Published in IEEE Transactions on Mobile Computing (2023) 22 (6), 3604-3618.
DOI: 10.1109/TMC.2021.3134711

Privacy-Preserving Ranked Spatial Keyword Query in Mobile Cloud-Assisted Fog Computing

Qiuyun Tong, Yinbin Miao, Hongwei Li, *Member, IEEE*, Ximeng Liu, *Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

Abstract—With the increasing popularity of GPS-equipped mobile devices in cloud-assisted fog computing scenarios, massive spatio-textual data is generated and outsourced to cloud servers for storage and analysis. Existing privacy-preserving range query or ranked keyword search schemes does not support a unified index, and are just applicable for the symmetric environment where all users sharing the same secret key. To solve this issue, we propose a Privacy-preserving Ranked Spatial keyword Query in mobile cloud-assisted Fog computing (PRSQ-F). Specifically, we design a novel comparable product encoding strategy that combines both spatial and textual conditions tightly to retrieve the objects in query range and with the highest textual similarity. Then, we use a new conversion protocol and attribute-based encryption to support privacy-preserving retrieval and malicious user traceability in the asymmetric environment where different query users have different keys. Furthermore, we construct an R-tree-based index to achieve faster-than-linear retrieval. Our formal security analysis shows that data security can be guaranteed. Our empirical experiments using a real-world dataset demonstrate the efficiency and feasibility of PRSQ-F.

Index Terms—Mobile cloud-assisted fog computing, spatio-textual data, privacy-preserving, ranked spatial keyword query

1 Introduction

In the mobile cloud-assisted fog computing scenario [1], many GPS-equipped mobile devices are used to provide location-based services such as dating apps, route navigation, and venue recommendation. These service providers tend to outsource their spatio-textual data to cloud-assisted fog servers for cost-saving, flexibility, and low latency. Encryption-before-outsourcing is necessary to ensure data and query confidentiality as honest-but-curious service providers (i.e., cloud and fog servers) can speculate some private information of data owners and data users from the plaintext data, but it complicates data sharing and analysis such as spatial keyword query[1] [2].

Existing schemes [3], [4], [5] have achieved privacy-preserving spatial keyword query (e.g., top-k k-Nearest Neighbor (kNN) query[2] [3], Boolean range query[3] [4], [5]). However, they cannot choose an appropriate textual and spatial similarity ratio to obtain results of interest, or have strict query conditions such that no results are returned. We are committed to a more practical spatial keyword query, namely: Ranked Spatial keyword Query (RSQ). Specifically, in such a query, we retrieve k objects in the query range and with the highest textual similarity. An example using RSQ to find Mr.Right is shown in Fig. 1. A female smartphone user wishes to find a *well-built* and *optimistic* man user inside the region R ("red rectangle"). She sends the query $Q = \{R, W^*\}$ to the cloud server, where $W^* = \{well-built, optimistic\}$ with weights {0.3,0.7}. The cloud server performs RSQ and returns the objects o_1, o_2 as the top-2 search results since o_1, o_2 locate in R and have the textual similarity 0.7 and 0.3, respectively. However, if the top-k kNN query or Boolean range query is used, the female user may obtain an object that is far (o_4) or finds nothing. To achieve RSQ, it is not feasible to combine existing range query [6], [7], [8], [9] and ranked keyword search [10], [11], [12] directly, because there is no effective encoding mechanism to combine spatial and textual conditions together, such that a unified encryption method can be used to encrypt them. Therefore, *the first challenge is how to encode the spatial and textual conditions into a unified index*. In addition, all users in the above

- *Qiuyun Tong and Yinbin Miao are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province, Zhejiang University, Hangzhou 310027, China. E-mail: qytong0820@163.com, ybmiao@xidian.edu.cn.*
- *Hongwei Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China. E-mail: hongweili@uestc.edu.cn.*
- *Ximeng Liu is with the Key Laboratory of Information Security of Network Systems, School of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China. E-mail: snbnix@gmail.com.*
- *Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 26 July 2021; revised 15 November 2021; accepted 7 December 2021. Date of publication 13 December 2021; date of current version 5 May 2023.
This work was supported in part by the National Natural Science Foundation of China under Grants 62072361 and 62125205, in part by the Fundamental Research Funds for the Central Universities under Grant JB211505, in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201917, in part by the Henan Key Laboratory of Network Cryptography Technology & State Key Laboratory of Mathematical Engineering and Advanced Computing under Grant LNCT2020-A06, and in part by the Innovation Fund of Xidian University under Grant YJS2114.
(Corresponding author: Yinbin Miao.)
Digital Object Identifier no. 10.1109/TMC.2021.3134711

1. In this paper, we mainly focus on privacy-preserving spatial keyword query over sensitive spatio-textual data.
2. Top-k kNN query aims to retrieve k objects with the highest scores, measured as a combination of spatial similarity and textual similarity.
3. Boolean range query aims to retrieve all objects located in the query range and containing all queried keywords.

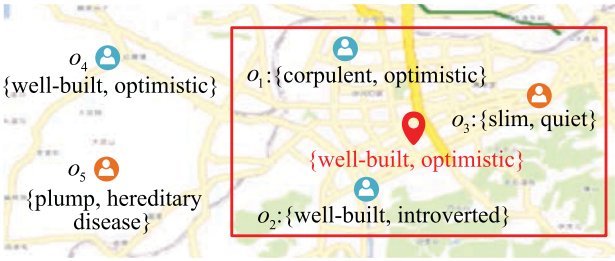

Fig. 1. An example of ranked spatial keyword query.

spatial keyword query schemes [3], [4], [5] share the same secret key, which causes heavy key management burden and even brings key leakage risk in multi-user settings. Obviously, the system must regenerate and distribute a new secret key to ensure data confidentiality when the dating app user logs off his account. Unauthorized access will occur once a dating app user leaks his secret key to unauthorized users. Therefore, *the second challenge is how to deploy* RSQ*to the asymmetric environment where different query users have different secret keys*. To further a step, efficient retrieval is greatly important, especially for large-scale datasets. Therefore, *the third challenge is how to organize the outsourced spatio-textual data, such that the objects not meeting the query condition can be efficiently pruned*.

To solve the above challenging issues, we propose Privacy-preserving Ranked Spatial keyword Query in mobile cloud-assisted Fog computing (PRSQ-F). First, a novel comparable product encoding strategy is designed to encode spatial and textual conditions into a unified index to achieve RSQ in plaintext domain. Then, a conversion protocol and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) are used to support privacy-preserving RSQ in the asymmetric environment. Finally, an R-tree-based index is constructed to achieve sublinear retrieval. Specifically, the main contributions of our work are listed as follows.

- We design a novel comparable product encoding strategy to encode a spatio-textual object and a spatial query as vectors, such that their inner product results can reflect whether the object falls in the query range and deduces its textual similarity simultaneously.
- We design a conversion protocol which converts the multi-key index/trapdoor into the index/trapdoor encrypted with the same key to achieve privacy-preserving RSQ in the asymmetric environment. Moreover, we use Traceable Large Universe CP-ABE (T-LU-CPABE) [13] to generate object decryption keys according to query users' attributes and identities, which are used to decrypt RSQ results and trace malicious query users if they leak their object decryption keys.
- We construct an R-tree-based index to support RSQ with fast-than-linear search complexity, where an intersection decision method is designed to encode each Minimum Boundary Rectangle (MBR) and rectangle query range into vectors, such that the inner product results can determine whether they are intersected, and thereby batching cut the objects that do not meet the query request.

2 Related Work

Secure spatial query. Many schemes have been proposed to support privacy-preserving spatial query such as [6], [7], [8], [9], [14]. The schemes [6], [7], [8], [14] achieved geometric range query based on Shen-Shi-Waters (SSW) encryption or secure kNN. Their search complexities are faster-than-linear due to the construction of the R-tree-based index or inverted list index. The scheme [9] achieved rectangle range query based on secure kNN, which achieves secure linear retrieval in the known background model. However, these schemes only support retrieval in the symmetric environment, where all users (including data owner and query users) share the same secret key.

Secure Multi-Keyword Search. Abundant works have been proposed to achieve multi-keyword search based on secure kNN and Term Frequency-Inverse Document Frequency (TF-IDF) such as [10], [15], [16], [17]. Moreover, [17] supported fine-grained access control by using the polynomial-based access strategy. However, query users in these schemes share the same secret key, which incurs heavy key management burden and brings key leakage risk. To this end, [11], [12], [18] generated different private keys for different query users to support privacy-preserving keyword search in the asymmetric environment, and furthermore, [11], [12] supported fine-grained access control based on CP-ABE [19]. However, query users in [11], [12] can decrypt some ciphertexts if their shared attributes satisfy the ciphertexts' access policies, and their private keys are generated according to commonly shared attributes. Given a private key, there is no feasible method to find out the original key owner. This motivates malicious query users to leak their private keys for financial interest or any other incentive, especially when there is no risk of getting caught. Although, Wang *et al.* [20] adopted T-LU-CPABE to achieve malicious user traceability, the encryption and decryption algorithms based on time-consuming modular exponentiation and pairing operation hinder the use of resource-limited IoT devices. Ma *et al.* [21] proposed an efficient pairing-free dual-server certificateless searchable public-key encryption scheme, which can resist the chosen keyword attack and has better efficiency than the previous schemes.

Spatial Keyword Query. Since Zhou *et al.* [22] first constructed a hybrid index to integrate inverted files and R-tree for both textual and location-aware queries, many spatial keyword query schemes have been proposed to support top-k kNN query [23], [24] and Boolean range query [25], [26]. However, none of the above schemes consider the problem of retrieval over encrypted spatial data. To this end, Su *et al.* [3] used the secure k NN to encrypt indexes and query requests for secure spatio-textual similarity computation, but its similarity is defined as the combination of spatial similarity and textual similarity in a certain ratio. It is difficult for query users to select an appropriate ratio to obtain results of interest. They may retrieve an object matching with textual description but far from them. Cui *et al.* [4] achieved privacy-preserving Boolean range query by designing a spatio-textual Bloom filter encoding method to map the spatial and textual information into their Bloom filters. Wang *et al.* [5] also supported Boolean range query over encrypted spatial data by using Gray code, bitmap and symmetric-key hidden vector encryption technique. However, these schemes only support retrieval in the symmetric environment.

A comparative summary of PRSQ-F and the existing schemes is presented in Table 1.

TABLE 1
Comparison Between Prior Schemes and PRSQ-F

Schemes	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$	$\mathbb{F}_6$
[3]	Top-k kNN query	R-tree	No	No	No	No
[4]	Boolean range query	R-tree	No	No	No	No
[5]	Boolean range query	Quadtree	No	No	No	No
[6]	Range query	R-tree	No	No	No	No
[7]	Range query	Inverted list	No	No	No	No
[11]	Keyword search	Liner	Yes	Yes	No	Yes
[17]	Keyword search	Liner	No	Yes	No	No
[19]	—	—	Yes	Yes	No	Yes
[20]	—	—	Yes	Yes	Yes	No
[23]*	Top-k kNN search	R-tree	—	No	No	No
[25]*	Boolean range query	Quadtree	—	No	No	No
PRSQ-F	RSQ	R-tree	Yes	Yes	Yes	Yes

— $\mathbb{F}_1$: *Search method;* $\mathbb{F}_2$: *Index structure;* $\mathbb{F}_3$: *Not sharing secret key;* $\mathbb{F}_4$: *Supporting fine-grained access control;* $\mathbb{F}_5$: *Malicious user traceability;* $\mathbb{F}_6$: *Retrieval in cloud-assisted fog computing.*
— *: *Retrieval in plaintext domain.*

3 Preliminaries

In this section, we mainly review some related background knowledge, including the bilinear paring, linear secret sharing scheme [27], and secure k-nearest neighbor [28].

3.1 Bilinear Paring

Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map with the following properties:

- *Bilinearity*: $\forall u, v \in \mathbb{G}, a, b \in \mathbb{Z}_p, e(u^a, v^b) = e(u, v)^{ab}$.
- *Non-degeneracy*: $\exists g \in \mathbb{G}, e(g, g) \neq 1$.
- *Computability*: $\forall u, v \in \mathbb{G}$, $e(u, v)$ can be efficiently computed.

3.2 Linear Secret Sharing Scheme

Linear Secret Sharing Scheme (LSSS) over a group of parties $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$ is called linear over $\mathbb{Z}_p$ if

- The share of each party comes from a vector over $\mathbb{Z}_p$.
- For each access structure Γ, there exists a sharing-generating matrix $\mathcal{M} \in \mathbb{Z}_p^{\ell \times \vartheta}$ and a monotone function ρ, such that for the randomly selected vector $\vec{y} = (s, y_2, \ldots, y_\vartheta)^\top \in \mathbb{Z}_p^\vartheta$, $\vec{\lambda} = (\lambda_1, \ldots, \lambda_\ell)^\top = \mathcal{M} \cdot \vec{y}$ is ℓ shares of the secret s. Here, λ_i denotes the share of the party $\rho(i)$.

LSSS has the property of reconstruction requirement. Let S be the attribute set matching the access structure Γ and $\mathcal{I} = \{i : \rho(i) \in S\} (\mathcal{I} \subseteq [\ell])$. Here, $[\ell]$ denotes the value range $1, 2, \ldots, \ell$. Note that the vector $(1, 0, \ldots, 0)$ is in the span of $\mathcal{I}$. If the tuple $\{\lambda_i\}_{i \in \mathcal{I}}$ are valid shares of the secret s, there must exist a set of constants $\{\varpi_i \in \mathbb{Z}_p\}_{i \in \mathcal{I}}$ such that $\sum_{i \in \mathcal{I}} \varpi_i \lambda_i = s$. On the other hand, if S does not match Γ, there exists a vector $\vec{\varpi} = (\varpi_1, \ldots, \varpi_\vartheta)^\top$ satisfying $\varpi_1 = -1$ and $\mathcal{M}_i \cdot \vec{\varpi} = 0$ for all $i \in \mathcal{I}$, where $\mathcal{M}_i$ denotes the ith row of $\mathcal{M}$.

3.3 Secure k-Nearest Neighbor

Secure k-Nearest Neighbor (called kNN) which calculates the vector inner product without revealing privacy. Assuming that $\mathbf{p}$ and $\mathbf{q}$ are two d-dimensional vectors, kNN consisting of four algorithms is shown as follows:

- $\mathsf{KGen}(1^\eta) \rightarrow \mathcal{SK}$: Given the security parameter η, this algorithm generates the secret key $\mathcal{SK} = \{\mathbf{s}, \mathbf{M}, \bar{\mathbf{M}}\}$, where $\mathbf{s}$ is a random d-dimensional bit vector, $\mathbf{M}, \bar{\mathbf{M}}$ are two random $d \times d$ invertible matrices.
- $\mathsf{EncD}(\mathbf{p}, \mathcal{SK}) \rightarrow [\![\mathbf{p}]\!]_{\mathcal{SK}}$: The data vector $\mathbf{p}$ is encrypted as $[\![\mathbf{p}]\!]_{\mathcal{SK}} = (\mathbf{M}^\top \mathbf{p}^a, \bar{\mathbf{M}}^\top \mathbf{p}^b)$, where $\mathbf{p}$ is split into two vectors $\mathbf{p}^a, \mathbf{p}^b$ by using the bit vector $\mathbf{s}$ via Eq. (1).

$$\begin{cases} \mathbf{p}^a[\kappa] = \mathbf{p}^b[\kappa] = \mathbf{p}[\kappa], if\ \mathbf{s}[\kappa] = 0; \\ \mathbf{p}^a[\kappa] + \mathbf{p}^b[\kappa] = \mathbf{p}[\kappa], if\ \mathbf{s}[\kappa] = 1. \end{cases} \tag{1}$$

- $\mathsf{EncQ}(\mathbf{q}, \mathcal{SK}) \rightarrow [\![\mathbf{q}]\!]_{\mathcal{SK}}$: The query vector is encrypted as $[\![\mathbf{q}]\!]_{\mathcal{SK}} = (\mathbf{M}^{-1}\mathbf{q}^a, \bar{\mathbf{M}}^{-1}\mathbf{q}^b)$, where $\mathbf{q}$ is split into two vectors $\mathbf{q}^a, \mathbf{q}^b$ by using $\mathbf{s}$ via Eq. (2).

$$\begin{cases} \mathbf{q}^a[\kappa] + \mathbf{q}^b[\kappa] = \mathbf{q}[\kappa], if\ \mathbf{s}[\kappa] = 0; \\ \mathbf{q}^a[\kappa] = \mathbf{q}^b[\kappa] = \mathbf{q}[\kappa], if\ \mathbf{s}[\kappa] = 1. \end{cases} \tag{2}$$

- $\mathsf{Cal}([\![\mathbf{p}]\!]_{\mathcal{SK}}, [\![\mathbf{q}]\!]_{\mathcal{SK}}) \rightarrow \mathbf{p}^\top \cdot \mathbf{q}$: The inner product of $[\![\mathbf{p}]\!]_{\mathcal{SK}}$ and $[\![\mathbf{q}]\!]_{\mathcal{SK}}$ is

$$\begin{aligned} [\![\mathbf{p}]\!]_{\mathcal{SK}}^\top \cdot [\![\mathbf{q}]\!]_{\mathcal{SK}} =& ((\mathbf{p}^a)^\top \mathbf{M}) \cdot (\mathbf{M}^{-1}\mathbf{q}^a) \\ &+ ((\mathbf{p}^b)^\top \bar{\mathbf{M}}) \cdot (\bar{\mathbf{M}}^{-1}\mathbf{q}^b) = \mathbf{p}^\top \cdot \mathbf{q}. \end{aligned}$$

4 Problem Formulation

We introduce the system model, problem definition, threat model, and privacy requirements of PRSQ-F.

4.1 System Model

In this paper, we consider privacy-preserving data sharing in mobile cloud-assisted fog computing scenario which can be used in most location-based services for real-time response and computation offload. As illustrated in Fig. 2, PRSQ-F mainly consists of five entities: Trusted Authority (TA), Cloud Service Provider (CSP), Fog Servers (FSs), Data Owners (DOs), and Query Users (QUs).

- *Trusted authority*. TA is responsible for generating, distributing, managing keys. Besides, TA uses suspicious keys to trace malicious QUs who leak them.
- *Cloud service provider*. CSP stores the outsourced

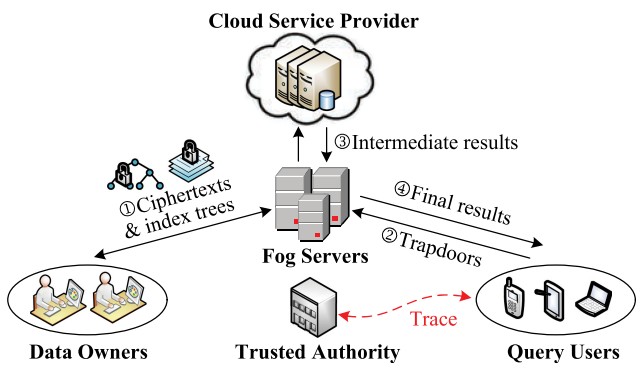

Fig. 2. System model of PRSQ-F.

ciphertexts for DOs and performs RSQ for QUs.

- *Fog servers*. FSs execute partial key decryption and cooperate with DOs to encrypt their objects under the access policies to further reduce the computation overhead of QUs and DOs.
- *Data owners*. Before outsourcing the object ciphertexts, DOs build corresponding index trees.
- *Query users*. For issuing search requests, the resource-limited QUs (e.g., smartphone, tablet) generate trapdoors. Besides, QUs decrypt the final results to obtain top-k object plaintexts.

After receiving an owner key and public parameters from TA, each DO encrypts each spatio-textual object via a symmetric key, encrypts the symmetric key under the access policy by cooperating with the connected FS, encrypts a constructed index tree via the owner key, and sends the ciphertexts (i.e., object ciphertexts, and key ciphertexts, encrypted index tree) to CSP (Step ①). Before storing each index tree, CSP converts the index tree using the corresponding switch key. When an authorized QU wants to issue a query request, he/she first encrypts a spatial query to obtain the trapdoor, then sends it to CSP (Step ②). CSP first converts the trapdoor via the corresponding switch key, then performs RSQ on the index trees to obtain top-k relevance scores, finally returns corresponding k objects' identities to the FS which is connected with the authorized QU (Step ③). The connected FS first finds out the key ciphertexts and object ciphertexts according to the top-k object identities, then partially decrypts the key ciphertexts, finally sends them along with the object ciphertexts to the authorized QU (Step ④). The authorized QU conducts the final decryption via his/her private key to obtain the symmetric keys, and uses them to decrypt the returned object ciphertexts. In addition, if TA finds a suspicious key, TA can trace the malicious QU who leaks it. Note that FSs can be base stations, RSU, or other specially deployed local servers in real life.

4.2 Problem Definition

Since previous spatial keyword search schemes [3], [4], [5] have unreasonable query conditions, heavy key management burdens, and risk of privacy leakage, we hope to design a privacy-preserving RSQ in the asymmetric environment. We cannot directly combine range query schemes and ranked multi-keyword search schemes to achieve RSQ over encrypted spatio-textual data, as there is no effective encoding mechanism to combine spatial and textual conditions together.

For RSQ in plaintext domain, we first encode the spatial point $\mathsf{p} = (\mathsf{x}, \mathsf{y})$ and keyword set W of each object o_i as vectors $\mathbf{v}^x, \mathbf{v}^y$ and $\mathbf{v}_W$ respectively to obtain the object vectors

$$\mathbf{v}_{o_i}^{(1)} = (\mathbf{v}^x, \mathbf{v}_W^{(1)}), \mathbf{v}_{o_i}^{(2)} = (\mathbf{v}^y, \mathbf{v}_W^{(2)}), \tag{3}$$

where $\mathbf{v}_W = \mathbf{v}_W^{(1)} + \mathbf{v}_W^{(2)}$. As for a spatial query $Q = \{R, W^*\}$, the rectangle query range R is represented by its lower-left point (R_1^{ll}, R_2^{ll}) and upper-right point (R_1^{ur}, R_2^{ur}). We encode $R = \{(R_1^{ll}, R_2^{ll}), (R_1^{ur}, R_2^{ur})\}$ and query keyword set W^* as vectors $\mathbf{v}_1^{ll}, \mathbf{v}_2^{ll}, \mathbf{v}_1^{ur}, \mathbf{v}_2^{ur}$ and $\mathbf{v}_{W^*}$ respectively to obtain the query vectors

$$\begin{aligned}\mathbf{v}_Q^{l,1} &= \left(\mathbf{v}_1^{ll}, \mathbf{v}_{W^*}^{(1)}\right), \mathbf{v}_Q^{l,2} = \left(\mathbf{v}_2^{ll}, \mathbf{v}_{W^*}^{(2)}\right),\\ \mathbf{v}_Q^{r,1} &= \left(\mathbf{v}_1^{ur}, \mathbf{v}_{W^*}^{(3)}\right), \mathbf{v}_Q^{r,2} = \left(\mathbf{v}_2^{ur}, \mathbf{v}_{W^*}^{(4)}\right).\end{aligned} \tag{4}$$

Here, $\mathbf{v}_{W^*} = \mathbf{v}_{W^*}^{(1)} + \mathbf{v}_{W^*}^{(3)} = \mathbf{v}_{W^*}^{(2)} + \mathbf{v}_{W^*}^{(4)}$. Then, we check whether the object o_i locates in R by computing the inner product of object vectors and query vectors, namely

$$\begin{aligned}s_{i,1} &= \left(\mathbf{v}_{o_i}^{(1)}\right)^\top \cdot \mathbf{v}_Q^{l,1}, s_{i,2} = \left(\mathbf{v}_{o_i}^{(2)}\right)^\top \cdot \mathbf{v}_Q^{r,1},\\ s_{i,3} &= \left(\mathbf{v}_{o_i}^{(1)}\right)^\top \cdot \mathbf{v}_Q^{l,2}, s_{i,4} = \left(\mathbf{v}_{o_i}^{(2)}\right)^\top \cdot \mathbf{v}_Q^{r,2}.\end{aligned} \tag{5}$$

If $\forall j \in [4], s_{i,j} > 0$, then $R_1^{ll} < x < R_1^{ur}, R_2^{ll} < y < R_2^{ur}$, indicating o_i is in R. Moreover, $sco_i = \sum_{j=1}^{4} s_{i,j}$ is equal to the sum of o_i's textual similarity and a fixed value, thus sco_i can be used to perform textual similarity comparison. For example, if $sco_i - sco_j > 0$, the textual similarity of the object o_i is higher than that of the object o_j. If sco_i is in top-k, the object o_i is a RSQ result.

For privacy-preserving RSQ in the asymmetric environment, we first calls kNN.KGen to obtain the master key $\mathcal{SK} = \{\mathbf{s}, \mathbf{M}, \bar{\mathbf{M}}\}$, then for each each DO or QU we generate a pair of keys (K_1, K_2), where $K_1 = \{\mathbf{s}, \mathbf{M}_1, \bar{\mathbf{M}}_1\}$, $K_2 = \{\mathbf{M}_2, \bar{\mathbf{M}}_2\}$ and $\mathbf{M}_1, \bar{\mathbf{M}}_1, \mathbf{M}_2, \bar{\mathbf{M}}_2$ are four random invertible matrices satisfying $\mathbf{M} = \mathbf{M}_1 \cdot \mathbf{M}_2, \bar{\mathbf{M}} = \bar{\mathbf{M}}_1 \cdot \bar{\mathbf{M}}_2$. Thus, for each object vector or query vector $\mathbf{v}$, we have $[\![\mathbf{v}]\!]_{\mathcal{SK}} = [\![[\![\mathbf{v}]\!]_{K_1}]\!]_{K_2}$. Moreover, the inner product of the encrypted vectors is equal to that of the original vectors, namely

$$\begin{aligned}s_{i,1} &= [\![\mathbf{v}_{o_i}^{(1)}]\!]_{\mathcal{SK}}^\top \cdot [\![\mathbf{v}_Q^{l,1}]\!]_{\mathcal{SK}}, s_{i,2} = [\![\mathbf{v}_{o_i}^{(2)}]\!]_{\mathcal{SK}}^\top \cdot [\![\mathbf{v}_Q^{r,1}]\!]_{\mathcal{SK}},\\ s_{i,3} &= [\![\mathbf{v}_{o_i}^{(1)}]\!]_{\mathcal{SK}}^\top \cdot [\![\mathbf{v}_Q^{l,2}]\!]_{\mathcal{SK}}, s_{i,4} = [\![\mathbf{v}_{o_i}^{(2)}]\!]_{\mathcal{SK}}^\top \cdot [\![\mathbf{v}_Q^{r,2}]\!]_{\mathcal{SK}}.\end{aligned} \tag{6}$$

Let $\mathtt{RSQ}(\mathcal{O}, Q)$ be a set of k objects' identities $\{id_\varrho\}_{\varrho \in [k]}$ satisfying $\forall j \in [4], s_{i,j} > 0, sco_i \in$ top-k. We encrypt each object o_i under its access policy Γ (i.e., $\mathtt{Enc}_\Gamma(o_i)$). We generate the private key $SK_{ID,S}$ for each QU according to his identity ID and attribute set S. If the attribute set S satisfies the access policy Γ, QU can use his private key $SK_{ID,S}$ to decrypt the object ciphertext, i.e., $o_i = \mathtt{Dec}_{SK_{ID,S}}(\mathtt{Enc}_\Gamma(o_i))$. If QU leaks $SK_{ID,S}$ to others, TA can trace the malicious QU according to the identity included in $SK_{ID,S}$.

To this end, the definition of PRSQ-F can be given as follows.

Definition 1 (PRSQ-F). *An object $o_i = \{\mathsf{p}, \mathsf{W}\}$ encrypted under the access policy Γ is an accessible RSQ result for QU with spatial query $Q = \{R, W^*\}$ and attribute set S, if $id_{o_i} \in \mathtt{RSQ}(\mathcal{O}, Q)$ and $\mathtt{Dec}_{SK_{ID,S}}(\mathtt{Enc}_\Gamma(o_i)) = o_i$.*

4.3 Threat Model

Similar to most keyword or range query schemes, TA and DOs are considered to be fully trusted, while CSP and FSs are considered as honest-but-curious entities who honestly follow the established protocol but are sufficiently curious to infer some valuable information from the spatio-textual data. Malicious authorized QUs may leak their private keys to unauthorized QUs for profits. Besides, all communications with TA are assumed to be secure and QUs cannot collude with CSP. We consider a threat model: CSP or FS has access to ciphertexts and trapdoors, and can deduce specific contents in a spatial query based on the statistical information, but cannot obtain the plaintext-ciphertext pairs.

TABLE 2
Notations

Notation	Definition	Notation	Definition
$[\ell]$	$1, 2, \ldots, \ell$	$Q = \{R, W^*\}$	spatial query
Γ	access tree	$o_i = \{p, W\}$	spatio-textual object
$[\![\cdot]\!]$	ciphertext	(R_1^{ll}, R_2^{ll})	lower-left point of R
m	number of objects	(R_1^{ur}, R_2^{ur})	upper-right point of R
d_2	size of keyword set	$d = d_1 + d_2$	size of object vector
I_ϕ	index tree	(M_1^{ll}, M_2^{ll})	lower-left point of MBR
S	user attribute set	(M_1^{ur}, M_2^{ur})	upper-right point of MBR

4.4 Privacy Requirements

PRSQ-F must meet the following privacy requirements under the above threat model.

- *Data security*. PRSQ-F should protect object confidentiality from unauthorized entities and keep available for authorized users.
- *Index confidentiality and query confidentiality*. PRSQ-F should prevent CSP from using the index trees and trapdoors to obtain valid knowledge except access pattern and search pattern.
- *Trapdoor unlinkability*. PRSQ-F should prevent CSP from inferring whether two trapdoors are from the same spatial query.

5 Proposed PRSQ-F

In this section, we first introduce how to achieve RSQ in plaintext domain, then specify the concrete construction of PRSQ-F. Table 2 gives some important notations used in the following paper.

5.1 RSQ in Plaintext Domain

RSQ aims to return the objects inside the query range and with top-k textual similarity. According to Fig. 3, if $R_1^{ll} < x < R_1^{ur}, R_2^{ll} < y < R_2^{ur}$ hold, the spatial point (x, y) is inside the query range R. For textual similarity, coordinate matching [10] and TF-IDF [16] are two popular similarity measure methods used in keyword search schemes. Compared with coordinate matching, TF-IDF can capture more semantic information, thereby in our scheme we choose TF-IDF to measure the relevance of object textual description to query keywords. The key of RSQ is to encode both spatial and textual conditions into a unified index. To solve this, we design a comparable product encoding strategy to encode a spatio-textual object

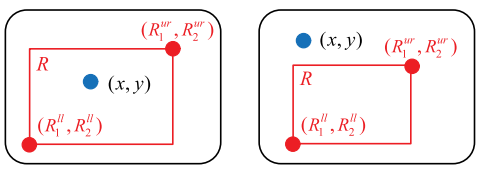

Fig. 3. All possible situations between a spatial point and a rectangle query range.

and a spatial query as vectors, where each vector includes both spatial information and textual information. The main idea of the comparable product encoding strategy is to ensure the sign of $x - R_1^{ll}$ (or $R_1^{ur} - x, y - R_2^{ll}, R_2^{ur} - y$) is the same as that of the corresponding vector inner product. Moreover, the sum of four inner products can reflect the textual similarity.

Algorithm 1. Spatio-Textual Object Encoding

Input: Object $o_i = \{\mathsf{p}, \mathsf{W}\}$, a d_1-dimensional bit vector $\mathbf{s}'$
Output: Object vector set $\mathbf{v}_{o_i}$
for $\iota = 1; \iota \le d_1; \iota{+}{+}$ **do**
  if $\mathbf{s}'[\iota] == 1$ **then**
    $\mathbf{v}^x[\iota] = x$, $\mathbf{v}^y[\iota] = y$;
  else
    $\mathbf{v}^x[\iota] = 1$, $\mathbf{v}^y[\iota] = 1$;
for $\iota = 1; \iota \le d_2; \iota{+}{+}$ **do**
  Select a random real number $\varepsilon_\iota \in (0, 1)$;
  $\mathbf{v}_W^{(1)}[\iota] = \varepsilon_\iota$, $\mathbf{v}_W^{(2)}[\iota] = \mathbf{v}_W[\iota] - \varepsilon_\iota$;
$\mathbf{v}_{o_i}^{(1)} = (\mathbf{v}^x, \mathbf{v}_W^{(1)})$, $\mathbf{v}_{o_i}^{(2)} = (\mathbf{v}^y, \mathbf{v}_W^{(2)})$;
return $\mathbf{v}_{o_i} = \{\mathbf{v}_{o_i}^{(1)}, \mathbf{v}_{o_i}^{(2)}\}$.

Algorithm 2. Query Encoding

Input: Spatial query $Q = \{([R_1^{ll}, R_1^{ur}], [R_2^{ll}, R_2^{ur}]), W^*\}$, a d_1-dimensional bit vector $\mathbf{s}'$
Output: Query vector sets $\mathbf{v}_Q^l, \mathbf{v}_Q^r$
Select a random number μ;
for $\kappa \in \{1, 2\}$ **do**
  Select $d_1/2$ random numbers $\mu_{\kappa,1}, \ldots, \mu_{\kappa,d_1/2}$ such that $\delta_\kappa \ge \sum_{\iota=1}^{d_1/2} \mu_{\kappa,\iota} \ge d_2/MD > 0$;
  Set $i = j = 1$;
  for $\iota = 1; \iota \le d_1; \iota{+}{+}$ **do**
    if $\mathbf{s}'[\iota] == 1$ **then**
      $\mathbf{v}_\kappa^{ll}[\iota] = \mu_{\kappa,i}$, $\mathbf{v}_\kappa^{ur}[\iota] = -\mu_{\kappa,i}, i = i + 1$;
    else
      $\mathbf{v}_\kappa^{ll}[\iota] = -b_\kappa^{ll}\mu_{\kappa,j}$, $\mathbf{v}_\kappa^{ur}[\iota] = b_\kappa^{ur}\mu_{\kappa,j}, j = j + 1$;
  for $\iota = 1; \iota \le d_2; \iota{+}{+}$ **do**
    Select a random real number $\epsilon_{\kappa,\iota} \in (0, 1)$;
    $\mathbf{v}_{W^*}^{\kappa,1}[\iota] = \mu * \epsilon_{\kappa,\iota}$, $\mathbf{v}_{W^*}^{\kappa,2}[\iota] = \mu * (\mathbf{v}_{W^*}[\iota] - \epsilon_{\kappa,\iota})$;
$\mathbf{v}_Q^{l,1} = (\mathbf{v}_1^{ll}, \mathbf{v}_{W^*}^{1,1})$, $\mathbf{v}_Q^{l,2} = (\mathbf{v}_2^{ll}, \mathbf{v}_{W^*}^{2,1})$;
$\mathbf{v}_Q^{r,1} = (\mathbf{v}_1^{ur}, \mathbf{v}_{W^*}^{1,2})$, $\mathbf{v}_Q^{r,2} = (\mathbf{v}_2^{ur}, \mathbf{v}_{W^*}^{2,2})$;
return $\mathbf{v}_Q^l = \{\mathbf{v}_Q^{l,1}, \mathbf{v}_Q^{l,2}\}, \mathbf{v}_Q^r = \{\mathbf{v}_Q^{r,1}, \mathbf{v}_Q^{r,2}\}$.

The comparable product encoding strategy consists of two algorithms: spatio-textual object encoding (Algorithm 1) and query encoding (Algorithm 2). In Algorithm 1, each object $o_i = \{\mathsf{p}, \mathsf{W}\}$ is encoded into two object vectors $\mathbf{v}_{o_i}^{(1)}, \mathbf{v}_{o_i}^{(2)}$. For the spatial point $\mathsf{p} = (\mathsf{x}, \mathsf{y})$, we encode latitude x and longitude y into two d_1-dimensional spatial vectors $\mathbf{v}^x, \mathbf{v}^y$, where d_1 is an even number. For keyword description W, we use TF-IDF [10] to construct a d_2-dimensional textual vector $\mathbf{v}_W$ by Eq. (7).

$$\mathbf{v}_W[\iota] = \begin{cases} tf_\iota, & if\ w_\iota \in \mathcal{W} \\ 0, & if\ w_\iota \notin \mathcal{W} \end{cases} \quad \iota \in [d_2], \tag{7}$$

where d_2 is the size of textual keyword set $\mathcal{W}$ extracted from m objects and $tf_\iota = (1 + \ln n_\iota)/\sum_{w_\iota \in W} \sqrt{(1 + \ln n_\iota)^2}$ is the normalized term frequency of the keyword w_ι. If d_2 is an odd number, we add a dummy keyword to $\mathcal{W}$ so that d_2 is an even number. Here, n_ι is the number of times the keyword w_ι

appearing in o_i. Then, we split $\mathbf{v}_W$ into two random vectors $\mathbf{v}_W^{(1)}, \mathbf{v}_W^{(2)}$ to increase randomness and mask the real distance. Finally, combining the spatial vectors with corresponding textual vectors, we obtain $\mathbf{v}_{o_i}^{(1)} = (\mathbf{v}^x, \mathbf{v}_W^{(1)}), \mathbf{v}_{o_i}^{(2)} = (\mathbf{v}^y, \mathbf{v}_W^{(2)})$.

In Algorithm 2, the spatial query $Q = \{R, W^*\}$ is encoded into the query vector sets $\mathbf{v}_Q^l, \mathbf{v}_Q^r$. To obtain the range query results and textual relevance scores simultaneously, for each dimension (i.e., latitude or longitude) we first generate $d_1/2$ random numbers $\mu_{\kappa,1}, \mu_{\kappa,2}, \ldots, \mu_{\kappa,d_1/2}$ satisfying $\sum_{\iota=1}^{d_1/2} \mu_{\kappa,\iota} \geq d_2/MD$, where $\kappa \in [2]$ and MD is the minimum value of the horizontal or vertical distance between the spatial points and the range query R. Then, we select a random number δ_1 satisfying $\delta_1 \geq \sum_{\iota=1}^{d_1/2} \mu_{1,\iota}$ for the latitude range $[R_1^{ll}, R_1^{ur}]$ and encode the endpoints R_1^{ll}, R_1^{ur} into two d_1-dimensional endpoint vectors $\mathbf{v}_1^{ll}, \mathbf{v}_1^{ur}$. Similarly, the longitude range $[R_2^{ll}, R_2^{ur}]$ is also encoded into two d_1-dimensional endpoint vectors $\mathbf{v}_2^{ll}, \mathbf{v}_2^{ur}$. According to TF-IDF and QU's preference, QU transforms his/her query keyword set W^* into a query keyword vector $\mathbf{v}_{W^*}$. For each keyword $w_\iota \in \mathcal{W}$, if $w_\iota \in W^*$ holds, QU gives it a preference degree $\zeta_\iota \in [10]$, computes its weight ω_ι in Eq. (8) satisfying the sum of these weights equal to 1, and sets $\mathbf{v}_{W^*}[\iota] = \omega_\iota$; otherwise, QU sets $\mathbf{v}_{W^*}[\iota] = 0$. Here, f_ι denotes the number of objects containing the keyword w_ι.

$$\omega_\iota = \frac{\zeta_\iota \ln(1 + m/f_\iota)}{\sum_{w_\iota \in W^*} \zeta_\iota \ln(1 + m/f_\iota)}, \tag{8}$$

Then, for latitude or longitude range, we split $\mathbf{v}_{W^*}$ into two random vectors $\mathbf{v}_{W^*}^{\kappa,1}, \mathbf{v}_{W^*}^{\kappa,2}$ and combine them with corresponding endpoint vectors (i.e., $\mathbf{v}_Q^{l,\kappa} = (\mathbf{v}_\kappa^{ll}, \mathbf{v}_{W^*}^{\kappa,1})), \mathbf{v}_Q^{r,\kappa} = (\mathbf{v}_\kappa^{ur}, \mathbf{v}_{W^*}^{\kappa,2})$ to obtain the query vector sets $\mathbf{v}_Q^l = \{\mathbf{v}_Q^{l,1}, \mathbf{v}_Q^{l,2}\}, \mathbf{v}_Q^r = \{\mathbf{v}_Q^{r,1}, \mathbf{v}_Q^{r,2}\}$. Note that the number of 0s is equal to the number 1s in $\mathbf{s}'$.

Algorithm 3. Ranked Spatial Keyword Query

Input: Object vectors $\mathbf{v}_{o_i}^{(1)}, \mathbf{v}_{o_i}^{(2)}$, query vectors $\mathbf{v}_Q^l, \mathbf{v}_Q^r$
Output: Relevance score sco_i or $null$

1: $\{s_{i,1}, s_{i,2}\} = \mathbf{v}_{o_i}^\top \circ \mathbf{v}_Q^l = \{(\mathbf{v}_{o_i}^{(1)})^\top \cdot \mathbf{v}_Q^{l,1}, (\mathbf{v}_{o_i}^{(2)})^\top \cdot \mathbf{v}_Q^{l,2}\} = \{\delta_1(x - R_1^{ll}) + \alpha_1, \delta_2(y - R_2^{ll}) + \alpha_2\}$;
2: $\{s_{i,3}, s_{i,4}\} = \mathbf{v}_{o_i}^\top \circ \mathbf{v}_Q^r = \{(\mathbf{v}_{o_i}^{(1)})^\top \cdot \mathbf{v}_Q^{r,1}, (\mathbf{v}_{o_i}^{(2)})^\top \cdot \mathbf{v}_Q^{r,2}\} = \{\delta_1(R_1^{ur} - x) + \alpha_3, \delta_2(R_2^{ur} - y) + \alpha_4\}$;
3: **if** $(s_{i,1} > 0) \wedge (s_{i,2} > 0) \wedge (s_{i,3} > 0) \wedge (s_{i,4} > 0)$ **then**
4: $\quad sco_i = s_{i,1} + s_{i,2} + s_{i,3} + s_{i,4} = \delta_1 \times (R_1^{ur} - R_1^{ll}) + \delta_2 \times (R_2^{ur} - R_2^{ll}) + \mu \times \mathbf{v}_W^\top \cdot \mathbf{v}_{W^*}$;
5: $\quad$ **return** sco_i;
6: **else**
7: $\quad$ **return** $null$.

Algorithm 3 shows the correctness of the comparable product encoding strategy. We calculate the inner product of the object vectors $\mathbf{v}_{o_i}^{(1)}, \mathbf{v}_{o_i}^{(2)}$ and query vectors $\mathbf{v}_Q^l, \mathbf{v}_Q^r$ to obtain $s_{i,1}, s_{i,2}, s_{i,3}, s_{i,4}$, where

$$\alpha_1 = \mu * \sum_{\iota=1}^{d_2} \varepsilon_\iota \epsilon_{1,\iota}, \alpha_2 = \mu * \sum_{\iota=1}^{d_2} (\mathbf{v}_W[\iota] - \varepsilon_\iota)\epsilon_{2,\iota},$$
$$\alpha_3 = \mu * \sum_{\iota=1}^{d_2} \varepsilon_\iota (\mathbf{v}_{W^*}[\iota] - \epsilon_{1,\iota}),$$
$$\alpha_4 = \mu * \sum_{\iota=1}^{d_2} (\mathbf{v}_W[\iota] - \varepsilon_\iota)(\mathbf{v}_{W^*}[\iota] - \epsilon_{2,\iota}). \tag{9}$$

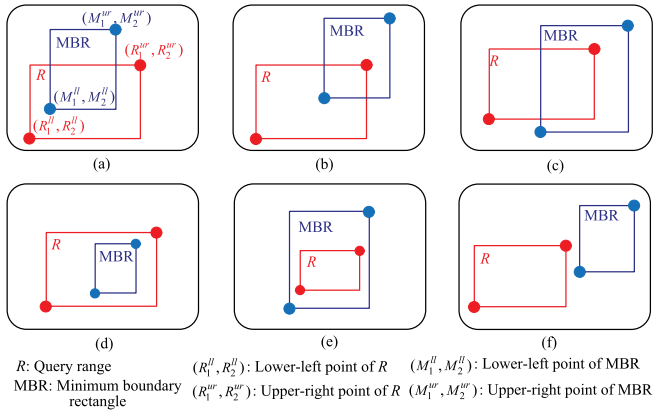

Fig. 4. All possible situations between a rectangle query range and an MBR.

We have the object o_i within the query range R if $s_{i,1} > 0, s_{i,2} > 0, s_{i,3} > 0, s_{i,4} > 0$. The proof is shown as follows: Setting $x - R_1^{ll}, y - R_2^{ll}, R_1^{ur} - x, R_2^{ur} - y$ as a_1, a_2, a_3, a_4 respectively, we have $0 < MD \leq |a_\iota|$ for $\iota \in [4]$ according to the definition of MD. Thus, $\delta_1, \delta_2 \geq d_2/MD \geq d_2/|a_\iota|$. Also, $\alpha_\iota < d_2$ for $\iota \in [4]$. Therefore, if $a_\iota < 0$, then $\delta_\kappa a_\iota + \alpha_\iota \leq -d_2 + \alpha_\iota < 0$; otherwise, $\delta_\kappa a_\iota + \alpha_\iota > 0$. That it to say, the sign of a_ι is the same as that of s_ι. If $s_{i,1} > 0, s_{i,3} > 0, s_{i,2} > 0, s_{i,4} > 0$, then $x - R_1^{ll} > 0, R_1^{ur} - x > 0, y - R_2^{ll} > 0, R_2^{ur} - y > 0$, i.e., $R_1^{ll} < x < R_1^{ur}, R_2^{ll} < y < R_2^{ur}$.

If the object o_i is inside R, we compute $s_{i,1} + s_{i,2} + s_{i,3} + s_{i,4}$ to obtain its relevance score sco_i. As $\delta_1(R_1^{ur} - R_1^{ll}) + \delta_2(R_2^{ur} - R_2^{ll})$ is the same for all objects during one query and $\mu > 0$, we can obtain k RSQ results by selecting top-k relevance scores of the objects inside R. Unfortunately, the above search complexity linearly increases with the size of the spatio-textual dataset, which is difficult to meet the needs of large-scale datasets. To perform efficient retrieval, we construct an R-tree-based index, where each non-leaf node represents an MBR and each leaf node represents an object. The search process on R-tree is described as follows:

1) Starting from the root node, if the current node intersects with the rectangle query range R, CSP retrieves its child nodes.
2) When traversing to a leaf node, CSP performs Algorithm 3 to check whether the object is likely to be a RSQ result.

Let MBR be represented by its lower-left and upper-right points $\{(M_1^{ll}, M_2^{ll}), (M_1^{ur}, M_2^{ur})\}$. According to Fig. 4, if $M_1^{ll} < R_1^{ur}, M_2^{ll} < R_2^{ur}, R_1^{ll} < M_1^{ur}, R_2^{ll} < M_2^{ur}$ hold, MBR intersects with the rectangle query range R. To check the intersection of MBR and R, an intersection decision method is designed to encode MBR and R into vectors such that the sign of $R_1^{ur} - M_1^{ll}$ (or $R_2^{ur} - M_2^{ll}, M_1^{ur} - R_1^{ll}, M_2^{ur} - R_2^{ll}$) is the same as that of corresponding vector inner product.

The intersection decision method includes two encoding algorithms: lower-left encoding (Algorithm 4) and upper-right encoding (Algorithm 5). Algorithm 4 encodes the lower-left points $(M_1^{ll}, M_2^{ll}), (R_1^{ll}, R_2^{ll})$ as $\mathbf{v}_{\mathsf{MBR}}^l = \{\mathbf{v}_{\mathsf{MBR}}^{l,1}, \mathbf{v}_{\mathsf{MBR}}^{l,2}\}, \mathbf{v}_R^l = \{\mathbf{v}_R^{l,1}, \mathbf{v}_R^{l,2}\}$ respectively, where $d = d_1 + d_2$ can be divisible by 4. Algorithm 5 encodes the upper-right points $(M_1^{ur}, M_2^{ur}), (R_1^{ur}, R_2^{ur})$ as $\mathbf{v}_{\mathsf{MBR}}^r =$

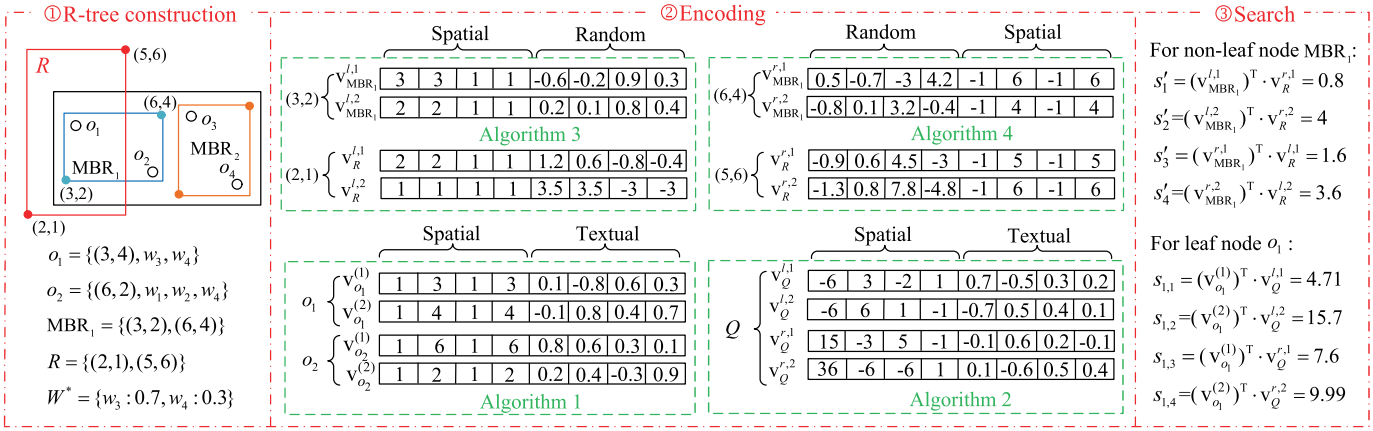

Fig. 5. A toy example of PRSQ-F in plaintext domain.

$\{\mathbf{v}^{r,1}_{\mathsf{MBR}}, \mathbf{v}^{r,2}_{\mathsf{MBR}}\}, \mathbf{v}^r_R = \{\mathbf{v}^{r,1}_R, \mathbf{v}^{r,2}_R\}$, respectively. Then, the inner product of lower-left vectors and upper-right vectors is calculated as follows:

$$\begin{aligned}\{s'_1, s'_2\} &= (\mathbf{v}^l_{\mathsf{MBR}})^\top \circ \mathbf{v}^r_R = \{(\mathbf{v}^{l,1}_{\mathsf{MBR}})^\top \cdot \mathbf{v}^{r,1}_R, (\mathbf{v}^{l,2}_{\mathsf{MBR}})^\top \cdot \mathbf{v}^{r,2}_R\} \\ &= \{(\pi_1 + \overline{\pi}'_1)(R^{ur}_1 - M^{ll}_1), (\pi_2 + \overline{\pi}'_2)(R^{ur}_2 - M^{ll}_2)\}; \\ \{s'_3, s'_4\} &= (\mathbf{v}^r_{\mathsf{MBR}})^\top \circ \mathbf{v}^l_R = \{(\mathbf{v}^{r,1}_{\mathsf{MBR}})^\top \cdot \mathbf{v}^{l,1}_R, (\mathbf{v}^{r,2}_{\mathsf{MBR}})^\top \cdot \mathbf{v}^{l,2}_R\} \\ &= \{(\pi'_1 + \overline{\pi}_1)(M^{ur}_1 - R^{ll}_1), (\pi'_2 + \overline{\pi}_2)(M^{ur}_2 - R^{ll}_2)\};\end{aligned}$$

Due to $\pi_1, \pi_2, \overline{\pi}'_1, \overline{\pi}'_2 > 0$, $s'_1 > 0, s'_2 > 0$ is equal to $M^{ll}_1 < R^{ur}_1, M^{ll}_2 < R^{ur}_2$. Similarly, $s'_3 > 0, s'_4 > 0$ is equal to $R^{ll}_1 < M^{ur}_1, R^{ll}_2 < M^{ur}_2$ due to $\pi'_1, \pi'_2, \overline{\pi}_1, \overline{\pi}_2 > 0$. Therefore, if $s'_1 > 0, s'_2 > 0, s'_3 > 0, s'_4 > 0$ hold, the non-leaf node representing by MBR intersects with R. It worth noting that the number of 0s is equal to the number of 1s in the bit vector $\mathbf{s}_1$ (or $\mathbf{s}_2$).

Algorithm 4. Lower-Left Encoding

Input: Lower-left point (v^{ll}_1, v^{ll}_2), two $d/2$-dimensional bit vectors $\mathbf{s}_1, \mathbf{s}_2$
Output: Lower-left vectors $\mathbf{v}^l$

for $\kappa = 1; \kappa \le 2; \kappa{+}{+}$ **do**
  for $\iota = 1; \iota \le \frac{d}{2}; \iota{+}{+}$ **do**
    if $\mathbf{s}_1[\iota] == 1$ **then**
      $\mathbf{v}^{l,\kappa}[\iota] = v^{ll}_\kappa$;
    else
      $\mathbf{v}^{l,\kappa}[\iota] = 1$;
  Select $d/4$ random numbers $\xi_{\kappa,1}, \ldots, \xi_{\kappa,d/4}$ such that $\pi_\kappa = \sum_{\iota=1}^{d/4} \xi_{\kappa,\iota} > 0$;
  Set $i = j = 1$;
  for $\iota = 1; \iota \le d/2; \iota{+}{+}$ **do**
    if $\mathbf{s}_2[\iota] == 1$ **then**
      $\mathbf{v}^{l,\kappa}[\iota + d/2] = g^{ll}_\kappa \times \xi_{\kappa,i},\ i = i + 1$;
    else
      $\mathbf{v}^{l,\kappa}[\iota + d/2] = \xi_{\kappa,j},\ j = j + 1$;
return $\mathbf{v}^l = \{\mathbf{v}^{l,1}, \mathbf{v}^{l,2}\}$.

Example. Fig. 5 gives a toy example of RSQ in plaintext domain, where an R-tree is constructed over the dataset $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$ and $d_1 = 4, d_2 = 4$. For MBR_1 in the non-leaf node, we run *Algorithm 4* to encode its lower-left point (3,2) into two vectors $\mathbf{v}^{l,1}_{\mathsf{MBR}_1}, \mathbf{v}^{l,2}_{\mathsf{MBR}_1}$, and run Algorithm 5 to encode its upper-right point (6,4) into two vectors $\mathbf{v}^{r,1}_{\mathsf{MBR}_1}, \mathbf{v}^{r,2}_{\mathsf{MBR}_1}$. The rectangle query range $R = \{(2,1),(5,6)\}$ is processed in the same way to obtain $\mathbf{v}^{l,1}_R, \mathbf{v}^{l,2}_R, \mathbf{v}^{r,1}_R, \mathbf{v}^{r,2}_R$. For the object o_1, we run Algorithm 1 to encode it into two object vectors $\mathbf{v}^{(1)}_{o_1}, \mathbf{v}^{(2)}_{o_1}$. For the spatial query $Q = \{R, W^*\}$, we run Algorithm 2 to encode it into query vectors $\mathbf{v}^{l,1}_Q, \mathbf{v}^{l,2}_Q, \mathbf{v}^{r,1}_Q, \mathbf{v}^{r,2}_Q$, where $\delta_1 = 4, \delta_2 = 5, \mu = 1$. When performing RSQ, for the non-leaf node MBR_1 we find $s'_\iota > 0 (\forall \iota \in [4])$, meaning that MBR_1 intersects with R. Then, we check its child nodes. For the object o_1 under MBR_1, we compute the inner product of object vectors $\mathbf{v}^{(1)}_{o_1}, \mathbf{v}^{(2)}_{o_1}$ and query vectors $\mathbf{v}^{l,1}_Q, \mathbf{v}^{l,2}_Q, \mathbf{v}^{r,1}_Q, \mathbf{v}^{r,2}_Q$ to obtain $s_{1,\iota} > 0 (\forall \iota \in [4])$, which means that o_1 falls inside R. Moreover, o_1's relevance score is $sco_1 = \sum_{\iota=1}^4 s_{1,\iota} = 4 \times (5-2) + 5 \times (6-1) + 1 = 38$.

Algorithm 5. Upper-Right Encoding

Input: Upper-right point (v^{ur}_1, v^{ur}_2), two $d/2$-dimensional bit vectors $\mathbf{s}_1, \mathbf{s}_2$
Output: Upper-right vectors $\mathbf{v}^r$

for $\kappa = 1; \kappa \le 2; \kappa{+}{+}$ **do**
  Select $d/4$ random numbers $\overline{\xi}_{\kappa,1}, \ldots, \overline{\xi}_{\kappa,d/4}$ such that $\overline{\pi}_\kappa = \sum_{\iota=1}^{d/4} \overline{\xi}_{\kappa,\iota} > 0$;
  Set $i = j = 1$;
  for $\iota = 1; \iota \le d/2; \iota{+}{+}$ **do**
    if $\mathbf{s}_1[\iota] == 1$ **then**
      $\mathbf{v}^{r,\kappa}[\iota] = -\overline{\xi}_{\kappa,i},\ i = i + 1$;
    else
      $\mathbf{v}^{r,\kappa}[\iota] = v^{ur}_\kappa \times \overline{\xi}_{\kappa,j},\ j = j + 1$;
  for $\iota = 1; \iota \le d/2; \iota{+}{+}$ **do**
    if $\mathbf{s}_2[\iota] == 1$ **then**
      $\mathbf{v}^{r,\kappa}[\iota + d/2] = -1$;
    else
      $\mathbf{v}^{r,\kappa}[\iota + d/2] = v^{ur}_\kappa$;
return $\mathbf{v}^r = \{\mathbf{v}^{r,1}, \mathbf{v}^{r,2}\}$.

5.2 Concrete Construction of PRSQ-F

In this section, PRSQ-F combines the designed conversion protocol and kNN to obtain encrypted RSQ results in the asymmetric environment. Moreover, PRSQ-F employs T-LU-CPABE to generate object decryption keys according to QUs' attributes and identities, which achieves malicious user traceability and decryption of the RSQ result ciphertexts without sharing the same secret key. Since the ciphertext generation and decryption algorithms in T-LU-CPABE are computationally expensive for resource-limited client devices, we delegate large computation tasks of DOs or

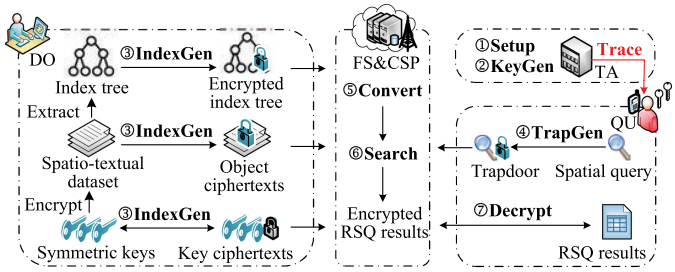

Fig. 6. Framework of PRSQ-F.

QUs to FSs without losing data confidentiality. The framework of PRSQ-F is shown in Fig. 6, which includes the following eight algorithms.

Setup (1^η). Given the security parameter η, TA first runs the group generator algorithm $\mathcal{G}(1^\eta)$ to obtain a bilinear mapping $\Theta = (\mathbb{G}, \mathbb{G}_T, p, e)$, then randomly chooses $g, u, \varrho, \nu \in \mathbb{G}$ and $\alpha, \beta, \gamma \in \mathbb{Z}_p$. TA also selects a probabilistic encryption scheme (Enc, Dec) [29] from a binary string to $\mathbb{Z}_p^*$ with two different secret keys $\hat{k}_1, \hat{k}_2$. Furthermore, TA initializes an instance of Shamir's (t, n) threshold secret sharing $INS_{(t,n)}$ [30], and keeps the polynomial $f(x)$ and $t-1$ points $\{(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\}$ secret. The public parameters PP and the master key MSK are generated as

$$PP = \{\Theta, g, u, \varrho, \nu, g^\beta, g^\gamma, e(g,g)^\alpha\};$$
$$MSK = \{\alpha, \beta, \gamma, \hat{k}_1, \hat{k}_2\}. \tag{10}$$

KeyGen (PP, MSK, S, d). Given the dimension d of the object vector and the user attribute set S, TA generates an owner key and corresponding switch key for each DO, a user key, corresponding switch key and a private key for each QU, as well as a private key for QU's connected FS. Note that the key generation is completed in the system initialization and user registration stages, which will not affect the efficiency of RSQ.

- TA calls the algorithm kNN.KGen to obtain the secret key $\mathcal{SK} = \{\mathbf{s}, \mathbf{M}, \bar{\mathbf{M}}\}$.
- Then, for each DO O_ϕ, TA generates an owner key $OK_{O_\phi} = \{\mathbf{s}, \mathbf{M}_{O_\phi}, \bar{\mathbf{M}}_{O_\phi}\}$ and its switch key $SK_{O_\phi} = \{\mathbf{M}'_{O_\phi}, \bar{\mathbf{M}}'_{O_\phi}\}$, where $\mathbf{M}_{O_\phi}, \mathbf{M}'_{O_\phi}, \bar{\mathbf{M}}_{O_\phi}, \bar{\mathbf{M}}'_{O_\phi}$ are $d \times d$ matrices satisfying $\mathbf{M} = \mathbf{M}_{O_\phi} \cdot \mathbf{M}'_{O_\phi}$ and $\bar{\mathbf{M}} = \bar{\mathbf{M}}_{O_\phi} \cdot \bar{\mathbf{M}}'_{O_\phi}$.
- Similarly, for each QU U_θ, TA generates a user key $UK_{U_\theta} = \{\mathbf{s}, \mathbf{M}_{U_\theta}, \bar{\mathbf{M}}_{U_\theta}\}$ and its switch key $SK_{U_\theta} = \{\mathbf{M}'_{U_\theta}, \bar{\mathbf{M}}'_{U_\theta}\}$, where $\mathbf{M}_{U_\theta}, \mathbf{M}'_{U_\theta}, \bar{\mathbf{M}}_{U_\theta}, \bar{\mathbf{M}}'_{U_\theta}$ are also $d \times d$ matrices satisfying $\mathbf{M}^{-1} = \mathbf{M}'_{U_\theta} \cdot \mathbf{M}_{U_\theta}$ and $\bar{\mathbf{M}}^{-1} = \bar{\mathbf{M}}'_{U_\theta} \cdot \bar{\mathbf{M}}_{U_\theta}$.
- Let id_θ be the identity of QU U_θ and $S = \{A_1, \ldots, A_{|S|}\} \subseteq \mathbb{Z}_p$ be the attribute set of QU U_θ. TA calculates $\varsigma = Enc_{\hat{k}_1}(id_\theta), c = Enc_{\hat{k}_2}(\varsigma || f(\varsigma))$ and randomly chooses $r, r_1, \ldots, r_{|S|} \in \mathbb{Z}_p$ to generate the private keys $sk_{U_\theta,S} = K_0 = g^{(\alpha+(\gamma+c)r)/\beta}, sk_{F_\theta,S}$ for U_θ and its connected FS F_θ respectively, where

$$sk_{F_\theta,S} = (K' = c, L = g^r, L' = g^{\gamma r}, \{K_{j,1} = g^{r_j},$$
$$K_{j,2} = (u^{A_j}\varrho)^{r_j}\nu^{-(\gamma+c)r})\}_{j\in[|S|]}).$$

IndexGen $(PP, OK_{O_\phi}, \{k_\iota\}, \mathcal{O}, \Gamma)$. For the object dataset $\mathcal{O} = \{o_1, \ldots, o_{m_\phi}\}$ of each DO O_ϕ, O_ϕ constructs an index tree, encrypts each object using a symmetric key, and generates the key ciphertexts.

- O_ϕ constructs an R-tree-based index I_ϕ for the dataset $\mathcal{O}$. Specifically, for each leaf node representing a spatio-textual object $o_i = \{\mathsf{p}, \mathsf{W}\}$, O_ϕ first runs Algorithm 1 to encode it into two object vectors $\mathbf{v}_{o_i}^{(1)}, \mathbf{v}_{o_i}^{(2)}$, then calls $\mathsf{kNN.EncD}(\mathbf{v}_{o_i}^{(j)}, \mathsf{OK}_{O_\phi})(\mathsf{j} \in [2])$ to encrypt $\mathbf{v}_{o_i}^{(j)}$ as an initial subindex $[\![\mathbf{v}_{o_i}^{(j)}]\!]$. For each non-leaf node representing a MBR, O_ϕ first runs Algorithms 4, 5 to encode its lower-leaf point (M_1^{ll}, M_2^{ll}) and upper-right point (M_1^{ur}, M_2^{ur}) into $\mathbf{v}^l_{\mathsf{MBR}} = \{\mathbf{v}^{l,1}_{\mathsf{MBR}}, \mathbf{v}^{l,2}_{\mathsf{MBR}}\}, \mathbf{v}^r_{\mathsf{MBR}} = \{\mathbf{v}^{r,1}_{\mathsf{MBR}}, \mathbf{v}^{r,2}_{\mathsf{MBR}}\}$ respectively, then runs kNN.EncD with OK_{O_ϕ} to encrypt them as initial indexes $[\![\mathbf{v}^l_{\mathsf{MBR}}]\!] = \{[\![\mathbf{v}^{l,1}_{\mathsf{MBR}}]\!], [\![\mathbf{v}^{l,2}_{\mathsf{MBR}}]\!]\}, [\![\mathbf{v}^r_{\mathsf{MBR}}]\!] = \{[\![\mathbf{v}^{r,1}_{\mathsf{MBR}}]\!], [\![\mathbf{v}^{r,2}_{\mathsf{MBR}}]\!]\}$.
- O_ϕ encrypts the private data corresponding to each object $o_i (i \in [m_\phi])$ as $\mathbf{c}_i$ using the symmetric key k_i. Note that k_i is the object encryption key.
- To encrypt each symmetric key $k_i (i \in [m_\phi])$, O_ϕ sends the access policy $\Gamma = (\mathcal{M}, \rho) \in (\mathbb{Z}_p^{\ell\times\vartheta}, [\ell] \to \mathbb{Z}_p)$ to the nearest FS F_{O_ϕ} and they cooperate to generate the key ciphertext CT_i^* with two steps.

  `Step 1`: F_{O_ϕ} first randomly generates a vector $\vec{y} = (s, y_2, \ldots, y_\vartheta)^\top \in \mathbb{Z}_p^\vartheta$ and computes the shares $\vec{\lambda} = (\lambda_1, \ldots, \lambda_\ell)^\top = \mathcal{M} \cdot \vec{y}$. Then, F_{O_ϕ} randomly picks ℓ elements $t_1, \ldots, t_\ell \in \mathbb{Z}_p$, computes the intermediate ciphertext CT_ι in Eq. (11), and returns it to O_ϕ.

$$CT_\iota = (C_1 = g^s, \{C_{\iota,1} = g^{\lambda_\iota}\nu^{t_\iota},$$
$$C_{\iota,2} = (u^{\rho(\iota)}\varrho)^{-t_\iota}, C_{\iota,3} = g^{t_\iota}\}_{\iota\in[\ell]}). \tag{11}$$

  `Step 2`: O_ϕ randomly picks $h \in \mathbb{Z}_p$, then obtains key ciphertext CT_i^* as follows:

$$CT_i^* = (\Gamma, C = k_i \cdot e(g,g)^{\alpha h}, C'_0 = g^{\beta h},$$
$$C'_1 = C_1 g^h, \{C_{\iota,1}, C_{\iota,2}, C_{\iota,3}\}_{\iota\in[\ell]}).$$

Convert (SK_{O_ϕ}, I_ϕ). Before storing the index tree I_ϕ, CSP converts the initial indexes included in each node using the switch key SK_{O_ϕ}. Specifically, for each initial index $[\![\mathbf{v}]\!] = (\mathbf{M}^\top_{O_\phi}\mathbf{v}^a, \bar{\mathbf{M}}^\top_{O_\phi}\mathbf{v}^b)$ stored in I_ϕ, CSP converts it into $[\![\mathbf{v}]\!]_{\mathcal{SK}} = SK_{O_\phi} \circ [\![\mathbf{v}]\!] = (\mathbf{M}^\top\mathbf{v}^a, \bar{\mathbf{M}}^\top\mathbf{v}^b)$. In this way, $[\![\mathbf{v}_{o_i}^{(j)}]\!]$ stored in each leaf node is converted into $[\![\mathbf{v}_{o_i}^{(j)}]\!]_{\mathcal{SK}}$. $[\![\mathbf{v}^l_{\mathsf{MBR}}]\!], [\![\mathbf{v}^r_{\mathsf{MBR}}]\!]$ stored in each non-leaf node are converted into $[\![\mathbf{v}^l_{\mathsf{MBR}}]\!]_{\mathcal{SK}} = \{[\![\mathbf{v}^{l,1}_{\mathsf{MBR}}]\!]_{\mathcal{SK}}, [\![\mathbf{v}^{l,2}_{\mathsf{MBR}}]\!]_{\mathcal{SK}}\}, [\![\mathbf{v}^r_{\mathsf{MBR}}]\!]_{\mathcal{SK}} = \{[\![\mathbf{v}^{r,1}_{\mathsf{MBR}}]\!]_{\mathcal{SK}}, [\![\mathbf{v}^{r,2}_{\mathsf{MBR}}]\!]_{\mathcal{SK}}\}$.

TrapGen (UK_{U_θ}, Q). Given the spatial query $Q = \{R, W^*\}$, QU U_θ generates two kinds of token corresponding to the retrieval on the non-leaf node and leaf node, respectively.

- For retrieval on the non-leaf node, U_θ first calls Algorithms 4, 5 to encode the lower-leaf point (R_1^{ll}, R_2^{ll}) and upper-right point (R_1^{ur}, R_2^{ur}) of the rectangle query range R into $\mathbf{v}^l_R = \{\mathbf{v}^{l,1}_R, \mathbf{v}^{l,2}_R\}, \mathbf{v}^r_R = \{\mathbf{v}^{r,1}_R, \mathbf{v}^{r,2}_R\}$ respectively. Then, U_θ runs kNN.EncQ with the user key UK_{U_θ} to encrypt them as non-leaf token

$TK_{nleaf} = \{[\![\mathbf{v}_R^l]\!], [\![\mathbf{v}_R^r]\!]\}$, where $[\![\mathbf{v}_R^l]\!] = \{[\![\mathbf{v}_R^{l,1}]\!], [\![\mathbf{v}_R^{l,2}]\!]\}$, $[\![\mathbf{v}_R^r]\!] = \{[\![\mathbf{v}_R^{r,1}]\!], [\![\mathbf{v}_R^{r,2}]\!]\}$

- For retrieval on the leaf node, U_θ first calls Algorithm 2 to encode Q into $\mathbf{v}_Q^l = \{\mathbf{v}_Q^{l,1}, \mathbf{v}_Q^{l,2}\}, \mathbf{v}_Q^r = \{\mathbf{v}_Q^{r,1}, \mathbf{v}_Q^{r,2}\}$, then runs kNN.EncQ with UK_{U_θ} to encrypt them as $TK_{leaf} = \{[\![\mathbf{v}_Q^l]\!], [\![\mathbf{v}_Q^r]\!]\}$.
- U_θ obtains the trapdoor $TK = \{TK_{nleaf}, TK_{leaf}\}$, and sends it to CSP.

Search $(TK, SK_{U_\theta}, \{I_\phi\}, k)$. Upon receiving the trapdoor TK and the value of k from U_θ, CSP first converts TK with the switch key SK_{U_θ}, then performs RSQ on each DO's index tree I_ϕ.

- For each encrypted vector $[\![\mathbf{v}]\!] = (\mathbf{M}_{U_\theta}\mathbf{v}^a, \bar{\mathbf{M}}_{U_\theta}\mathbf{v}^b)$ in the trapdoor TK, CSP converts it into $[\![\mathbf{v}]\!]_{\mathcal{SK}} = SK_{U_\theta} \circ [\![\mathbf{v}]\!] = (\mathbf{M}^{-1}\mathbf{v}^a, \bar{\mathbf{M}}^{-1}\mathbf{v}^b)$. In this way, TK_{nleaf} is converted into $TK_{nleaf}^* = \{[\![\mathbf{v}_R^l]\!]_{\mathcal{SK}}, [\![\mathbf{v}_R^r]\!]_{\mathcal{SK}}\}$ and TK_{leaf} is converted into $TK_{leaf}^* = \{[\![\mathbf{v}_Q^l]\!]_{\mathcal{SK}}, [\![\mathbf{v}_Q^r]\!]_{\mathcal{SK}}\}$.
- After that, CSP searches each DO's index tree I_ϕ. For each child of the current node, CSP performs $[\![\mathbf{v}_{\mathsf{MBR}}^l]\!]_{\mathcal{SK}}^\top \circ [\![\mathbf{v}_R^r]\!]_{\mathcal{SK}}$ and $[\![\mathbf{v}_{\mathsf{MBR}}^r]\!]_{\mathcal{SK}}^\top \circ [\![\mathbf{v}_R^l]\!]_{\mathcal{SK}}$ to obtain $\{s_1', s_2', s_3', s_4'\}$. If $s_\iota' > 0 (\forall \iota \in [4])$, CSP moves to search its child nodes. When traversing to the leaf node represented by the object o_i, CSP runs Algorithm 3 to compute $[\![\mathbf{v}_{o_i}]\!]_{\mathcal{SK}}^\top \circ [\![\mathbf{v}_Q^l]\!]_{\mathcal{SK}}$ and $[\![\mathbf{v}_{o_i}]\!]_{\mathcal{SK}}^\top \circ [\![\mathbf{v}_Q^r]\!]_{\mathcal{SK}}$, thereby obtains $\{s_{i,1}, s_{i,2}, s_{i,3}, s_{i,4}\}$. If $s_{i,\iota} > 0 (\forall \iota \in [4])$ hold, o_i falls inside R and CSP outputs o_i's relevance score $sco_i = \sum_{\iota=1}^4 s_{i,\iota}$.
- Finally, CSP sorts all relevance scores in descending order, chooses top-k relevance scores and returns corresponding object ciphertexts to F_θ.

Decrypt $(\{CT_i^*\}, sk_{F_\theta,S}, sk_{U_\theta,S})$. To decrypt top-$k$ object ciphertexts, we need to obtain corresponding symmetric keys embedded in $\{CT_i^*\}$. The specific decryption is performed by F_θ and U_θ as follows:

- F_θ first checks whether U_θ's attribute set S matches with the access policy $\Gamma = (\mathcal{M}, \rho)$. If not, output $\perp$; otherwise, F_θ defines a set $\mathcal{I} = \{i : \rho(i) \in S\} (\mathcal{I} \subseteq [\ell])$, and there must exist a set of constants $\{\varpi_i \in \mathbb{Z}_p\}_{i\in\mathcal{I}}$ such that $\sum_{i\in\mathcal{I}} \varpi_i \lambda_i = s$. F_θ computes

$$A = \prod_{i\in\mathcal{I}} (e(L^{K'}L', C_{i,1}) e(K_{j,1}, C_{i,2}) e(K_{j,2}, C_{i,3}))^{\varpi_i} = e(g,g)^{(\gamma+c)rs},$$

  where j is the attribute $\rho(i)$'s index in S.
- After that, F_θ computes D^* via Eq. (12), then sends $\{C, C_0', D^*\}$ to U_θ.

$$D^* = \frac{e(L^{K'}L', C_1')}{A} = e(g,g)^{(\gamma+c)rh}. \tag{12}$$

- Finally, U_θ recovers the symmetric key k_i as follows and decrypts the object ciphertext $\mathbf{c}_i$ to obtain corresponding private data.

$$k_i = \frac{C \cdot D^*}{e(sk_{U_\theta,S}, C_0')} = \frac{k_i \cdot e(g,g)^{\alpha h} \cdot e(g,g)^{(\gamma+c)rh}}{e(g^{(\alpha+(\gamma+c)r)/\beta}, g^{\beta h})}.$$

Trace $(INS_{(t,n)}, PP, MSK, sk_{F_\theta,S}, sk_{U_\theta,S})$. When detecting a suspicious key $(sk_{U_\theta,S}, sk_{F_\theta,S})$, TA makes a key sanity check to verify whether $(sk_{F_\theta,S}, sk_{U_\theta,S})$ is well-formed:

- $sk_{F_\theta,S}$ is in the form of $(K', L, L', \{K_{j,1}, K_{j,2}\}_{j\in[l]})$ and $K' \in \mathbb{Z}_p^*, L, L', K_{j,1}, K_{j,2} \in \mathbb{G}$;
- $e(L', g) = e(L, g^\gamma)$;
- $\exists j \in [l], s.t.\ e(K_{j,2}, g) \cdot e(L^{K'}L', \nu) = e(K_{j,1}, \varrho) \cdot e(K_{j,1}, u)^{A_j}$;
- $e(sk_{U_\theta,S}, g^\beta) = e(g,g)^\alpha \cdot e(L^{K'}L', g)$,

which guarantees that $(sk_{F_\theta,S}, sk_{U_\theta,S})$ can be used in the well-formed decryption process. Otherwise, $(sk_{F_\theta,S},\ sk_{U_\theta,S})$ cannot be used to correctly decrypt ciphertexts, thereby not needing to be traced. For the well-formed $(sk_{F_\theta,S}, sk_{U_\theta,S})$, TA traces the identity embedded in it via *Algorithm 6*. Specifically, TA first extracts (x^*, y^*) by running $Dec_{\hat{k}_2}(c)$ (line 1), then checks whether (x^*, y^*) is on the function f. If so, c is not be forged and TA calls $Dec_{\hat{k}_1}(x^*)$ to determine the identity of malicious QU U_θ (line 2-7). Otherwise, $(sk_{U_\theta,S}, sk_{F_\theta,S})$ does not need to be traced (line 8-9). Note that malicious user tracing is independent of RSQ, thereby not affecting the retrieval efficiency of other QUs. Moreover, the tracing algorithm does not introduce computationally expensive operations. Thus, TA does not introduce significant latency in malicious user tracing.

Algorithm 6. Identity Trace

Input: $INS_{(t,n)}, MSK, sk_{F_\theta,S}$
Output: Identity id_θ or $\perp$
1: Extract $(x^* = \varsigma, y^* = f(\varsigma))$ from $\varsigma || f(\varsigma) = Dec_{\hat{k}_2}(c)$;
2: **if** $(x^*, y^*) \in \{(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\}$ **then**
3: **return** $id_\theta = Dec_{\hat{k}_1}(x^*)$;
4: **else**
5: Recover the secret a_0^* of $INS_{(t,n)}$ by interpolating with $t-1$ points $\{(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\}$ and (x^*, y^*);
6: **if** $a_0^* = f(0)$ **then**
7: **return** $id_\theta = Dec_{\hat{k}_1}(x^*)$;
8: **else**
9: **return** $\perp$.

6 Security Analysis

In this section, we analyze the privacy requirements of PRSQ-F described in Section 4.4. Before proving that PRSQ-F guarantees data security against the Selective Chosen-Plaintext Attacks (IND-SCPA) in Theorem 1, we introduce the q-type assumption [13], l-SDH assumption [31] and corresponding security game.

q-type Assumption. Given the bilinear map parameters $(\mathbb{G}, \mathbb{G}_T, p, e)$ and $g \in \mathbb{G}, a, s, b_1, b_2, \ldots, b_q \in \mathbb{Z}_p$, then it computes D including the following terms:

$$g, g^s,$$
$$g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i/b_j^2}, \forall (i,j) \in [q,q],$$
$$g^{a^i b_j / b_{j'}^2}, \forall (i,j,j') \in [2q,q,q], j \neq j',$$
$$g^{a^i/b_j}, \forall (i,j) \in [2q,q], i \neq q+1,$$
$$g^{sb_j/b_{j'}}, g^{sb_j/b_{j'}^2}, \forall (i,j,j') \in [q,q,q], j \neq j'.$$

If there is no Probabilistic Polynomial-Time (PPT) algorithm $\mathcal{S}$ that can distinguish $e(g,g)^{sa^{q+1}}$ from an element φ randomly selected from $\mathbb{G}_T$, we can say that the PPT algorithm $\mathcal{S}$ dose not have a non-negligible advantage ϵ_1 in solving the q-type problem, namely $\mathrm{P}_1 = |\Pr[\mathcal{S}(D, e(g,g)^{sa^{q+1}}) = 1] - \Pr[\mathcal{S}(D,\varphi) = 1]| < \epsilon_1$. For a random element $h \in \mathbb{Z}_p$, $\mathcal{S}$ knows the knowledge of h (i.e., $g^h, g^\beta, g^{\beta h}$ for $\beta \in \mathbb{Z}_p$), which is less than that of s, thus Eq. (13) holds if the PPT algorithm $\mathcal{S}$ has a negligible advantage ϵ_1 in solving the q-type problem.

$$\left| \begin{array}{l} \Pr[\mathcal{S}(D, g^{\beta h}, e(g,g)^{ha^{q+1}}) = 1] \\ -\Pr[\mathcal{S}(D, g^h, g^\beta, g^{\beta h}, \varphi) = 1] \end{array} \right| < \mathrm{P}_1 < \epsilon_1. \tag{13}$$

Definition 2. We say that the q-type assumption holds if no PPT algorithm has a non-negligible advantage ϵ_1 in solving the q-type problem.

l-SDH Assumption. Given the bilinear map parameters $(\mathbb{G}, \mathbb{G}_T, p, e)$ and the tuple $(g, g^x, g^{x^2}, \ldots, g^{x^l})$ ($x \in \mathbb{Z}_p^*$, g is a generator of $\mathbb{G}$) as inputs, output a pair $(c, g^{\frac{1}{c+x}}) \in \mathbb{Z}_p \times \mathbb{G}$. The t-time algorithm $\mathcal{S}$ does not have a non-negligible advantage ϵ_2 in solving the l-SDH problem if Eq. (14) holds.

$$\Pr\left[\mathcal{S}(g, g^x, g^{x^2}, \ldots, g^{x^l}) = (c, g^{\frac{1}{c+x}})\right] < \epsilon_2. \tag{14}$$

Definition 3. We say that the (l, t, ϵ_2)-SDH assumption holds in $\mathbb{G}$ if no t-time algorithm has a non-negligible advantage ϵ_2 in solving the l-SDH problem.

Definition 4. (IND-SCPA data security of PRSQ-F). Let $\mathcal{S}$ be a PPT simulator, $\mathcal{A}$ be a PPT attacker with challenge access structure Γ^*. The game between $\mathcal{S}$ and $\mathcal{A}$ is shown as follows:

- Initi: $\mathcal{A}$ declares the attacked access structure Γ^*, and sends it to $\mathcal{S}$.
- Setup: $\mathcal{S}$ runs **Setup** to generate the public parameters PP and the master key MSK, then sends PP to $\mathcal{A}$.
- Phase 1: $\mathcal{A}$ adaptively selects an attribute set S from the sets of the attributes $\{S_1, S_2, \ldots, S_{Q_1}\}$ and sends it to $\mathcal{S}$, who runs **KeyGen** to output the private key sk and sends it to $\mathcal{A}$. There is a restriction that $\mathcal{A}$ cannot query the sets that satisfy Γ.
- Challenge: $\mathcal{A}$ picks two equal-length symmetric keys k_0, k_1 and sends them to $\mathcal{S}$, then $\mathcal{S}$ flips a coin $b \in \{0,1\}$, runs **IndexGen** to output the final key ciphertext CT_b^*, and sends CT_b^* to $\mathcal{A}$.
- Phase 2: In the same restriction, $\mathcal{A}$ asks for the private key for another sets of attributes.
- Guess: $\mathcal{A}$ outputs his guess b' for b.

We say that PRSQ-F is IND-SCPA data security if for any PPT attacker in the above security game, it has at most a negligible advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{IND-SCPA-Data}}(1^\eta) = [\Pr[b' = b] - 1/2] < \frac{\epsilon_1 \cdot \epsilon_2}{2}$.

Theorem 1. *PRSQ-F is IND-SCPA data security if the q-type assumption and l-SDH assumption hold.*

Proof. To prove the data security of PRSQ-F, we simulate the security game defined in Definition 4 with a PPT simulator $\mathcal{S}$ and a PPT attacker $\mathcal{A}$. The proof process is based on the schemes [13], [32].

- *Initi:* $\mathcal{S}$ receives the challenging access structure $\Gamma^* = (\mathcal{M}^*, \rho^*) \in (\mathbb{Z}_p^{\ell \times \vartheta}, [\ell] \to \mathbb{Z}_p)$, where $\ell, \vartheta \le q$.
- *Setup:* [13] gives its following public parameters to $\mathcal{S}$:

$$u = g^{\tilde{u}} \cdot \prod_{(i,j) \in [\ell,\vartheta]} (g^{a^j/b_i^2})^{\mathcal{M}_{i,j}^*},$$
$$\varrho = g^{\tilde{\varrho}} \cdot \prod_{(i,j) \in [\ell,\vartheta]} (g^{a^j/b_i^2})^{-\rho^*(i)\mathcal{M}_{i,j}^*},$$
$$\nu = g^{\tilde{\nu}} \cdot \prod_{(i,j) \in [\ell,\vartheta]} (g^{a^j/b_i})^{\mathcal{M}_{i,j}^*},$$
$$e(g,g)^\alpha = e(g^a, g^{a^q}) \cdot e(g,g)^{\tilde{\alpha}}. \tag{15}$$

  Then, $\mathcal{S}$ randomly selects $\beta, \gamma \in \mathbb{Z}_p$ and gives the public parameters of PRSQ-F $PP = \{\Theta, g, u, \varrho, \nu, g^\beta, g^\gamma, e(g,g)^\alpha\}$ to $\mathcal{A}$.
- *Phase 1:* When receiving (id, S) from $\mathcal{A}$, $\mathcal{S}$ first obtains $K', L, L', \{K_{j,1}, K_{j,2}\}_{j \in [|S|]}$, which are the same with that of the selective security proof of [13], then computes

$$K_0 = \left(g^{\tilde{\alpha}} \cdot g^{\tilde{r}} \cdot \prod_{i=2}^{\vartheta} \left(g^{a^{q+2-i}}\right)^{w_i}\right)^{\frac{1}{\beta}}. \tag{16}$$

  Finally, $\mathcal{S}$ returns the private key $sk = (K_0, K', L, L', \{K_{j,1}, K_{j,2}\}_{j \in [|S|]})$ to $\mathcal{A}$.
- *Challenge:* $\mathcal{A}$ randomly picks two symmetric keys with equal length and sends them to $\mathcal{S}$. $\mathcal{S}$ first obtains $C_{\iota,2}, C_{\iota,3} (\iota \in [\ell])$, which are same with that of the selective security proof of [13], then randomly selects $h \in \mathbb{Z}_p^*$ and flips a coin $b \in \{0,1\}$ to compute

$$C = k_b \cdot T' \cdot e(g,g)^{h\tilde{\alpha}}, C_1 = g^s,$$
$$C_0' = (g^\beta)^h, C_1' = C_1 \cdot g^h,$$
$$C_{\iota,1} = g^{\tilde{\lambda}_i} \cdot (g^{sb_\iota})^{-\tilde{\nu}} \cdot \prod_{(i,j) \in [\ell,\vartheta]\, i \neq \iota} (g^{sa^j} b_\iota / b_i)^{-\mathcal{M}_{i,j}^*}.$$

  Finally, $\mathcal{S}$ sends the tuple $CT_b^* = ((\mathcal{M}^*, \rho^*), C, C_0', C_1', \{C_{\iota,1}, C_{\iota,2}, C_{\iota,3}\}_{\iota \in [\ell]})$ to $\mathcal{A}$.
- *Phase 2:* $\mathcal{S}$ repeats the process in *Phase 1*.
- *Guess:* $\mathcal{A}$ outputs his guess b' for b. If $b' = b$, $\mathcal{S}$ outputs 0, i.e., it indicates that the challenge term is $T' = e(g,g)^{ha^{q+1}}$. Otherwise, $\mathcal{S}$ outputs 1 indicating that T' is a random element in group $\mathbb{G}_T$.

If $T' = e(g,g)^{ha^{q+1}}$, then $\mathcal{A}$ can break the security game at least with a non-negligible advantage $\epsilon_1 \cdot \epsilon_2$, and the advantage of $\mathcal{S}$ in solving the q-type assumption and l-SDH assumption is $\frac{1}{2} + \epsilon_1 \cdot \epsilon_2$ and $\frac{1}{2}$ otherwise. Thus, the advantage of $\mathcal{S}$ in breaking the above security game is $\frac{1}{2} \cdot (\frac{1}{2} + \epsilon_1 \cdot \epsilon_2) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon_1 \cdot \epsilon_2}{2}$. Therefore, we can declare that PRSQ-F is IND-SCPA data security on condition that the q-type assumption and l-SDH assumption hold. □

Before proving that PRSQ-F is index confidentiality, query confidentiality and trapdoor unlinkability in Theorem

2, we introduce the definition of non-adaptive semantic security the Theorem 2 is based.

Definition 5. (Non-adaptive semantic security). The probabilistic experiments between $\mathcal{A}$ and $\mathcal{S}$ are shown as follows:

Real$_{\mathcal{A}}(\eta)$: the challenger runs **KeyGen**(1^{η}) to generate an owner key OK_{ϕ} and corresponding switch key SK_{ϕ} for each DO O_{ϕ} and a user key UK_{θ} and corresponding switch key SK_{θ} for each QU U_{θ}. $\mathcal{A}$ outputs the dataset $\mathcal{O}$ and receives an index tree $E(\mathcal{O})$ from the challenger such that $E(\mathcal{O}) \leftarrow$ **IndexGen**$(\mathcal{O}, OK_{\phi})$. Based on $E(\mathcal{O})$, $\mathcal{A}$ makes a polynomial number of spatial queries $\mathcal{Q} = (Q_1, \ldots, Q_n)$ and for each query $Q_j \in \mathcal{Q}$, $\mathcal{A}$ receives a trapdoor TK_j from the challenger such that $TK_j \leftarrow$ **TrapGen**(Q_j, UK_{θ}). Finally, $\mathcal{A}$ outputs $\mathrm{v} = (E(\mathcal{O}), \mathcal{T})$, where $\mathcal{T} = (TK_1, \ldots, TK_n)$.

Sim$_{\mathcal{A},\mathcal{S}}(\eta)$: $\mathcal{A}$ outputs the dataset $\mathcal{O}$. Given the leakage $\mathcal{L}(\mathcal{O}, \mathcal{Q})$, $\mathcal{S}$ generates and sends the index tree $E(\mathcal{O})$ to $\mathcal{A}$. Besides, $\mathcal{S}$ generates an appropriate trapdoor TK_j for each spatial query Q_j. Finally, $\mathcal{A}$ outputs $\mathrm{v} = (E(\mathcal{O}), \mathcal{T})$, where $\mathcal{T} = (TK_1, \ldots, TK_n)$.

We say that PRSQ-F guarantees the index confidentiality, query confidentiality and trapdoor unlinkability in our threat model if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for all PPT distinguishers $\mathcal{D}$,

$$\left| \begin{matrix} \Pr[\mathcal{D}(\mathrm{v}) = 1 : \mathrm{v} \leftarrow \mathbf{Real}_{\mathcal{A}}(\eta)] \\ -\Pr[\mathcal{D}(\mathrm{v}) = 1 : \mathrm{v} \leftarrow \mathbf{Sim}_{\mathcal{A},\mathcal{S}}(\eta)] \end{matrix} \right| \leq \mathrm{neg}(\eta).$$

Theorem 2. *Index confidentiality, query confidentiality and trapdoor unlinkability of PRSQ-F are guaranteed in our threat model.*

We first introduce some auxiliary notions:

- *History* includes an object dataset $\mathcal{O} = \{o_1, \ldots, o_m\}$ and a set of spatial queries $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$, defined as $H = (\mathcal{O}, \mathcal{Q})$, where $Q_j = \{R_{j_1}, w_{j_2}\}$ and R_{j_1}, w_{j_2} are randomly selected from the query range set and query keyword set W^*, respectively.
- *View* is defined as $\mathrm{v}(H) = \{E(\mathcal{O}), \mathcal{T}\}$, where $E(\mathcal{O})$ is an index tree built by $\mathcal{O} = \{o_1, \ldots, o_m\}$, $\mathcal{T} = \{TK_1, \ldots, TK_n\}$ is the trapdoor set.
- *Trace* captures the information learned by CSP, defined as $Tr(H) = \{d, m, U, sp(H), ap(H)\}$. Here, U is the number of cloud clients. $sp(H)$ is the search pattern defined as a $n \times n$ symmetric bit matrix such that for $i, j \in [n]$, $sp(H)[i][j] = 1$ if $TK_i = TK_j$. $ap(H)$ is the access pattern defined as $ap(H) = \{\mathcal{F}(Q_1), \ldots, \mathcal{F}(Q_n)\}$, where $\mathcal{F}(Q_j)$ includes the objects' identities in the range R_{j_1} and contains w_{j_2}.

Proof. We describe a PPT simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A}$, the outputs of **Real**$_{\mathcal{A}}(\eta)$ and **Sim**$_{\mathcal{A},\mathcal{S}}(\eta)$ are computationally indistinguishable. Given the dimension of an object vector d and number of cloud clients U leaked in the *trace* $Tr(H)$, the simulator $\mathcal{S}$ randomly selects two $d \times d$ invertible matrices $\mathbf{A}^*_{\tau,1}, \mathbf{B}^*_{\tau,1}$ for $\tau \in [U]$, and sets $\mathbf{M}^* = \mathbf{A}^*_{1,1} \times \cdots \times \mathbf{A}^*_{U,1}, \bar{\mathbf{M}}^* = \mathbf{B}^*_{1,1} \times \cdots \times \mathbf{B}^*_{U,1}$. After that, it sets the switch key $SK^*_{\tau} = \{(\mathbf{A}^*_{\tau,1})^{-1}\mathbf{M}^*, (\mathbf{B}^*_{\tau,2})^{-1}\bar{\mathbf{M}}^*\}$. Finally, the simulator $\mathcal{S}$ with the *trace* $Tr(H)$ generates a string $\mathrm{v}^*(H) = (E'(\mathcal{O}), \mathcal{T}')$. First, according to the size of the dataset leaked in $\mathcal{L}(\mathcal{O}, Q)$, $\mathcal{S}$ randomly selects m points $\{(x_i, y_i) \in \mathbb{R}^2\}$, then simulates $\mathcal{T}'$ and $E'(\mathcal{O})$ as follows.

Simulate $\mathcal{T}'$: For each query $Q_j (j \in [n])$, the simulator $\mathcal{S}$ generates the trapdoor $TK'_j = \{[\![\mathbf{v}^l_{Q_j}]\!]', [\![\mathbf{v}^r_{Q_j}]\!]', [\![\mathbf{v}^l_{R^*}]\!]', [\![\mathbf{v}^r_{R^*}]\!]'\}$ as follows. Then, $\mathcal{S}$ obtains $\mathcal{T}' = \{TK'_1, \ldots, TK'_n\}$.

- According to the access pattern $ap(H)$, $\mathcal{S}$ constructs two empty sets R_x, R_y. For $i \in [m]$, if (x_i, y_i) is in Q_j's query range, then it puts x_i in R_x and y_i in R_y.
- $\mathcal{S}$ chooses four random numbers $R^{ll}_1, R^{ll}_2, R^{ur}_1, R^{ur}_2$ such that $x^- < R^{ll}_1 < \min R_x, \max R_x < R^{ur}_1 < x^+, y^- < R^{ll}_2 < \min R_y, \max R_y < R^{ur}_2 < y^+$, where $\min R_x, \max R_x$ are the minimal value and maximal value in R_x, respectively; $\min R_y, \max R_y$ are the minimal value and maximal value in R_y, respectively ; x^-, x^+ are the first values in $\{x_1, \ldots, x_m\}$ smaller than $\min R_x$ and larger than $\max R_x$, respectively; y^-, y^+ are the first values in $\{y_1, \ldots, y_m\}$ smaller than $\min R_y$ and larger than $\max R_y$, respectively. It obtains the query range $R^* = \{(R^{ll}_1, R^{ll}_2), (R^{ur}_1, R^{ur}_2)\}$.
- For each $w_{j_2} \in W^*$, $\mathcal{S}$ generates a d_2-dimensional zero vector $\mathbf{v}_{W^*}$, randomly selects an unselected cell $l_{j_2} \in [d_2]$ and sets $\mathbf{v}_{W^*}[i] = 1$. Then, $\mathcal{S}$ encodes $R^*, \mathbf{v}_{W^*}$ with *Algorithm 2*, and randomly picks a pair of elements $(\mathbf{A}^*_{i,1}, \mathbf{B}^*_{i,1})_{i \in [U]}$ from $\{(\mathbf{A}^*_{j,1}, \mathbf{B}^*_{j,1})\}$ as the user key to encrypt the encoded query as $[\![\mathbf{v}^l_Q]\!]', [\![\mathbf{v}^r_Q]\!]'$.
- $\mathcal{S}$ encodes R^* with *Algorithm 4*, *Algorithm 5*, then runs the algorithm kNN.EncQ with $(\mathbf{A}^*_{j,1}, \mathbf{B}^*_{j,1})$ to obtain $[\![\mathbf{v}^l_{R^*}]\!]', [\![\mathbf{v}^r_{R^*}]\!]'$.

Simulate $E'(\mathcal{O})$: According to $\mathcal{O}$, $\mathcal{S}$ generates an encrypted index tree $E'(\mathcal{O})$ as follows:

- For each spatio-textual object $o_i (i \in [m])$, $\mathcal{S}$ generates a d_2-dimensional zero vector $\mathbf{v}_W$. Then, for each $w_{j_2} \in W^*$, $\mathcal{S}$ sets $\mathbf{v}_W[i] = l_{j_2}$ if w_{j_2} in o_i.
- $\mathcal{S}$ encodes $(x_i, y_i), \mathbf{v}_W (i \in [m])$ with Algorithm 1, then randomly picks a pair of unselected elements $(\mathbf{A}^*_{i,1}, \mathbf{B}^*_{i,1})$ from $\{(\mathbf{A}^*_{t,1}, \mathbf{B}^*_{t,1})\}_{t \in [U]}$ as the owner key to encrypt the encoded object as $[\![\mathbf{v}_{o_i}]\!]'$.
- $\mathcal{S}$ first builds an R-tree-based index for m selected points $\{(x_i, y_i)\}$, then encodes each MBR with Algorithms 4, 5, finally runs the algorithm kNN.EncD with $(\mathbf{A}^*_{i,1}, \mathbf{B}^*_{i,1})$ to obtain $[\![\mathbf{v}^l_{\mathsf{MBR}}]\!]'$, $[\![\mathbf{v}^r_{\mathsf{MBR}}]\!]'$.

We now claim that no PPT distinguisher $\mathcal{D}$ can distinguish $\mathrm{v} = (E(\mathcal{O}), \mathcal{T})$ and $\mathrm{v}^* = (E'(\mathcal{O}), \mathcal{T}')$. From the simulation process, we find the indistinguishability of the index tree and trapdoors is based on the indistinguishability of kNN and the introduced randomness. The schemes [10], [28] have shown that the encrypted contents of kNN are indistinguishable in our threat model. In addition, the randomness introduced in encoding and splitting prevents $\mathcal{D}$ from using the leaked specific contents in the spatial queries to distinguish $\mathcal{T}$ and $\mathcal{T}'$, namely randomness ensures the trapdoor unlinkability. Thus, $Adv(\mathcal{D}(E(\mathcal{O}), E'(\mathcal{O}))) \leq \mathrm{neg}_1(\eta)$ and $Adv(\mathcal{D}(\mathcal{T}, \mathcal{T}')) \leq \mathrm{neg}_2(\eta)$. Therefore, we have Eq. (17), indicating that the index confidentiality and query

TABLE 3
Theoretical Computation Cost in Different Schemes

Schemes	Index construction	Key encryption		Trap. generation	Search	Ciphertext decryption	
	DO	FS	DO	QU	CSP	FS	QU
GRSE-tree [6]	$(2+6L_B)(m+m')T_E$	—	—	$8L_BT_E$	$m''(2L_B+2)T_P$	—	—
Fastgeo [7]	$(2+6L_V)mT_E+mT_R$	—	—	$k'(8L_VT_E+T_R)$	$m''(2L_V+2)T_P$	—	—
MRSF [17]	$2(d_2+d_3)^2mT_M$	—	—	$2(d_2+d_3)^2T_M$	$2m(d_2+d_3)T_M$	—	—
CPAD [19]	—	—	$(3\ell+2)mT_E$	—	—	—	$\lvert\mathcal{I}\rvert m(2T_P+T_E)$
PRSQ-F	$(2m+4m')*2d^2T_M$	$(5\ell+1)mT_E$	$3mT_E$	$8*2d^2T_M$	$4(d+m'')*2dT_M$	$\lvert\mathcal{I}\rvert k(3T_P+T_E)$	kT_P

L_V: Length of vector; L_B: Size of Bloom filter; k': Number of encoded query vectors; m': Number of non-leaf nodes; m'': Number of nodes encountered during retrieval; d_3: Number of access roles and random numbers.

confidentiality of PRSQ-F are guaranteed in our threat model.

$$|\Pr[\mathcal{D}(E(\mathcal{O}),\mathcal{T})=1]-\Pr[\mathcal{D}(E'(\mathcal{O}),\mathcal{T}')=1]| \leq \mathrm{neg}_1(\eta)+\mathrm{neg}_2(\eta):=\mathrm{neg}(\eta). \quad (17)$$

7 Performance Analysis

In this section, we analyze the performance of PRSQ-F theoretically and experimentally by comparing it with GRSE-tree [6], FastGeo [7], MRSF [17] and CPAD [19].

7.1 Theoretical Analysis

We present the computation cost of PRSQ-F and comparison schemes in Table 3. We mainly consider several time-consuming operations, i.e., modular exponentiation, bilinear pairing, inner product, and pseudo-random function operations. Let $\mathsf{T_E}, \mathsf{T_P}, \mathsf{T_{DOT}}, \mathsf{T_R}$ be corresponding time complexities. For the inner product of two d-dimensional vectors, we have $\mathsf{T_{DOT}} \approx d * \mathsf{T_M}$, where $\mathsf{T_M}$ is the time complexity of multiplication operation. In **IndexGen**, PRSQ-F encrypts each vector stored in R-tree using the algorithm kNN.EncD, which costs $2d^2\mathsf{T_M}$. There are two object vectors in each leaf node and four MBR vectors in each non-leaf node. Thus, the computation cost of encrypting the index tree is $(2m+4m')*2d^2\mathsf{T_M}$, where m' is the number of non-leaf nodes in the R-tree index. Besides, FS and DO in PRSQ-F cooperate to encrypt each object under the access policy Γ, which cost $(5\ell+1)m\mathsf{T_E}$ and $3m\mathsf{T_E}$, respectively. Although the computation cost of CPAD to encrypt an object is less than that of PRSQ-F, QU in CPAD bears the whole computation cost. In **TrapGen**, PRSQ-F encrypts each query vector using the algorithm kNN.EncQ, which costs $2d^2\mathsf{T_M}$. Thus, the computation cost of generating the trapdoor TK is $8*2d^2\mathsf{T_M}$. In **Search**, PRSQ-F first costs $8*2d^2\mathsf{T_M}$ to convert the trapdoor with corresponding switch key, then costs $4m''*2d\mathsf{T_M}$ to obtain top-k object ciphertexts, where m'' is the number of nodes encountered during retrieval. The computation cost of PRSQ-F is less than that of MRSF and FastGeo, GRSE-tree due to $m'' \ll m$ and $\mathsf{T_M} \ll \mathsf{T_P}$, respectively. In **Decrypt**, FS and QU in PRSQ-F cooperate to decrypt k object ciphertexts, which cost $|\mathcal{I}|k(3\mathsf{T_P}+\mathsf{T_E})$ and $k\mathsf{T_P}$ respectively, where $|\mathcal{I}|$ is the size of $\mathcal{I}$. In CPAD, without search function QU needs to perform m decryption operations.

TABLE 4
Descriptions About Test Dataset

Name	Test dataset
Number of objects	16182
Number of attributes	44
Number of keywords	1804
Spatial location area	$\{[34.9°, 35.5°], [-81.2°, -80.5°]\}$

7.2 Experimental Tests

We conduct experiments on an Ubuntu 16.04 Server with 3.60 GHz 3.19 GHz Intel(R) Core(TM) i5-6500K CPU by using Python and Paring Based Cryptography (PBC) library. In the PBC library, the pairing operation is evaluated on the curve $E(\mathbb{F}_q): y^2=x^3+x$, where $q=512$ and the order of group $\mathbb{G}$ and group $\mathbb{G}_T$ is $p=160$. The test dataset consists of 16,182 North Carolina's business objects from the Yelp dataset,[4] The description and distribution of the test dataset are shown in Table 4 and Fig. 7a, respectively. Each object includes spatial location, attribute, and category description. We extract 1,804 keywords from the category description and 44 attributes from the attribute description. In the following experiments, the degree of index tree in PRSQ-F is set to 20; d_1 in PRSQ-F is set to 12; d_3 in MRSF is set to 20 including 15 access roles and 5 random numbers; L_V in FastGeo is set to 100. It is worth noting that the search precision of PRSQ-F is 100%, since the computation result of Algorithm 3 does not introduce randomness, and the encryption mechanism kNN ensures that the computation result in plaintext domain is equal to that in ciphertext domain, i.e., $[\![\mathbf{p}]\!]^\top \cdot [\![\mathbf{q}]\!] = \mathbf{p}^\top \cdot \mathbf{q}$. Therefore, we focus on testing the efficiency of our scheme.

KeyGen. In Fig. 7b, we plot the key generation time of PRSQ-F in different number of textual keywords d_2, different number of DOs N_o, and different number of QUs N_u. The key generation time is greatly affected by the number of textual keywords. The reason is that the key generation time is mainly used for matrix multiplication, and the matrix dimension is mainly determined by d_2. Besides, we notice that the key generation time of ten QUs and one DO is higher than that of one QU and ten DOs. The reason is that TA needs to generate an additional pair of private keys for each QU. Table 5 shows that the key generation time of PRSQ-F increases with the number of user attributes $|S|$, where $d_2=1800$, $N_u=1$. In the following tests, we assume that the number of DOs and QUs is 1.

IndexGen. Figs. 7c and 7d show the index construction time of all schemes in different number of objects m and different number of textual keywords d_2, respectively. We find that the index construction time of FastGeo is much higher

4. The Yelp dataset contains 160,585 business objects are distributed in 34 states in the United States. https://www.yelp.com/dataset.

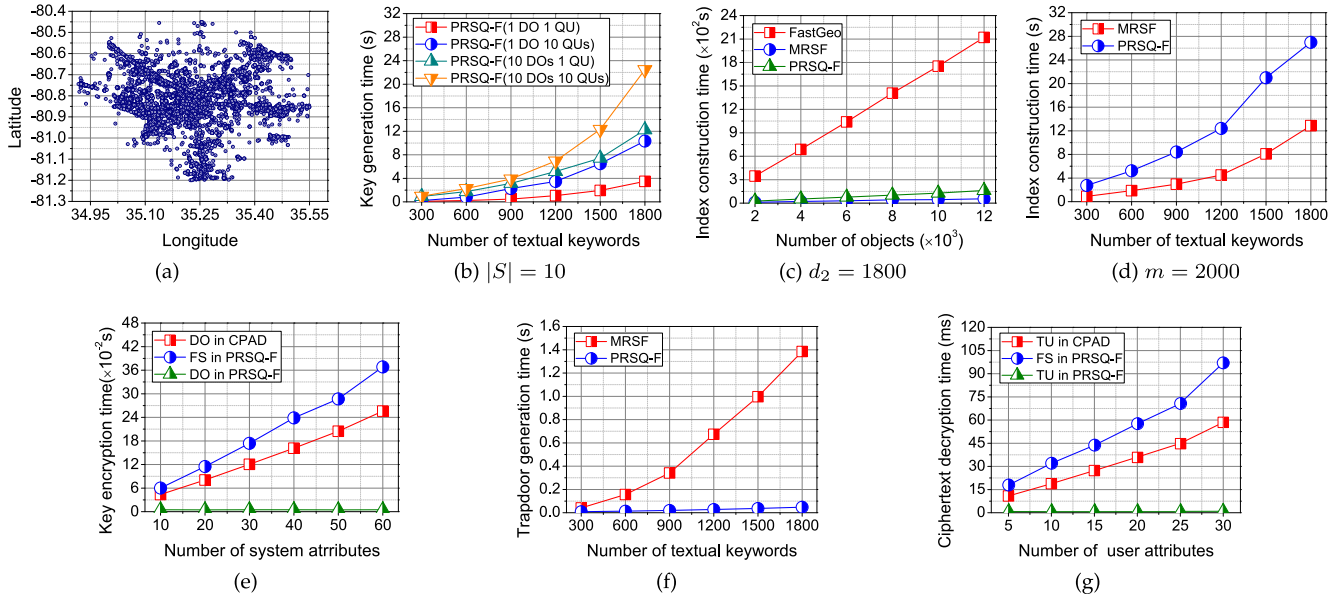

Fig. 7. Practical performance analysis.

than that of PRSQ-F, as FastGeo introduces time-consuming modular exponentiation operation. Compared with MRSF, PRSQ-F takes longer time to construct the index, but it is more efficient in trapdoor generation and search phases.

Fig. 7e shows that the key encryption time of PRSQ-F and CPAD increases with the number of system attributes ℓ. Moreover, in PRSQ-F, DO's key encryption time is hardly affected by ℓ and FS bears most of the key encryption cost. When $\ell = 20$, the key encryption time of DO in CPAD is about 25 times that in PRSQ-F, which indicates that the cloud-assisted fog computing framework reduces the computation burden of DOs. Table 6 shows that the computation cost of encrypting an object key is higher than that of encrypting an object vector, where $d_2 = 1800, \ell = 20$. Fortunately, PRSQ-F shifts most of the key encryption computation from DO to FS without loss of data confidentiality.

TrapGen. In Fig. 7f, we notice that the trapdoor generation time of PRSQ-F, MRSF increases with the number of textual keywords d_2, which is consistent with the theoretical analysis. The difference is that the trapdoor generation time of PRSQ-F is much less than that of MRSF. This is because in MRSF large numbers are added to query vector and large number multiplication is performed to achieve fine-grained access control.

TABLE 5
Key Generation Time of PRSQ-F in Different Number of User Attributes

$\|S\|$	5	10	15	20	25	30
Time (s)	3.51	3.53	3.55	3.58	3.60	3.63

TABLE 6
Comparison of Index Construction Time and Key Encryption Time

Algorithm	Index construction	Key encryption	
Entities	DO	FS	DO
Time (ms)	11.86	119.29	4.83

Search. To demonstrate the effect of index tree, we use PRSQ-L to denote our scheme without constructing index tree and compare its performance with PRSQ-F. We conduct 10 random queries. For each query, we randomly select a point from $\{[34.9°, 35.5°], [-81.2°, -80.5°]\}$ as a center point to generate a square with the length of $0.1°$. Taking this square as the query range R, we obtain the average query results shown in Tables 7 and 8. In Table 7, the search time of all schemes grows with the number of textual keywords d_2, where $m = 2000$. Compared with PRSQ-L, the search time of PRSQ-F is reduced by about half due to the use of index tree. Table 8 shows the search time in different number of objects m, where $d_2 = 1800$. Although the search time of PRSQ-F increases with m, PRSQ-F is extremely efficient. Even the search time of PRSQ-L is about 20 faster than that of Fastgeo. In addition, the search time of MRSF is much

TABLE 7
Search Time in Different Number of Textual Keywords (ms)

d_2	MRSF	PRSQ-L	PRSQ-F
300	249.54	29.22	2.99
600	480.42	31.28	4.43
900	705.36	35.97	9.25
1200	937.72	41.59	15.13
1500	1167.55	46.71	21.89
1800	1398.95	50.79	30.38

TABLE 8
Search Time in Different Number of Objects (ms)

m	MRSF	FastGeo	PRSQ-L	PRSQ-F
2000	1440.15	994.65	53.67	31.21
4000	2888.62	1664.99	96.91	32.54
6000	4265.60	2478.43	138.40	33.73
8000	5589.61	3223.68	181.02	34.13
10000	6895.18	4687.36	226.29	35.22
12000	7998.45	5312.40	276.68	36.38

TABLE 9
Comparison of Trapdoor Generation Time and Ciphertext Decryption Time

Algorithms	Trapdoor generation	Decryption	
Entities	QU	FS	QU
Time (ms)	47.23	32.21	0.89

higher than that of PRSQ-F, as the access control machine in MRSF introduces large number multiplication which is time-consuming. Note that the search time of PRSQ-L, PRSQ-F includes the trapdoor conversion time.

Decrypt. In Fig. 7g, we plot the computation cost of decrypting a key ciphertext in different number of user attributes $|S|$. In PRSQ-F, the ciphertext decryption time of FS is much more than that of QU and it is greatly affected by $|S|$, while the decryption time of QU is about 1 millisecond and is not affected by $|S|$. The ciphertext decryption of QU in CPAD is about 100 times that in PRSQ-F. We conclude that the cloud-assisted fog computing framework reduces the computation burden of resource-limited QUs. Table 9 shows the comparison of trapdoor generation time and ciphertext decryption time, where $d_2 = 1800, |S| = 10$. Through shifting most of encryption computation from QU to FS, the computation cost of QU in each query is nearly halved.

8 Conclusion

In this work, we proposed a privacy-preserving ranked spatial keyword query scheme in mobile cloud-assisted fog computing, namely PRSQ-F. It first designed a novel comparable product encoding strategy to support Ranked Spatial keyword Query (RSQ), then used a conversion protocol to achieve privacy-preserving RSQ in the asymmetric environment. In addition, PRSQ-F employed T-LU-CPABE to achieve decryption of RSQ results without sharing the same secret key and malicious user traceability. Moreover, PRSQ-F constructed an R-tree-based index to achieve efficient retrieval. Both security and performance analysis demonstrated that PRSQ-F could guarantee the security of outsourced spatio-textual data and achieve efficient retrieval.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[2] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial keyword query processing: An experimental evaluation," *Proc. VLDB Endowment*, vol. 6, no. 3, pp. 217–228, 2013.

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

[4] N. Cui, J. Li, X. Yang, B. Wang, M. Reynolds, and Y. Xiang, "When geo-text meets security: Privacy-preserving boolean spatial keyword queries," in *Proc. Int. Conf. Data Eng.*, 2019, pp. 1046–1057.

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

[6] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 704–719, Apr. 2016.

[7] B. Wang, M. Li, and L. Xiong, "FastGeo: Efficient geometric range queries on encrypted spatial data," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 2, pp. 245–258, Mar./Apr. 2019.

[8] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 870–885, Apr. 2019.

[9] Q. Liu, S. Wu, S. Pei, J. Wu, T. Peng, and G. Wang, "Secure and efficient multi-attribute range queries based on comparable inner product encoding," in *Proc. Conf. Commun. Netw. Secur.*, 2018, pp. 1–9.

[10] 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.

[11] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 772–785, Sep./Oct. 2019.

[12] Y. Miao, R. Deng, K.-K. R. Choo, X. Liu, J. Ning, and H. Li, "Optimized verifiable fine-grained keyword search in dynamic multi-owner settings," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1804–1820, Jul./Aug. 2021.

[13] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1274–1288, Jun. 2015.

[14] Y. Luo, S. Fu, D. Wang, M. Xu, and X. Jia, "Efficient and generalized geometric range search on encrypted spatial data in the cloud," in *Proc. IEEE/ACM Int. Symp. Quality of Service*, 2017, pp. 1–10.

[15] C. Chen *et al.*, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.

[16] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017.

[17] J. Li, J. Ma, Y. Miao, Y. Ruikang, X. Liu, and K.-K. R. Choo, "Practical multi-keyword ranked search with access control over encrypted cloud data," *IEEE Trans. Cloud Comput.*, early access, Sep. 15, 2020, doi: 10.1109/TCC.2020.3024226.

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

[19] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, and B. Yang, "Assured data deletion with fine-grained access control for fog-based industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4538–4547, Oct. 2018.

[20] X. Wang, J. Ma, X. Liu, and Y. Miao, "Search in my way: Practical outsourced image retrieval framework supporting unshared key," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2485–2493.

[21] M. Ma, M. Luo, S. Fan, and D. Feng, "An efficient pairing-free certificateless searchable public key encryption for cloud-based IIoT," *Wireless Commun. Mobile Comput.*, vol. 2020, 2020, Art. no. 8850520.

[22] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid index structures for location-based web search," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2005, pp. 155–162.

[23] J. Yang, Y. Zhang, X. Zhou, J. Wang, H. Hu, and C. Xing, "A hierarchical framework for top-k location-aware error-tolerant keyword search," in *Proc. Int. Conf. Data Eng.*, 2019, pp. 986–997.

[24] D. Wu, G. Cong, and C. S. Jensen, "A framework for efficient spatial web object retrieval," *The VLDB J.*, vol. 21, no. 6, pp. 797–822, 2012.

[25] C. Zhang, Y. Zhang, W. Zhang, and X. Lin, "Inverted linear quadtree: Efficient top k spatial keyword search," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1706–1721, Jul. 2016.

[26] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling verifiable boolean range queries over blockchain databases," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 141–158.

[27] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, 2011, pp. 53–70.

[28] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure KNN computation on encrypted databases," in *Proc. ACM Conf. Manage. Data*, 2009, pp. 139–152.

[29] V. Goyal, "Reducing trust in the PKG in identity based cryptosystems," in *Proc. Annu. Int. Cryptol. Conf.*, 2007, pp. 430–447.

[30] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[31] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, pp. 56–73.
[32] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 463–474.

Qiuyun Tong received the BS degree from the Department of Information and Computing Science, Shaanxi Normal University, Xi'an, China, in 2019. She is working toward the PhD degree in the Department of Cyber Engineering, Xidian University, Xi'an, China. Her research interests include information security and applied cryptography.

Yinbin Miao received the BE degree from the Department of Telecommunication Engineering, Jilin University, Changchun, China, in 2011, and the PhD degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He is also a postdoctoral in Nanyang Technological University, Singapore from September 2018 to September 2019. He is currently a lecturer with the Department of Cyber Engineering, Xidian University, Xi'an, China.

Hongwei Li (Member, IEEE) received the PhD degree in computer software and theory from the University of Electronic Science and Technology of China, Chengdu, China, in 2008. He is currently a professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. His research interests include network security, applied cryptography, and trusted computing. He is a member of China Computer Federation.

Ximeng Liu (Member, IEEE) received the BE degree from the Department of Electronic Engineering, Xidian University, Xi'an, China, in 2010, and the PhD degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2015. He is currently a postdoctoral with the Department of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and big data security.

Robert H. Deng (Fellow, IEEE) is currently AXA chair professor of cybersecurity and professor of information systems in the School of Information Systems, Singapore Management University, Singapore, since 2004. His research interests include data security and privacy, multimedia security, network and system security. He has served on the editorial boards of many international journals, including the IEEE Transactions on Information Forensics and Security, the *IEEE Transactions on Dependable and Secure Computing*. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017).