user u_i
sk_R, rk_R	secret key pair for requester u_R
sk_W, rk_W	secret key pair for worker u_W
geo	geographic coordinate (x, y)
q	range query
$CP\text{-}Geo$	cover path geographic coordinate
$Enc\text{-}Index$, $Re\text{-}Index$	encrypted index and re-encrypted index for geo
$MC\text{-}Query$	minimum cover query
$Trap$, $Re\text{-}Trap$	trapdoor, re-encrypted trapdoor
$SD\text{-}Tree$	secure and dynamic tree-based index

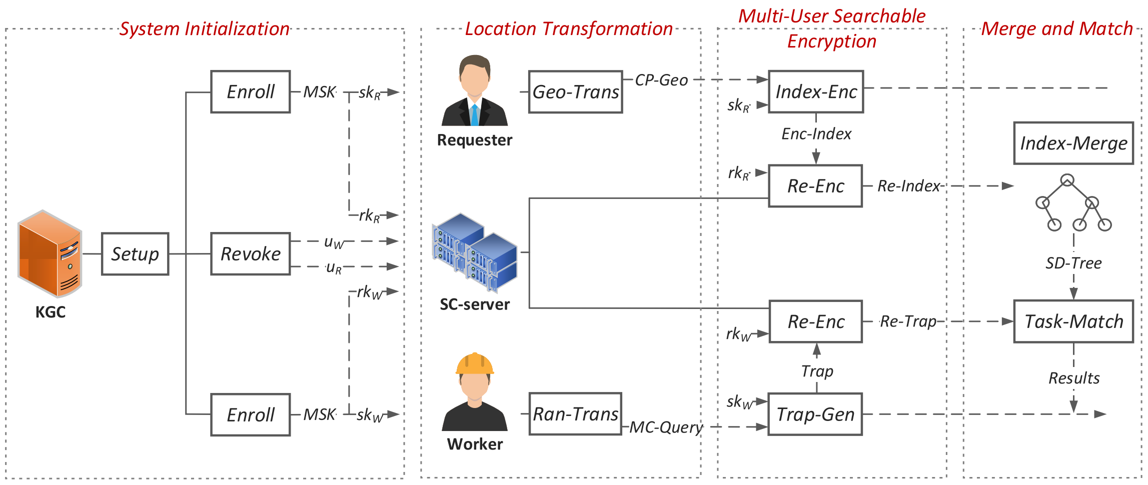

FIGURE 3. Overview of SESTM.

System Initialization: We first initialize the system and complete the user enrollment.

1) Setup $(1^\lambda) \rightarrow PK, MSK$: KGC takes a security parameter to produce a public key PK for all entities and a master key MSK which is only known to itself.
2) Enroll$(MSK, u_i) \rightarrow (sk_i, rk_i)$: KGC generates different secret key pairs (sk_i, rk_i) based on MSK for requesters and workers, whereby secret key sk_i is assigned to user for location encryption, and re-encryption key rk_i with user identity u_i is kept by SC-server for ciphertext re-encryption.
3) Revoke (u_i): KGC revokes user by removing corresponding identity u_i and re-encryption key rk_i from SC-server, which guarantees the revoked user cannot get the correct ciphertext for task matching.

After **System Initialization**, KGC will no longer participate in other phases, except for user enrollment and revocation.

Location Transformation: To achieve efficient range search over ciphertext in spatial task matching, we firstly utilize the segment tree to represent requester's geographic coordinate *geo* and worker's range query q.

1) Geo-Trans $(geo) \rightarrow CP\text{-}Geo$: Requester transforms the task geographic coordinate *geo* to a cover path geographic coordinate *CP-Geo*.
2) Range-Trans $(q) \rightarrow MC\text{-}Query$: Worker transforms the range query q to a minimum cover query *MC-Query*.

Multi-User Searchable Encryption: After transformation, requester and worker will encrypt their *CP-Geo* and *MC-Query* respectively using their unique secret keys, then submit them to SC-server for re-encryption.

1) Index-Enc $(CP\text{-}Geo, sk_R) \rightarrow Enc\text{-}Index$: Requester u_R encrypts the *CP-Geo* with own secret key sk_R, and outputs an encrypted index *Enc-Index*.
2) Trap-Gen $(MC\text{-}Query, sk_W) \rightarrow Trap$: Worker u_W encrypts the *MC-Query* with own secret key sk_W, and outputs a trapdoor *Trap*.
3) Re-Enc $(Enc\text{-}Index, rk_R) \rightarrow Re\text{-}index$, $(Trap, rk_W) \rightarrow Re\text{-}Trap$: Once receiving data from requester and worker, SC-server re-encrypts the *Enc-Index* to *Re-Index* and *Trap* to *Re-Trap* using their respective re-encryption keys rk_R and rk_W.

Merge and Match: After the encryption phase, SC-server merges different *Re-Index* to construct the secure and dynamic tree *SD-Tree*, and then provides the task matching services for worker.

1) Index-Merge $(Re\text{-}Index) \rightarrow SD\text{-}Tree$: SC-server runs the Index-Merge algorithm to merge different *Re-Index* to build the *SD-Tree*.
2) Task-Match $(SD\text{-}Tree, Re\text{-}Trap) \rightarrow Results$: SC-server matches the re-encrypted trapdoor *Re-Trap* with the *SD-Tree* and returns the results to corresponding worker.

B. SYSTEM INITIALIZATION

This phase is the initialization part of the system, which is performed by KGC and includes three vital sections, whereby **Setup** is responsible for generating the necessary parameters, **Enroll** and **Revoke** are executed when user enrollment and revocation occur, respectively.

Setup $(1^\lambda) \rightarrow (PK : G, G_1, p, g, g_1, e, H, MSK)$: KGC first generates an additive group G and a multiplicative group G_1 of prime order p with generators g and g_1, respectively. Then KGC produces a bilinear map $e : G \times G \rightarrow G_1$ and a public hash function H. Finally, KGC outputs G, G_1, p, g, g_1, e, H as public key PK for all entities and a master key MSK kept by itself.

Enroll$(MSK, u_i) \rightarrow \left(sk_i = g^{k_i}, rk_i = \frac{MSK}{k_i}\right)$: Given a user u_i, KGC first chooses a $k_i \in Z_p^+$ and computes $rk_i = \frac{MSK}{k_i}$.

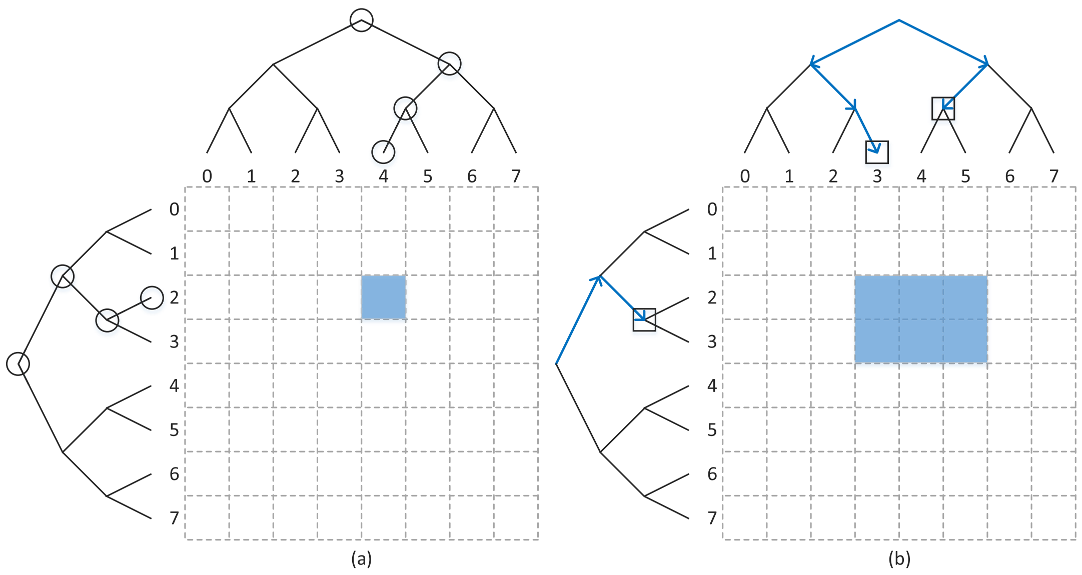

FIGURE 4. Location transformation. a) The geographic coordinate representation, and b) the range value representation.

Then KGC sends g^{k_i} to u_i as secret key sk_i and $\frac{MSK}{k_i}$ to SC-server as re-encrypted key rk_i of u_i.

Revoke(u_i): User identities and re-encrypted keys are saved as $\{u_i : rk_i\}$ in SC-server and KGC enables to revoke any user by deleting the corresponding u_i and rk_i from SC-server.

C. LOCATION TRANSFORM

Privacy task matching is essentially a range query search, but the challenge is how to check whether a geographic coordinate is within a specific range over ciphertext. To address this issue, previous works [13], [19], [23] utilized the segment tree to transform the location data and perform the range search. However, the time complexity is linear time $O(n)$ which is not efficient for the practical. Therefore, we propose a novel privacy range query search based on their works [13], [19], [23] and the difference is that we reduce the time complexity to logarithmic time $O(logn)$. Firstly, we introduce the fundamental of their method, then we explain our innovation in detail.

Now we give an example to introduce the location transform operation. Given a map $N_x \times N_y$, where N_x, N_y are the maximum value in each dimension, we divide the map into $N_x \times N_y$ zones and each one will be assigned a unique identity $l\,(i,j)$, where $0 \le i < N_x, 0 \le j < N_y$. Then we build two segment trees to index each coordinate point, as an example in Fig. 4(a) (the figure is modified from Fig.4 of study [19]), whereby a location $(4, 2)$ is expressed by $CP(4)$ and $CP(2)$ in each dimension, which is marked by circles. Similarly, MCS ([3, 5]) and MCS ([2, 3]) represent the range value $[3, 5] \times [2, 3]$ in Fig. 4(b), which is marked by squares.

Theorem 2: A coordinate point (x, y) is within a range $\left[q_{x_l}, q_{x_r}\right] \times \left[q_{y_l}, q_{y_r}\right]$ iff $CP(x)$ and $CP(y)$ intersect $MCS\left(\left[q_{x_l}, q_{x_r}\right]\right)$ and $MCS\left(\left[q_{y_l}, q_{y_r}\right]\right)$, respectively.

And for clarity, we utilize $CP(x, y)$ to donate $CP(x)$ and $CP(y)$, $MCS(q)$ to donate $MCS\left(\left[q_{x_l}, q_{x_r}\right]\right)$ and $MCS\left(\left[q_{y_l}, q_{y_r}\right]\right)$ in the following.

Therefore, the SC-server can perform the range search by checking the intersection of query q and task geographic coordinate (x, y) over the ciphertext domain. However, the drawback is that a $MCS(q)$ must traverse all $CP(x, y)$ to get results. Clearly, the time complexity is $O(2n)$ for n tasks. Therefore, to improve the task matching efficiency, we propose an innovative transformation to replace it. We first introduce it from two aspects: **Range-Trans** and **Geo-Trans**, then we will detail the operating mechanism in the following section.

Range-Trans $(q) \rightarrow MC\text{-}Query$: we utilize a new definition called minimum cover query $MC\text{-}Query\,(q)$ to replace $MCS\,(q)$. The difference between them is that $MC\text{-}Query\,(q)$ not only finds the minimum cover set $MCS\,(q)$ for each query, but also indexes them by segment tree, which means nodes in $MCS\,(q)$ are leaf nodes of $MC\text{-}Query\,(q)$. As an example, a range query $q = [3, 5] \times [2, 3]$ is expressed by $MC\text{-}Query$ ([3, 5]) and $MC\text{-}Query$ ([2, 3]), which is marked by blue arrow line in in Fig. 4(b).

Geo-Tran $(loc) \rightarrow CP\text{-}Index$: Similarly, we use cover path index $CP\text{-}Index\,(x, y)$ to replace $CP(x, y)$, which only changes the $CP(x, y)$ data structure into a linked-list.

We have the **Theorem 3** by this means to express location data:

Theorem 3: A geographic coordinate (x, y) is within a range $q = \left[q_{x_l}, q_{x_r}\right] \times \left[q_{y_l}, q_{y_r}\right]$ iff the $CP\text{-}Index\,(x)$ and $CP\text{-}Index\,(y)$ intersect the leaf nodes of $MC\text{-}Query\left(\left[q_{x_l}, q_{x_r}\right]\right)$ and $MC\text{-}Query\left(\left[q_{y_l}, q_{y_r}\right]\right)$, respectively.

D. MULTI-USER SEARCHABLE ENCRYPTION

After the **Location Transformation**, a task geographic coordinate (x, y) is replaced by $CP\text{-}Index(x, y)$ and range query

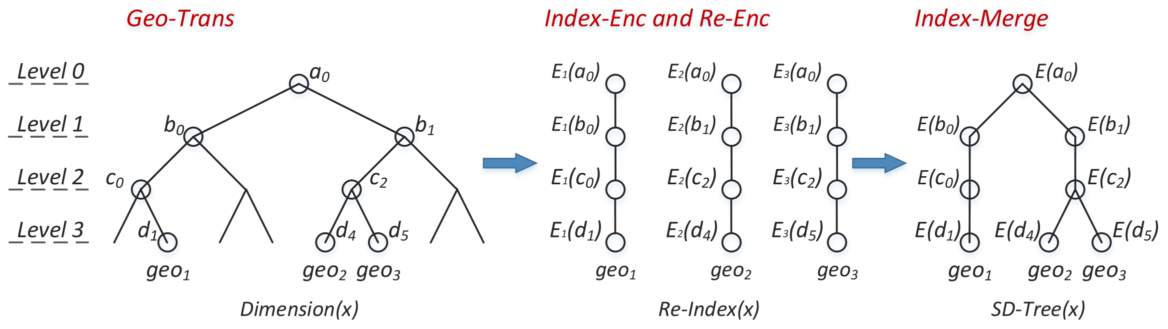

FIGURE 5. Merge process.

q is replaced by *MC-Query*(q). However, node labels in *CP-Index*(x, y) and *MC-Query*(q) all are plaintext, malicious adversaries may infer some useful information. In order to prevent privacy leakage, we propose a novel multi-user searchable encryption to protect the data. Meanwhile, this encryption scheme also allows the SC-server to perform some efficient mathematical operations on ciphertexts even if they are encrypted by different keys.

Index-Enc $(CP\text{-}Index, sk_R) \rightarrow Enc\text{-}Index$: *CP-Index* can be presented as $\{l_{R,1}, l_{R,2} \ldots l_{R,d}\}$, whereby R is the requester identity and d is the dth node. Requester u_R uses own $sk_R = g^{k_R}$ and randomly chooses a $r_{R,d} \in Z_p^+$ to encrypt each node label as:

$$\left(g^{k_R \cdot H(l_{R,d}) \cdot r_{R,d}}, g^{r_{R,d}}\right). \tag{1}$$

Finally, the requester outputs two encrypted indexes *Enc-Index*(x) and *Enc-Index*(y) for *CP-Index*(x) and *CP-Index*(y), respectively.

Trap-Gen $(MC\text{-}Query, sk_W) \rightarrow Trap$: Similarly, worker u_W uses own $sk_W = g^{k_W}$ and randomly chooses a $r_{W,d} \in Z_p^+$ to encrypt each node label in *MC-Query*(q_x) and *MC-Query*(q_y), then outputs trapdoors *Trap*(q_x) and *Trap*(q_y).

$$\left(g^{k_W \cdot H(l_{W,d}) \cdot r_{W,d}}, g^{r_{W,d}}\right). \tag{2}$$

Re-enc $(Enc\text{-}Index, rk_R) \rightarrow Re\text{-}index$, $(Trap, rk_W) \rightarrow Re\text{-}Trap$: Once receiving ciphertexts from requester u_R, SC-server re-encrypts *Enc-Index* by its $rk_R = \frac{MSK}{k_R}$

$$\left(\left(g^{k_R \cdot H(l_{R,d}) \cdot r_{R,d}}\right)^{rk_R}, g^{r_{R,d}}\right) = \left(g^{MSK \cdot H(l_{R,d}) \cdot r_{R,d}}, g^{r_{R,d}}\right) \tag{3}$$

For clarity, we let $P_{R,d} = g^{MSK \cdot H(l_{R,d}) \cdot r_{R,d}}$, $T_{R,d} = g^{r_{R,d}}$, E_R denotes the encrption scheme for requester u_R.

$$E_R\left(l_{R,d}\right) = \left(P_{R,d}, T_{R,d}\right) \tag{4}$$

Therefore, the *Re-Index* is denoted as $\{E_R(l_{R,1}), E_R(l_{R,2}) \ldots E_R(l_{R,d})\}$. Similarly, SC-server uses the same method for worker and outputs the re-encrypted trapdoor *Re-Trap*(x) and *Re-Trap*(y).

E. MERGE AND MATCH

To achieve efficient task matching, SC-server first uses **Index-Merge** to merge all *Re-Index* of different requesters and constructs two tree-based index *SD-Tree* for two dimension x and dimension y, then performs the **Task-Match** for all permissible workers' queries.

Index-Merge $(Re\text{-}Index) \rightarrow SD\text{-}Tree$: The depth of *Re-Index* is fixed for all requesters because they derive from the same segment tree, thereby the SC-server enables to merge all *Re-Index* from top to leaf node. There is an example to show how to check whether two node labels are equivalent. Given two re-encrypted node labels in SC-server:

$$E_i\left(l_{i,d}\right) = \left(g^{MSK \cdot H(l_{i,d}) \cdot r_{i,d}}, g^{r_{i,d}}\right) = \left(P_{i,d}, T_{i,d}\right) \tag{5}$$

$$E_j\left(l_{j,d}\right) = \left(g^{MSK \cdot H(l_{j,d}) \cdot r_{j,d}}, g^{r_{j,d}}\right) = \left(P_{j,d}, T_{j,d}\right) \tag{6}$$

SC-server first computes as follows:

$$\begin{aligned} e\left(P_{i,d}, T_{j,d}\right) &= e\left(g^{MSK \cdot H(l_{i,d}) \cdot r_{i,d}}, g^{r_{j,d}}\right) \\ &= e(g, g)^{MSK \cdot H(l_{i,d}) \cdot r_{i,d} \cdot r_{j,d}} \end{aligned} \tag{7}$$

$$\begin{aligned} e\left(P_{j,d}, T_{i,d}\right) &= e\left(g^{MSK \cdot H(l_{j,d}) \cdot r_{j,d}}, g^{r_{i,d}}\right) \\ &= e(g, g)^{MSK \cdot H(l_{j,d}) \cdot r_{j,d} \cdot r_{i,d}} \end{aligned} \tag{8}$$

It can easily get $e\left(P_{i,d}, T_{j,d}\right) = e\left(P_{j,d}, T_{i,d}\right)$ if and only if $l_{i,d} = l_{j,d}$. For clarity, we define a new operator #, it has the following property:

$$E_i\left(l_{i,d}\right) \# E_j\left(l_{j,d}\right) = \frac{e\left(P_{i,d}, T_{j,d}\right)}{e\left(P_{j,d}, T_{i,d}\right)} \tag{9}$$

If $E_i\left(l_{i,d}\right) \# E_j\left(l_{j,d}\right) = 1$, it means the two nodes are equivalent. Otherwise, it is different.

Now, we give an example to show the merging process. We assume the map size is 8 and three are three task locations geo_1, geo_2 and geo_3, their geographic coordinates are $(1, 1)$, $(4, 4)$ and $(5, 5)$ respectively. In order to express all points, we use two segment trees $tr(8)$ to index each dimension. For clarity, we only show the dimension x in Fig 5, whereby the tree height is $h = log8 + 1$. After **Enroll**, **Geo-trans**, **Index-Enc**, and **Re-Enc**, SC-server gets three *Re-Index*(x) as Fig 5 shows. Then SC-server merges them from root to leaf node.

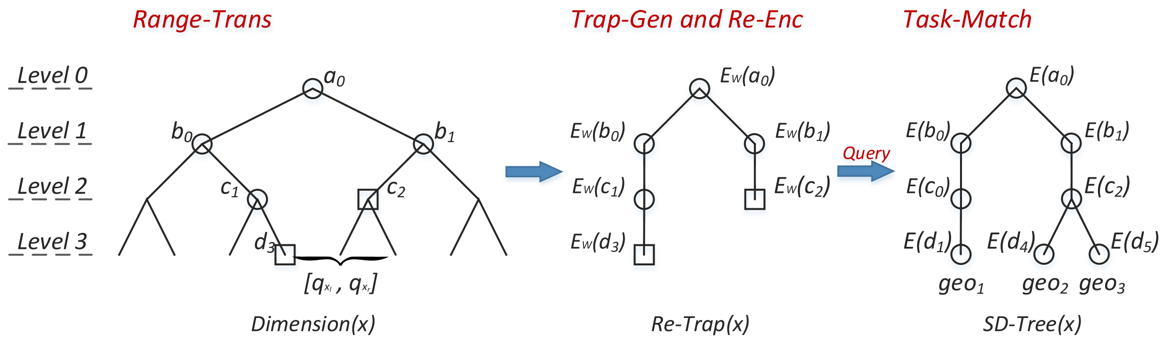

FIGURE 6. Match process.

At level 0:

$$E_1(a_0)\#E_2(a_0)=1,\quad E_2(a_0)\#E_3(a_0)=1 \tag{10}$$

Later on, SC-server merges them into one node kept as follows, whereby $E(a_0)$ can be $E_1(a_0)$, $E_2(a_0)$ or $E_3(a_0)$.

$$node_0 =< E(a_0), child_0, child_1 > \tag{11}$$

At level 1:

$$E_1(b_0)\#E_2(b_1)\neq 1,\quad E_2(b_1)\#E_3(b_1)=1 \tag{12}$$

SC-server splits the $node_0$ into two branches, whereby one branch is occupied by loc_0, then loc_1 and loc_2 merge the level 1 node and continue the merge process until they cannot find an equivalent node. Similarly, SC-server performs the same operation for another dimension, finally outputs *SD-Tree*(x) and *SD-Tree*(y).

Task-Match (*SD-Tree*, *Re-Trap*) $\rightarrow$ *results*: We also give an example to show the task matching operation for one dimension. Given a worker u_W with a range query $q_x = [q_{x_l}, q_{x_r}]$ as shown in Fig. 6. After **Enroll**, **Geo-trans**, **Index-Enc** and **Re-Enc**, SC-server gets the *Re-Trap*(x).

According to **Theorem 3**, SC-server should find whether the leaf node of *Re-Trap*(x) intersects with the *SD-Tree*(x). Therefore, we adopt Depth First Search (DFS) algorithm to match the *Re-Trap*(x) with the merged index. For example, in Fig. 6 SC-server first travels the left branch of *Re-Trap*(x), it is easy to know:

$$\begin{aligned} E_W(a_0)\#E(a_0) &= 1\\ E_W(b_0)\#E(b_0) &= 1\\ E_W(c_1)\#E(c_0) &\neq 1 \end{aligned} \tag{13}$$

Clearly, $E_W(c_1)$ and $E(c_0)$ are not equivalent at level 2, which means the leaf node $E_W(d_2)$ of this branch will not intersect any node in *SD-Tree*(x). Thus, SC-server continuously travels the right branch of the *Re-Trap*(x), luckily finding the leaf node $E_W(c_2)\#E(c_2) = 1$, which means $loc_2, loc_3 \in q_x$ and get the $results_x$. Similarly, SC-server performs the **Task-Match** to another dimension and find the intersection set $results_y$. Finally, SC-server returns the $results = results_x \cap results_y$ to worker.

F. DYNAMIC UPDATE

In a real spatial crowdsourcing environment, the number of tasks is not fixed since SC-Server needs to continuously insert new tasks and delete accepted tasks. However, the update operations are very time consuming for the static index, such as KD-tree. Therefore, we introduce the dynamic properties of our *SD-Tree* in the following.

Deletion: Tree branches in *SD-Tree* represent requesters' geographic coordinates, thus the deletion is removing the specific branch from the index. However, SC-server cannot directly delete it from root to leaf node because any tree node may be shared by multiple leaf nodes. Thus, SC-server should firstly find the leaf node, then remove this branch from leaf node to root. For example, if SC- server plans to delete geo_2 in Fig 6, it firstly finds the leaf node in *SD-Tree*(x), then removes tree node $E(d_4)$ and stop deletion because $E(c_2)$ is shared by other.

However, the worst time complexity of deletion for *SD-Tree*(x) is $O(2logN)$, it cannot reach the best logarithmic time $O(logN)$. Therefore, to improve the efficiency of deletion, we add a counting item into each tree node, which is kept as:

$$node =< E(a_0), child_0, child_1, count > \tag{14}$$

The role of the counting item is to record how many branches share this tree node, which also means how many leaf nodes it contains. Therefore, SC-server only needs to know the *count* value of tree node before deleting it. For example, if SC-server intends to delete geo_3 in Fig. 6, it only performs $count - 1$ for tree nodes $E(a_0)$, $E(b_1)$ and $E(c_2)$ because they are shared nodes, then removes $E(d_5)$ because its $count = 1$. Therefore, the time complexity of deletion reduces to $O(logN)$ on one dimension, which also means it is $O(2logN)$ for *SD-Tree*(x, y).

Insertion: The insert operation is same as the merging, thus the time complexity is $O(2logN)$ for one data.

G. SECURITY ANALYSIS

In this section, we give the security analysis of our scheme.

Theorem 4: Our scheme achieves security against chosen-plaintext attack under the DBDH assumption in selective

TABLE 2. Notations.

Notation	Description
E, P	Exponentiation and pairing operation on group G, respectively
H, f_s	Hash operation and key-based hash operation, respectively
h_x & h_y	The height of segment tree $tr(N)$ is $h = logN + 1$ in each dimension
m_x & m_y	The number of node labels in the $MCS\ (q_x)$ and $MCS\ (q_y)$, respectively

TABLE 3. Computation cost.

Steps	SC-MSDE	SESTM
Task publication	$(3E + f_s + H)\,(h_x + h_y)$	$(2E + H)\,(h_x + h_y)$
Trapdoor generation	$(5E + f_s)\,(m_x + m_y)$	$(2E + H)\,(m_x h_x + m_y h_y)$
Task matching (n tasks)	$nH\,(m_x h_x + m_y h_y)$	$(2P)\,(m_x h_x + m_y h_y)$

TABLE 4. Communication cost.

Steps	SC-MSDE	SESTM
Requester to SC-Server	$\left\|Z_p^*\right\| + \left(2\left\|\mathbb{G}\right\| + \left\|Z_p^*\right\|\right)(h_x + h_y)$	$\left\|Z_p^*\right\| + 2\left\|\mathbb{G}\right\|(h_x + h_y)$
Worker to SC-Server	$\left\|Z_p^*\right\| + 2\left\|\mathbb{G}\right\|(m_x + m_y)$	$\left\|Z_p^*\right\| + 2\left\|\mathbb{G}\right\|(m_x h_x + m_y h_y)$

security model and plaintext secrecy under discrete logarithm (DL) assumption in random oracle model.

Proof: Since our encryption scheme is based on PRMSM [17] and TBMSM [22], thus the security analysis of SESTM is similar as PRMSM and TBMSM, which ensures that no additional information except the ciphertexts will be leaked to adversaries. This proof process can be referred to the **Theorem 1** and **Theorem 2** in TBMSM.

VI. PERFORMANCE ANALYSIS

In this section, we take the time-consuming operations to evaluate our performance and compare with SC-MSDE [19] in the aspects of computation overhead and communication overhead.

A. SD-TREE CONSTRUCTION OVERHEAD

We evaluate the *SD-Tree* construction overhead from **Index construction** and **Index update**, and Table 2 shows the necessary notations.

Index construction: There are two major operations for the SC-server to construct the index: 1) re-encrypting the *Enc-Index* to *Re-Index* by **Re-enc** algorithm, and 2) merging *Re-Index* into the *SD-Tree* by **Index-Merge** algorithm. For each *Enc-Index*, $h_x + h_y$ node labels are contained, thereby SC-server takes $E\left(h_x + h_y\right)$ to re-encrypt it. Similarly, it costs $2P\left(h_x + h_y\right)$ to merge one into the *SD-Tree*. As whole, it takes at most $n\,(2P + E)\left(h_x + h_y\right)$ to build the *SD-Tree* for n spatial tasks.

Index update: The insertion and deletion are similar to the merge process, thereby the update operations are also $(2P + E)\left(h_x + h_y\right)$.

B. COMPUTATION OVERHEAD

The computation overhead of SESTM is tested through the items of **Task publication**, **Task search** and **Task matching**. Table 3 displays and compares the cost of time-consuming operations in each phase.

Task publication: Each transformed task location *CP-Geo*(x, y) includes $h_x + h_y$ node labels, thereby requester takes $(2E + H)\left(h_x + h_y\right)$ to encrypt it.

Trapdoor Generation: The trapdoor generation process of SESTM is similar to the one in SC-MSDE. It firstly finds the *MCS* (q) for a range query q and then transforms it into *MC-Query* (q) which contains up to $m_x h_x + m_y h_y$ node labels. Thus, the maximum computation overhead is $(2E + H)\left(m_x h_x + m_y h_y\right)$.

Task matching: In SESTM, we adopt the DFS algorithm to match the query and *SD-Tree*, SC-server only needs to travel the *Re-Trap* (q) rather than *SD-Tree*. Therefore, the computation overhead is $2P\left(m_x h_x + m_y h_y\right)$ at most, which has no relation with task number n.

C. COMMUNICATION OVERHEAD

The communication overhead mainly depends on the transmission cost between **Worker** to **SC-Server** and **Requester** to **SC-Server**. Table. 4 present the analysis of SC-MSDE and SESTM.

VII. EXPERIMENTAL EVALUATION

In this section, we evaluate our scheme with simulation experiments. Specifically, we first test the **SD-Tree construction overhead** of SESTM, and then compare the **Computation overhead** and **Communication overhead** of SC-MSDE and SESTM.

A. SETUP AND DATASET

We implement both schemes on a python3 platform, with the encryption based on the PBC library [44] and Charm framework [45]. The configuration is on a ubuntu 18.04 system with a i3-3240 CPU at 3.4GHz and 4 GB RAM. Due to the absence of a spatial crowdsourcing dataset, we adopt

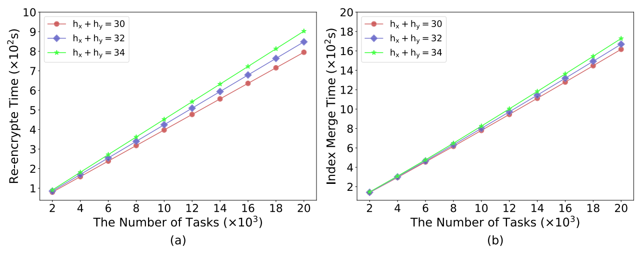

FIGURE 7. Time cost of index construction. a) The re-encrypte time of three $h_x + h_y$, and b) the index merge time of three $h_x + h_y$.

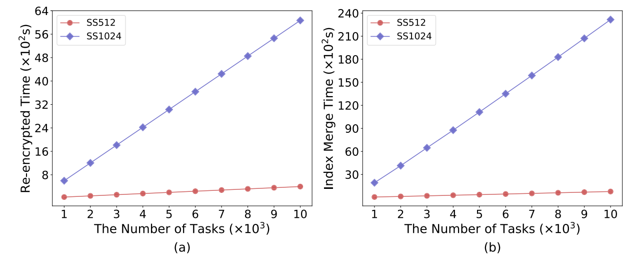

FIGURE 8. Time cost of index construction. a) The re-encrypted time of two elliptic curves, and b) the index merge time of two elliptic curves.

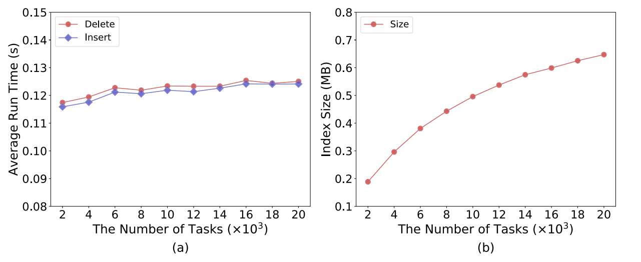

FIGURE 9. Average runtime and index size: a) the average delete and insert time, and b) the index size.

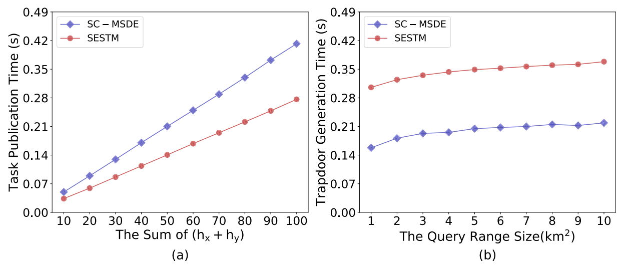

FIGURE 10. Runtime of task publication and trapdoor generation, a) the publication time with varying $h_x + h_y$, and b) trapdoor generation time with different query range size.

a real-world location share dataset Gowalla [46] to simulate our experiments, whereby we consider each Gowalla user as a requester and the check-in geographic coordinate as the spatial task location. Besides, we apply the Universal Transverse Mercator (UTM) projection to transform all geographic coordinates into integer points and control the accuracy to 1m, then map them into a fixed map $N \times N$.

In the following experiments, we set the default map size to $16km \times 16km$, which means it needs two $tr(2^{14})$ segment trees to index each coordinate. Thus the corresponding $\left(h_x + h_y\right)$ is 30 since the tree height equals $log2^{14} + 1$. And we also set the default query range is $1km \times 1km$. For the task number n, the default value is 20000.

B. SD-TREE CONSTRUCTION EVALUATION

We evaluate the *SD-Tree* with the aspects of **Index construction** and **Index update** in the following.

Index construction: As described in Section VI. A, the *SD-Tree* construction overhead is mainly decided by the tasks number n and the sum of $\left(h_x + h_y\right)$. Fig. 7(a) and (b) respectively show the re-encrypt time and index merge time. Clearly, both results are increasing with the number of tasks and also proportional to the sum of $\left(h_x + h_y\right)$, which show our *SD-Tree* has a good scalability. Besides, we also consider the influence of elliptic curves size. As shown in Fig. 8(a) and (b), it is shown that the runtime of SS1024 is much higher than that of SS512, which justifies the balance between index construction efficiency and privacy.

Index update: We evaluate the update overhead of our scheme by deleting and inserting one data from the *SD-Tree*. Fig. 9(a) illustrates the average runtime of 200 operations. Taking the insertion as an example, the figure has little changes when the number of tasks increases from 2000 to 20000, which justifies our theoretical analysis in Section VI. A and demonstrates that our *SD-Tree* achieves dynamic and fast updates.

On the other hand, we also measure the storage of our *SD-Tree*. As shown in Fig. 9(b), the number of tasks increases tenfold, but the index size only rises from 0.2 MB to 0.4 MB. The reason behind this is that more and more tree nodes are shared as the number of tasks increases, which greatly saves the storage overhead.

C. COMPUTATION OVERHEAD EVALUATION

In this section, we evaluate the computation overhead of SC-MSDE and SESTM with the items of **Task publication**, **Trapdoor generation** and **Task matching**. The analysis is in VI. B and shown in Table 3.

Task publication: We record the time cost of location encryption to evaluate the computation overhead in the task publication phase. As presented in Fig. 10(a), the results of both schemes are almost linear with the sum of $h_x + h_y$. And we also observe that SC-MSDE is more time-consuming than SESTM.

Trapdoor Generation: Fig. 10(b) illustrates the runtime of trapdoor generation operation, due to the **Theorem 1**, we observe that the figure of both schemes rises slightly and then stabilizes, as the query range size increases. Although SESTM relatively consumes more time than SC-MSDE, it is however acceptable since the maximum value is below 0.35s.

Task matching: In order to evaluate the task matching efficiency, we record the average searching time of 100 random queries for both schemes. Firstly, we vary the task

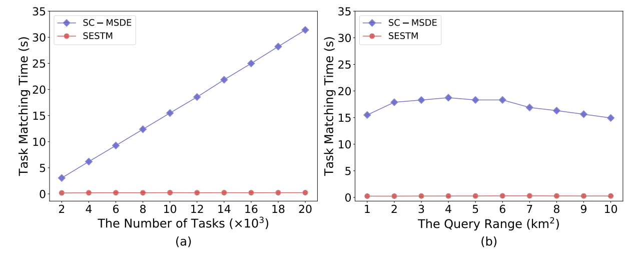

FIGURE 11. Average task matching time, a) with varying task number, and b) with varying query size.

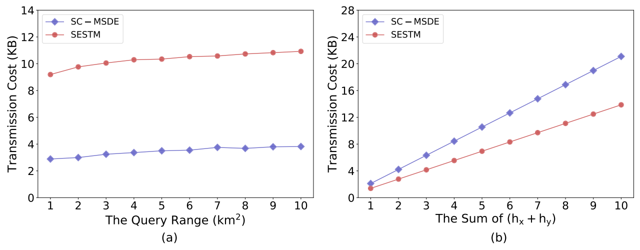

FIGURE 12. Transmission cost, a) with different query range size, and b) with different sum of $h_x + h_y$.

number from 2000 to 20000 with a default query range size $1km \times 1km$. Fig. 11(a) reports that the time cost is linear with the number of tasks in SC-MSDE, while the figure of SESTM remains around 0.3s. It accords with our theoretical analysis in VI. B. We also observe the influence of different range query size in Fig. 11(b). From the results in this chart, we observe that the trend is increasing first then decreasing. The reason behind it is that the sum of $m_x h_x + m_y h_y$ is increasing as the query range increases, which corresponds to the rising part of the graph. However, in order to improve query efficiency, both methods utilize high-level nodes to replace its child nodes, for example, $MCS(0)$, $MCS(1)$, $MCS(2)$, $MCS(3)$ can be replaced by $MCS([0, 3])$. Therefore, the sum of $m_x h_x + m_y h_y$ is decreasing as the area continues to decrease, which corresponds to the falling part of the graph. Actually, SESTM also follows this trend, however, it remains stable since the ratio of the y-axis of the graph is too large.

Clearly, our scheme has an apparent advantage over the SC-MSDE in the task matching phase. The reason is that SC-MSDE needs to match all ciphertexts to get the results, while SESTM built a tree-based index to improve the searching efficiency.

D. COMMUNICATION OVERHEAD EVALUATION

In this section, we perform the communication evaluation on SC-MSDE and SESTM. We record the transmission cost between **Worker** to **SC-Server** and **Requester** to **SC-Server**.

Worker to **SC-server**: The transmission cost between worker and SC-server mainly depends on the data size of ciphertexts. Fig. 12(a) shows the communication overhead with different query range sizes, the result of SESTM is slightly more than that of SC-MSDE. The reason is the transformed worker location is a tree-based structure in SESTM, not a set of labels in SC-MSDE.

Requester to **SC-server**: Similarly, we test the transmission cost of trapdoor from requester to SC-server. As shown in Fig. 12(b), the results of both schemes are proportional to the sum of $h_x + h_y$, and SESTM costs less communication overhead compared to SC-MSDE.

VIII. CONCLUSION

In this paper, we propose a secure and efficient task matching scheme for spatial crowdsourcing over the single server setting. Firstly, we adopt the multi-user searchable encryption protocol to protect all users' location privacy and also achieve effective user revocation by utilizing the proxy-encryption technique. We secondly design a secure and dynamic tree-based index *SD-Tree* to address the task matching efficiency issue. Moreover, our *SD-Tree* has favorable user scalability and enables to dynamic update compared with static index structure. Finally, we evaluate the *SD-Tree* construction overhead of our scheme and also compare SESTM with SC-MSDE in the aspects of computation overhead and communication overhead. Results show our scheme realizes the proposed goals and has an apparent advantage over SC-MSDE in the task matching phase.

In future work, we will evaluate the influence of task encryption schemes and task distributions on post-processing time.

REFERENCES

[1] L. Kazemi and C. Shahabi, "Geocrowd: Enabling query answering with spatial crowdsourcing," in *Proc. SIGSPATIAL*, Redondo Beach, CA, USA, Nov. 2012, pp. 189–198.
[2] B. Guo, Y. Liu, L. Wang, V. O. K. Li, J. C. K. Lam, and Z. Yu, "Task allocation in spatial crowdsourcing: Current state and future directions," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1749–1764, Jun. 2018.
[3] *Uber*. Accessed: Nov. 7, 2019. [Online]. Available: https://www.uber.com/
[4] *Amazon Mechanical Mturk*. Accessed: Jun. 18, 2019. [Online]. Available: https://www.mturk.com/
[5] *Witmart*. Accessed: Aug. 13, 2019. [Online]. Available: http://www.witmart.com/
[6] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. KDD*, San Diego, CA, USA, Aug. 2011, pp. 1082–1090.
[7] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: Optimal strategy against localization attacks," in *Proc. CCS*, Raleigh, NC, USA, Oct. 2012, pp. 617–627.
[8] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[9] J. Wei, Y. Lin, X. Yao, and J. Zhang, "Differential privacy-based location protection in spatial crowdsourcing," *IEEE Trans. Services Comput.*, vol. 16, no. 4, pp. 934–949, Jun. 2019.
[10] J. Scheck. (2010). *Stalkers Exploit Cellphone GPS*. [Online]. Available: http://www.wsj.com/
[11] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *Proc. MDM*, Brisbane, QLD, Australia, vol. 1, Jul. 2014, pp. 73–82.
[12] 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, Jun. 2014.

[13] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, ''Multi-dimensional range query over encrypted data,'' in *Proc. SP*, Berkeley, CA, USA, May 2007, pp. 350–364.
[14] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, ''Public key encryption with keyword search,'' in *Proc. EUROCRYPT*, Interlaken, Switzerland, May 2004, pp. 506–522.
[15] D. Wang, X. Jia, C. Wang, K. Yang, S. Fu, and M. Xu, ''Generalized pattern matching string search on encrypted data in cloud systems,'' in *Proc. INFOCOM*, Hong Kong, Apr. 2015, pp. 2101–2109.
[16] 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.
[17] 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.
[18] X. Yao, Y. Lin, Q. Liu, and J. Zhang, ''Privacy-preserving search over encrypted personal health record in multi-source cloud,'' *IEEE Access*, vol. 6, pp. 3809–3823, Jan. 2018.
[19] J. Shu, X. Liu, Y. Zhang, X. Jia, and R. H. Deng, ''Dual-side privacy-preserving task matching for spatial crowdsourcing,'' *J. Netw. Comput. Appl.*, vol. 123, pp. 101–111, Dec. 2018.
[20] B. Liu, L. Chen, X. Zhu, Y. Zhang, C. Zhang, and W. Qiu, ''Protecting location privacy in spatial crowdsourcing using encrypted data,'' in *Proc. Adv. Database Technol.-EDBT*, Mar. 2017, pp. 478–481.
[21] D. Zhai, Y. Sun, A. Liu, Z. Li, G. Liu, L. Zhao, and K. Zheng, ''Towards secure and truthful task assignment in spatial crowdsourcing,'' *World Wide Web*, vol. 22, no. 5, pp. 2017–2040, Sep. 2019.
[22] T. Peng, Y. Lin, X. Yao, and W. Zhang, ''An efficient ranked multi-keyword search for multiple data owners over encrypted cloud data,'' *IEEE Access*, vol. 6, pp. 21924–21933, Apr. 2018.
[23] Y. Lu, ''Privacy-preserving logarithmic-time search on encrypted data in cloud,'' in *Proc. NDSS*, San Diego, CA, USA, Feb. 2012, pp. 1–17.
[24] 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, May 2011.
[25] D. Deng, C. Shahabi, and L. Zhu, ''Task matching and scheduling for multiple workers in spatial crowdsourcing,'' in *Proc. SIGSPATIAL*, New York, NY, USA, Nov. 2015, p. 21.
[26] H. Chen, B. Guo, Z. Yu, L. Chen, and X. Ma, ''A generic framework for constraint-driven data selection in mobile crowd photographing,'' *IEEE Internet Things J.*, vol. 4, no. 1, pp. 284–296, Feb. 2017.
[27] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, ''Task assignment on multi-skill oriented spatial crowdsourcing,'' *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2201–2215, Aug. 2016.
[28] Y. Zheng, G. Li, and R. Cheng, ''DOCS: A domain-aware crowdsourcing system using knowledge bases,'' *Proc. VLDB Endowment*, vol. 10, no. 4, pp. 361–372, Nov. 2016.
[29] V. Ambati, S. Vogel, and J. Carbonell, ''Towards task recommendation in micro-task markets,'' in *Proc. AAAI*, San Francisco, CA, USA, Jan. 2011, pp. 80–83.
[30] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma, ''Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation,'' in *Proc. WWW*, Perth, WA, Australia, Apr. 2017, pp. 627–636.
[31] Y. Gong, L. Wei, Y. Guo, C. Zhang, and Y. Fang, ''Optimal task recommendation for mobile crowdsourcing with privacy control,'' *IEEE Internet Things J.*, vol. 3, no. 5, pp. 745–756, Oct. 2016.
[32] P. Samarati, ''Protecting respondents identities in microdata release,'' *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 6, pp. 1010–1027, Nov. 2001.
[33] B. Gedik and L. Liu, ''Protecting location privacy with personalized K-anonymity: Architecture and algorithms,'' *IEEE Trans. Mobile Comput.*, vol. 7, no. 1, pp. 1–18, Jan. 2008.
[34] J. Shu, X. Jia, K. Yang, and H. Wang, ''Privacy-preserving task recommendation services for crowdsourcing,'' *IEEE Trans. Services Comput.*, early access, Jan. 10, 2018, doi: 10.1109/TSC.2018.2791601.
[35] 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.*, early access, Oct. 12, 2019, doi: 10.1109/TDSC.2018.2875682.
[36] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, and K. Li, ''Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners,'' *Future Gener. Comput. Syst.*, vol. 100, pp. 689–700, Nov. 2019.
[37] P. Karras, A. Nikitin, M. Saad, R. Bhatt, D. Antyukhov, and S. Idreos, ''Adaptive indexing over encrypted numeric data,'' in *Proc. SIGMOD*, San Francisco, CA, USA, Jun. 2016, pp. 171–183.
[38] Z. Xu, Y. Lin, V. K. Arthur Sandor, Z. Huang, and X. Liu, ''A lightweight privacy and integrity preserving range query scheme for mobile cloud computing,'' *Comput. Secur.*, vol. 84, pp. 318–333, Jul. 2019.
[39] Z. Cai, H. Yan, P. Li, Z.-A. Huang, and C. Gao, ''Towards secure and flexible EHR sharing in mobile health cloud under static assumptions,'' *Cluster Comput.*, vol. 20, no. 3, pp. 2415–2422, Sep. 2017.
[40] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, ''Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing,'' *Inf. Sci.*, vol. 379, pp. 42–61, Feb. 2017.
[41] Y. Zhang, D. Zheng, and R. H. Deng, ''Security and privacy in smart health: Efficient policy-hiding attribute-based access control,'' *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018.
[42] H. Yin, Y. Xiong, T. Deng, H. Deng, and P. Zhu, ''A privacy-preserving and identity-based personalized recommendation scheme for encrypted tasks in crowdsourcing,'' *IEEE Access*, vol. 7, pp. 138857–138871, Sep. 2019.
[43] E. Shen, E. Shi, and B. Waters, ''Predicate privacy in encryption systems,'' in *Proc. TCC*, San Francisco, CA, USA, Feb. 2009, pp. 457–473.
[44] A. De Caro and V. Iovino, ''JPBC: Java pairing based cryptography,'' in *Proc. ISCC*, Kerkyra, Greece, Jun. 2011, pp. 850–855.
[45] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, ''Charm: A framework for rapidly prototyping cryptosystems,'' *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, Jun. 2013.
[46] *Gowalla*. [Online]. Available: https://snap.stanford.edu/data/loc-gowalla.html

FULIN ZHOU received the B.S. degree from Shanghai Maritime University, China, in 2018. He is currently pursuing the M.S. degree in computer science and technology with Hunan University. His research interests include cloud computing security and big data.

JUNYI LI received the bachelor's, master's, and Ph.D. degrees in computer science from Hunan University, in 1993, 2001, and 2008, respectively. He has been an Associate Professor with Hunan University, since 2005. From 2009 to 2010, he was a Visiting Researcher with Lakehead University, Canada. His research interests include big data analysis, software engineering, and autonomous driving.

YAPING LIN (Member, IEEE) received the B.S. degree in computer science from Hunan University, China, in 1982, the M.S. degree in computer applications from the National University of Defense Technology, China, in 1985, and the Ph.D. degree in control theory and its applications from Hunan University, in 2000. He has been a Professor and a Ph.D. Supervisor with Hunan University since 1996. From 2004 to 2005, he was a Visiting Researcher with The University of Texas at Arlington. His research interests include privacy protection, network security, and machine learning.

JIANHAO WEI (Graduate Student Member, IEEE) received the B.S. degree in information and computer science from Shangqiu Normal University, China, in 2014, and the M.S. degree in computer applications from Hunan Normal University, China, in 2017. He is currently pursuing the Ph.D. degree with the College of Computer Science and Electronics Engineering, Hunan University. His research interests include privacy-preserving in cloud computing, privacy issues in spatial crowdsourcing, and big data analysis.

VOUNDI KOE ARTHUR SANDOR received the B.E. degree in network and computer maintenance from the African Institute of Computer Science (IAI) at Cameroon Branch, in 2010, the B.S. degree in fundamental computer science from the University of Yaounde 1, Cameroon, in 2011, and the M.S. degree in computer and application technology from Hunan University, China, in 2015, where he is currently pursuing the Ph.D. degree in network and information security. His research interests include ethical hacking, machine learning, and security and privacy issues in cloud.