Privacy-Preserving Keyword Similarity Search Over Encrypted Spatial Data in Cloud Computing

Fuyuan Song, Zheng Qin, *Member, IEEE*, Liang Xue, *Graduate Student Member, IEEE*, Jixin Zhang, *Member, IEEE*, Xiaodong Lin, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—With the proliferation of cloud computing, data owners can outsource the spatial data from the Internet of Things devices to a cloud server to enjoy the pay-as-you-go storage resources and location-based services. However, the outsourced services may raise privacy concerns, since the cloud server may not be fully trusted for both data owners and search users. If the data owners and search users conventionally encrypt the spatial data and query requests, the efficiency and functionality of query processing are weakened. Most of the existing works only focus on spatial data search or keyword search and do not consider spatial keyword search over encrypted data. In this article, we first design a geometric range query (GRQ) scheme, which can generate an arbitrary geometric range to fit the search user's desired spatial data while protecting location privacy. Furthermore, based on GRQ, we propose a multidimensional spatial keyword similarity search scheme with access control (MSSAC) by integrating the polynomial function and matrix transformation. Specifically, an access control strategy is defined by a role-based polynomial function, which is embedded in the vectors of indices and trapdoors to achieve efficient and lightweight access control. Moreover, MSSAC enables the cloud server to execute compute-then-compare operations for spatial keyword search in a privacy-preserving manner by leveraging techniques of randomizable permutation and matrix multiplication. The formal security analyses and extensive experiments demonstrate that GRQ and MSSAC preserve the privacy of data owners and search users while achieving efficient spatial keyword search.**

***Index Terms*—Cloud computing, geometric range query (GRQ), Internet of Things (IoT), privacy preserving, spatial keyword search.**

Manuscript received April 20, 2021; revised July 15, 2021; accepted August 27, 2021. Date of publication September 6, 2021; date of current version April 7, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62002112, Grant 61902123, and Grant 61772191; in part by the National Key Research and Development Project under Grant 2018YFB0704000; in part by the Science and Technology Key Projects of Hunan Province under Grant 2019WK2072 and Grant 2018TP3001; in part by the China Scholarship Council; and in part by the Natural Sciences and Engineering Research Council of Canada. *(Corresponding author: Zheng Qin.)*

Fuyuan Song is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China, and also with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: fysong@hnu.edu.cn).

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

Liang Xue and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: liang.xue@uwaterloo.ca; sshen@uwaterloo.ca).

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

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

Digital Object Identifier 10.1109/JIOT.2021.3110300

I. Introduction

SPATIAL keyword search aims to find all related spatiotextual objects inside a geometric query range associated with a set of keywords. Substantially, spatial keyword search has been explored in many Internet of Things (IoT)-based applications [1], such as location-based services (LBSs) [2], vehicle networks [3], [4], valet parking [5], etc. In LBS, a location service provider (LSP) is incentivized to collect spatial data from IoT devices and provide spatial keyword search for users. Spatial keyword search requires abundant storage and computational resources to execute query processing. However, since IoT devices are resource limited, LSP would like to outsource the spatial data to a cloud server and enjoy pay-as-you-go storage and computational resources. Accordingly, a user can submit a query request to the cloud server to obtain the data of interest. After searching the spatial database, the cloud server returns the corresponding query results to the search user.

However, in many cloud-assisted IoT applications, the cloud server is not fully trusted [6]–[11], and some of the outsourced spatial data are sensitive, such as secrecy departments and military bases. There may exist privacy issues if the cloud server is curious about the stored spatial data and query requests. Thus, the outsourced spatial data and the query requests should be protected. Traditionally, the spatial data are encrypted before outsourced to the cloud server, which hinders the spatial data search for users. Moreover, spatial data are generally multidimensional, which usually contain locations and keywords. Most existing works only consider spatial data search or keyword search, which cannot be directly applied to the spatial keyword search. Generally speaking, LSP owns a number of high-dimensional spatial data that are collected from IoT devices, and the spatial data include the locations and the keywords, such as score, taste, price, and reputation in a restaurant. Additionally, to obtain the desired spatial data, users expect to generate arbitrary geometric ranges to cover the locations without disclosing the confidentiality of the spatial data. In general, it is hard to execute efficient spatial keyword search over encrypted data when the dimension of the spatial data is quite large. In addition, since some locations are secret data, e.g., military tunnels and military airports, these special

locations can only be accessed by the specific users who satisfy the access policy. Therefore, a lightweight access control and privacy-preserving arbitrary geometric range query (GRQ) mechanism needs to be designed for spatial keyword search in resource-limited IoT devices.

Currently, many mechanisms have been proposed for searching spatial data over the encrypted database in cloud computing. Zhang *et al.* [2] proposed a dual privacy-preserving spatial data search scheme that can only support continuous trajectory data for an IoT-based system. Wang *et al.* [8], [9], [12] proposed several secure GRQ schemes for 2-D spatial data search, which may limit practical IoT-based applications. Several GRQ schemes [8], [13], [14] may cause expensive computational costs due to the heavy cryptographic operations, and they cannot achieve arbitrary GRQs. To improve the search efficiency, some schemes adopted a bloom filter [15] or *R*-tree [13] to improve the spatial data search performance. Xu *et al.* [16] presented an efficient GRQ scheme that can support arbitrary search types by utilizing the polynomial fitting technique. In terms of access control of spatial data, the schemes in [17] and [18] achieve fine-grained access control over encrypted cloud data. However, their schemes suffer from low efficiency due to the expensive pairing and exponentiation operations, which may limit their usability in IoT-based spatial data search. Therefore, how to design an efficient spatial keyword search with access control that can support arbitrary GRQs in cloud computing is challenging.

In this article, we focus on how to securely and efficiently perform spatial keyword search in cloud computing. We first design a GRQ scheme, which can generate arbitrary geometric ranges that fit search users' desired spatial data. Based on GRQ, we propose a multidimensional spatial keyword similarity search scheme with access control (MSSAC), which can support lightweight access control and efficient privacy-preserving spatial keyword search by employing the role-based polynomial function and matrix multiplication. The proposed schemes protect the confidentiality of the spatial data, i.e., locations and keywords, query requests, and query results by utilizing randomizable matrix multiplication and permutation. The main contributions of this article can be summarized as follows.

1) We design a GRQ scheme that can generate arbitrary geometric ranges that enclose users' desired locations by utilizing the curve fitting technique. Compared with the existing methods, GRQ enables the cloud server to efficiently find locations located in the geometric range for the search user.
2) We propose an efficient and privacy-preserving keyword similarity search scheme over encrypted multidimensional spatial data (MSSAC). Specifically, we design a lightweight access control mechanism in which the access policy is defined by a polynomial function, which can be embedded into the indices and trapdoors. Only the user whose role is a root of the polynomial function can access the spatial data. Moreover, by limiting the desired spatial data into a small geometric range, MSSAC can significantly reduce the computational cost for spatial keyword search.
3) We theoretically prove the security of GRQ and MSSAC, which demonstrates that GRQ and MSSAC are secure under the IND-SCPA model. Experimental results show that the computational costs of GRQ and MSSAC outperform the existing schemes.

The remainder of this article is organized as follows. In Section II, we introduce the related works. In Section III, we provide our system and threat model, problem formulation, and design goals. In Section IV, we describe the preliminaries. We present our proposed scheme in Section V. Security definition and analysis are given in Section VI. We offer the performance analyses and experimental evaluations in Section VII. Finally, we conclude this article in Section VIII.

II. Related Work

In this section, we review some existing works on privacy-preserving GRQ and spatial keyword search.

A. Privacy-Preserving Geometric Range Query

With the rapid development of LBS, more and more works on privacy-preserving GRQ [8], [9], [12], [13] have been studied. For the constrained storage and computational resources of IoT devices, cloud computing is widely recommended as a fundamental technology to provide data storage and search services. The scheme in [8] can simultaneously preserve the privacy of IoT owners and users in public clouds, but the performance of GRQ is inefficient and can only support GRQ on limited polygons, e.g., circular range, triangular range, or rectangular range. Some schemes achieve GRQ by leveraging the technique of the data structure, such as the *R*-tree [14], [15]. However, their schemes cannot be directly applied to the GRQ on arbitrary polygons.

To reduce the computational overhead, searchable symmetric encryption (SSE) has been employed to achieve privacy-preserving geometric query [23], [24]. However, the scheme in [23] mainly focuses on exact data search, which may be somewhat restrictive in IoT systems. Wang *et al.* [8], [9], [12] proposed several GRQ schemes based on Shen–Shi–Waters (SSWs) encryption. Unfortunately, they only support GRQs over 2-D location data and the query performance is inefficient. Therefore, how to achieve efficient and arbitrary GRQs is a challenging issue.

B. Spatial Keyword Search

Spatial keyword search has been widely investigated recently, and spatial data search in public clouds may raise security problems, such as privacy leakage and access control. Encryption-before-outsourcing is an elementary method to protect data privacy in the cloud-assisted IoT system. To support spatial keyword search in the ciphertext environment, a variety of protocols has been proposed [9], [12], [21], [25]. Generally speaking, privacy issues in the outsourced spatial data search can be addressed by utilizing machine learning or homomorphic encryption (HE) [26]–[28]. However, using HE would incur heavy cryptographic operations, making it cannot be used in many practical spatial data search applications when the size of the IoT database becomes large.

TABLE I
COMPARISON WITH PRIOR WORKS

Schemes	Dimension	No trusted authority	Access control	Search method	Security	Performance
FastGeo [12]	2	√	×	Arbitrary	IND-SCPA	High
CRSE [8]	2	√	×	Circle	IND-SCPA	Low
GRSE [9]	2	√	×	Arbitrary	IND-SCPA	Low
EPIRM [19]	Multiple	×	CP-ABE	×	KBA	Very High
EGRQ [16]	2	√	Polynomial	Arbitrary	KBA	Very High
SRQC [13]	2	×	×	Circle	IND-CPA	High
Singer-Server [20]	Multiple	√	×	×	KPA	High
PBRQ-L [21]	Multiple	√	×	Arbitrary	IND-SCPA	High
PBRQ-Q [21]	Multiple	√	×	Arbitrary	IND-SCPA	Very High
Nair et al's Scheme [22]	×	√	Accumulator	×	KBA	Low
Our Design	Multiple	√	Polynomial	Arbitrary	IND-SCPA	Very high

Notes: "×" means that the scheme does not support/consider this functionality, "√" means that the scheme supports this functionality. IND-(S)CPA means Indistinguishability under (Selective) Chosen Plaintext Attack, KPA means Known Plaintext Attack, and KBA means Known Background Attack.

Chen *et al.* [29] proposed a novel search scheme based on the secure multiparty computation (SMC), by which the cloud server can perform spatial keyword search in the ciphertext environment. However, the scheme in [29] requires multiple interactions between the cloud server and the data owner. To achieve privacy-preserving data search, Xue *et al.* [30] presented a two-server architecture to perform spatial data search based on the expensive pairing operations such as the Paillier cryptosystem. To improve the search efficiency, Liang *et al.* [31] proposed a privacy-preserving decision tree classification scheme by using order preserving encryption (OPE), but the order of plaintexts is maintained in ciphertexts, causing the scheme to be subject to the inference attack. The structures, such as the PB-tree [32] and *R*-tree [33] have also been applied to improve the search efficiency. Unfortunately, these schemes cannot support multidimensional spatial data search and lack of privacy protection, since tree structures may reveal the single-dimensional value and data distribution [34].

To achieve efficient spatial data search, some encryption mechanisms have been presented by using matrix transformation [20], [35]–[37]. Zhou and Ren [36] designed a secure biometric authentication protocol based on the matrix transformation and threshold predicate encryption. Zhang *et al.* [37] presented two novel privacy-preserving task allocation schemes by using the properties of the polynomial function and matrix.

Concerning the access control, Zheng *et al.* [38] proposed an exact set similarity search scheme to realize access control by decomposing one secret key into two parts and distributing them to the search user and the cloud server. Subsequently, several data search schemes based on access control [19], [20], [39] have been proposed. However, their schemes rely on the assumption that the data owner and the search user are considered to be fully trusted [20], [39], and there is a trusted authority who distributes the secret keys to the data owner and the search user [19]. Besides, several schemes [17]–[19], [22] can achieve fine-grained access control. However, their schemes are inefficient in multidimensional spatial data search due to the heavy cryptographic operations. To address the above issues, we design a lightweight access control mechanism by exploiting the role-based polynomial function. In this way, the data owner can determine who can access the spatial data, and no trusted authority is required. We compare our design with related works in terms of the functionality, performance, and security achieved, and the results are shown in Table I. It shows that our design supports efficient arbitrary GRQs and lightweight access control while achieving security under the IND-SCPA model.

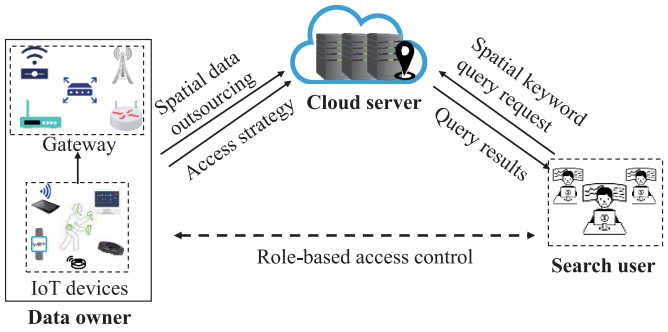


Fig. 1. System model.

III. MODELS AND DESIGN GOALS

In this section, we formalize our system model, threat model, problem formulation, and identify the design goals.

A. System Model

As depicted in Fig. 1, we consider a typical spatial keyword search in a cloud-assisted IoT scenario, which mainly contains three entities in our system model, namely, data owner, cloud server, and search user.

1) *Data Owner ($\mathcal{DO}$):* $\mathcal{DO}$ collects spatial data from IoT devices. To securely store the spatial data in $\mathcal{CS}$, $\mathcal{DO}$ encrypts the spatial data and then outsources them to $\mathcal{CS}$. Also, $\mathcal{DO}$ generates a role-based access strategy for each spatial data, which specifies who can access the spatial data. For resource-constrained IoT devices, $\mathcal{DO}$ needs to reduce the computational cost and the communication overhead of the spatial data encryption. For simplicity, we assume that $\mathcal{DO}$ and the LSP are the same entity.
2) *Cloud Server ($\mathcal{CS}$):* For the LBSs, $\mathcal{DO}$ uploads the encrypted spatial data to $\mathcal{CS}$ to enjoy the pay-as-you-go storage services. $\mathcal{SU}$ can submit an encrypted query

request to obtain the desired spatial data. After receiving the spatial keyword query request, $\mathcal{CS}$ first verifies whether $\mathcal{SU}$ can access the spatial data. If $\mathcal{SU}$ is authenticated, $\mathcal{CS}$ executes spatial keyword similarity search over the encrypted spatial data and returns the query results to the legitimate $\mathcal{SU}$.

3) *Search User ($\mathcal{SU}$):* To obtain the desired spatial data, $\mathcal{SU}$ encrypts a query request with his secret key and sends the search trapdoor to $\mathcal{CS}$. Once receiving query results from $\mathcal{CS}$, the authorized $\mathcal{SU}$ decrypts the corresponding results offline by using the secret key obtained from $\mathcal{DO}$.

B. Threat Model

In our system, we focus on the privacy issues caused by the cloud server and we assume that the communications between the IoT devices and the gateway are secure, which has already been discussed in [40] and [41]. In our threat model, $\mathcal{DO}$ is considered to be fully trusted. For $\mathcal{SU}$, the authorized $\mathcal{SU}$ is considered to be honest, and they would not leak their secret keys to any participating entities for privacy concerns [16], [38]. However, the unauthorized $\mathcal{SU}$ may launch malicious passive attacks. $\mathcal{CS}$ is semihonest, i.e., honest-but-curious, which means $\mathcal{CS}$ follows the designed protocol correctly, but it is curious about $\mathcal{SU}$'s query request and $\mathcal{DO}$'s spatial data. Besides, we assume there is no cloud-user collusion. This assumption is reasonable since the cloud-user collusion is easy to detect and may reduce the reputation of $\mathcal{CS}$ [42]. Based on the available information of the untrusted cloud server, we consider the following two attacks.

1) *Ciphertext Only Attack (COA):* $\mathcal{CS}$ only knows encrypted spatial data, indices, and trapdoors. $\mathcal{CS}$ may analyze the encrypted data to recover the ciphertexts or infer the relationships between the corresponding plaintexts and the ciphertexts.
2) *Chosen Plaintext Attack (CPA):* Except knowing the encrypted spatial data, indices, and trapdoors, $\mathcal{CS}$ can select a number of plaintexts and obtain their corresponding ciphertexts.

C. Problem Formulation

Let $\mathsf{DB} = \{P_1, \ldots, P_m\}$ be a spatial database. As shown in Fig. 4, each multidimensional spatial data P_i ($i \in [1, m]$) in DB is represented as a tuple $P_i = (L_i||W_i)$, where $L_i = (x_i, y_i)$ denotes a 2-D location coordinate and $W_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,n_2})$ denotes a keyword vector in multidimensional spatial data. Also, a query request consists of a geometric range and a query keyword vector.

To achieve efficient and privacy-preserving similarity search over encrypted multidimensional spatial data, we provide a two-stage framework of spatial keyword search. In the first stage, we design a GRQ scheme by utilizing the curve fitting technique, which can generate arbitrary geometric ranges that cover the $\mathcal{SU}$'s desired locations. In the second stage, instead of searching on the whole spatial database, $\mathcal{CS}$ only needs to perform keyword similarity search over the encrypted spatial data located in a small geometric range, which can significantly reduce the search scope in the stage of spatial keyword search. To be specific, based on GRQ, we propose an MSSAC. MSSAC can efficiently process the spatial keyword search in the geometric range and obtain similar spatial data for a given query request.

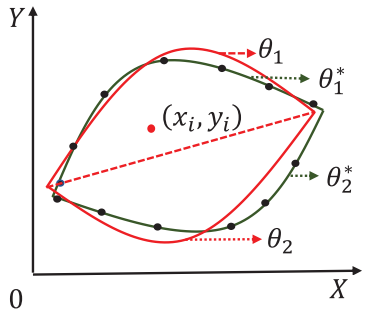


Fig. 2. Curve fitting in a planar space.

D. Design Goals

The proposed GRQ and MSSAC aim to achieve efficient keyword similarity search over multidimensional spatial data while protecting the confidentiality of $\mathcal{DO}$'s spatial data, as well as $\mathcal{SU}$'s query requests and query results. Specifically, our goals are described as follows.

1) *Security:* The security requirements in our schemes are shown as follows.
   a) *Data Confidentiality and Query Confidentiality:* $\mathcal{CS}$ cannot obtain the participating entities' private information, including spatial data, query requests, and query results. Namely, the confidentiality of data and queries should be guaranteed.
   b) *Trapdoor Unlinkability:* $\mathcal{CS}$ cannot distinguish whether two trapdoors are generated from the same query request. Namely, $\mathcal{CS}$ cannot infer the relationship between the trapdoors and their corresponding plaintexts.
2) *Correctness:* The correctness of the proposed schemes should be guaranteed, which means the spatial data in the geometric range should be correctly found, and the returned query results should correctly match the query request.
3) *Efficiency:* The overhead of query processing and data encryption should be minimized to reduce the communication and computational costs of $\mathcal{CS}$ and $\mathcal{DO}$.

IV. Preliminaries

In this section, we introduce the building blocks that are harnessed in our GRQ and MSSAC, including the curve fitting technique [43], and role-based access control [16].

A. Curve Fitting Technique

The curve fitting technique aims to construct a curve or function, which has the best fit to the specified points. Due to its scalability and effectiveness, in our GRQ, we use the curve fitting technique to construct functions that approximately cover the spatial data within a geometric range provided by $\mathcal{SU}$ for the GRQ. Generally speaking, $\mathcal{SU}$ needs to generate two curves to fit the desired locations, which means a geometric range that contains the desired locations is constructed by two curves. As illustrated in Fig. 2, a random geometric

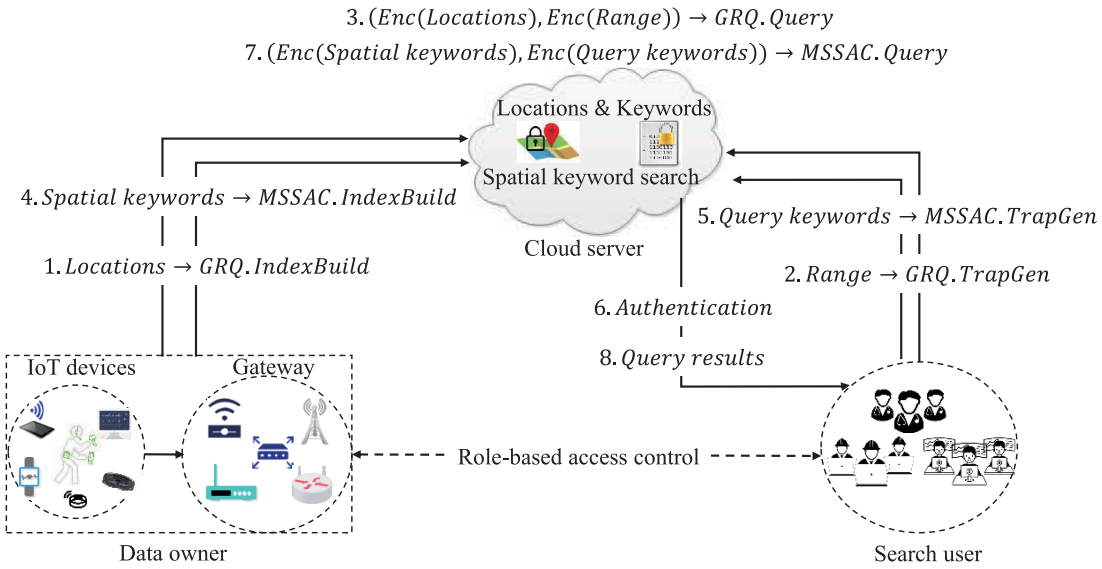


Fig. 3. Overview of the proposed GRQ and MSSAC.

range (i.e., the red region) can be fitted by two curves (i.e., the black curves) with two curvilinear equations being $\theta_1^*(x) = a_0 + a_1x + \cdots + a_{n_1}x^{n_1}$ and $\theta_2^*(x) = b_0 + b_1x + \cdots + b_{n_1}x^{n_1}$, where n_1 is the highest degree of $\theta_1^*(x)$ and $\theta_2^*(x)$, and a_i and b_i are the coefficients of $\theta_1^*(x)$ and $\theta_2^*(x)$, respectively. Therefore, given a geometric range denoted as θ_1^* and θ_2^*, and a location (x_i, y_i), we can determine whether the location (x_i, y_i) located in the geometric range as follows:

$$\begin{cases} \theta_1^*(x_i) - y_i \geq 0 \\ \theta_2^*(x_i) - y_i \leq 0. \end{cases} \tag{1}$$

B. Role-Based Access Control

Access control is a security technique that regulates who can access data resources. One of the most common access control mechanisms is role-based access control. In our scheme, we define our access policy by using a role-based polynomial function, in which coefficients are associated with the roots of the polynomial function. We can use Vieta's formulas [42] $g(x) = c_nx^n + \cdots + c_1x + c_0 = c_n(x - x_n)\cdots(x - x_1)$ to denote the role-based polynomial function. In $g(x)$, the coefficient of x^i is c_i, and $x_1, \ldots, x_n$ are the roots of $g(x)$.

To be specific, a user's role can be denoted as an integer, and if $\mathcal{SU}$'s role is z_i, only the spatial data related to z_i can be accessed and searched by the user. By doing so, each user can only access the owner's spatial data when the user's role satisfies the access policy. Specifically, for a spatial data that can be accessed by $\mathcal{SU}$ whose role belongs to the role set $\delta = \{z_1, z_2, \ldots, z_k\}$, its access policy can be constructed as $g(s) = \prod_{z_i \in \delta}(s - z_i) = \sum_{i=0}^{k} \widehat{z_i}s^i$. Therefore, to achieve the access control of the spatial data, we just need to set the roots of the polynomial function $g(s)$ to be the values of the roles $z_1, z_2, \ldots, z_k$. For instance, suppose there are six roles in the role set δ, i.e., $\delta = \{1, 2, 4, 6, 7, 8\}$, and spatial data can be accessed by the users whose roles belong to the subset $\delta_i = \{1, 2, 4\}$, the access policy of the spatial data can be set as $g_i(s) = \prod_{z_i \in \delta_i}(s - z_i) = s^3 - 7s^2 + 14s - 8$.

TABLE II
NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
sk_1, sk_2	Secret keys for index and query encryption
$\mathsf{DB} = \{P_1, P_2, \cdots, P_m\}$	Spatial database
$P_i = (L_i \| W_i)$	Spatial data in DB
$L_i = (x_i, y_i)$	Location vector of P_i
θ_1^*, θ_2^*	Two curves with the highest degree being n_1
$W_i = (w_{i,1}, \cdots, w_{i,n_2})$	n_2-dimensional spatial keyword vector of P_i
$Q = (q_1, \cdots, q_l)$	Query keyword vector
I_i	Encrypted location index
$\{\Gamma_1, \Gamma_2\}$	Encrypted geometric range trapdoor
C_{W_i}	Encrypted spatial keyword matrix
C_q	Encrypted query keyword matrix
$\delta = (z_1, \cdots, z_k)$	Role set for the spatial data
$g_i(s)$	Role-based polynomial function
$tr(\cdot)$	Matrix trace
UB	Upper bound of the function $tr(C_{W_i}C_q) - g_i(s)$

Therefore, whether an $\mathcal{SU}$ can access spatial data is equivalent to whether the $\mathcal{SU}$'s role is a root of the polynomial function $g(s)$. Our access control strategy can be defined as follows.

Definition 1 (Access Control Strategy): For a spatial data, given the role set $\delta = (z_1, z_2, \ldots, z_k)$, a role-based polynomial function can be constructed as $g_i(s) = N \cdot \prod_{z_i \in \delta}(s - z_i)^2 = \widehat{z_{2k}}s^{2k} + \widehat{z_{2k-1}}s^{2k-1} + \cdots + \widehat{z_0} = \sum_{i=0}^{2k} \widehat{z_i}s^i$, where $\widehat{z_{2k}} = N$, $\widehat{z_{2k-1}} = -N \cdot \sum_{i=1}^{2k} z_i$, $\widehat{z_0} = \prod_{i=1}^{2k} z_i$, and N is a positive integer. Then, an $\mathcal{SU}$ can access the spatial data if and only if $\mathcal{SU}$'s role is a root of the polynomial function $g_i(s)$. Otherwise, $\mathcal{SU}$ cannot access the spatial data when the value of the polynomial function $g_i(s)$ is larger than N.

V. Proposed Scheme

In this section, we first present the detailed construction of GRQ. Then, we introduce our MSSAC. Finally, we analyze the correctness of both GRQ and MSSAC. Before introducing the detailed construction of the proposed schemes, we first show some notations used in our schemes in Table II.

Algorithm 1 GRQ

Input: $L_i = (x_i, y_i)$, $\theta_1^*(x)$, $\theta_2^*(x)$.

Output: $\Lambda = \{0, 1\}$.

1: GRQ.KeyGen$(1^\lambda) \rightarrow sk_1$:
   1) Set the parameters as $\left(n_1, \{a_i\}_{i=0}^{n_1}, \{b_i\}_{i=0}^{n_1}\right)$.
   2) Randomly generate two $(n_1 + 2) \times (n_1 + 2)$ invertible matrices M_1 and M_2 and calculate their inverse matrices M_1^{-1} and M_2^{-1}.
   3) Randomly choose a permutation π_1 and an (n_1+2)-dimensional binary vector S.
   4) Set $sk_1 = \{M_1, M_2, S, \pi_1, M_1^{-1}, M_2^{-1}\}$.

2: GRQ.IndexBuild$(M_1, M_2, S, L_i) \rightarrow I_i$:
   1) Randomly generate a positive number r_i.
   2) Extend L_i to an $(n_1 + 2)$-dimensional vector $\widehat{R}_i = (r_i, r_i x_i, \cdots, r_i x_i^{n_1}, r_i y_i)$.
   3) Permute $\widehat{R}_i$ to obtain $R_i = \pi_1(\widehat{R}_i)$.
   4) For $j = 1$ to $n_1 + 2$, if $S[j] = 1$, $\mathcal{DO}$ randomly splits the value of $R_i[j]$ into $R_i'[j]$ and $R_i''[j]$. If $S[j] = 0$, $\mathcal{DO}$ sets both $R_i'[j]$ and $R_i''[j]$ to be $R_i[j]$.
   5) Calculate $I_i = (M_1^T R_i', M_2^T R_i'')$.

3: GRQ.TrapGen$(M_1^{-1}, M_2^{-1}, S, \theta_1^*, \theta_2^*) \rightarrow \{\Gamma_1, \Gamma_2\}$:
   1) Generate geometric range query as $\widetilde{T}_1 = (a_0, \cdots, a_{n_1}, -1)$ and $\widetilde{T}_2 = (b_0, \cdots, b_{n_1}, -1)$ from two curves $\theta_1^*(x)$ and $\theta_2^*(x)$.
   2) Randomly generate a positive number r_j and embed to $\widetilde{T}_1$ and $\widetilde{T}_2$ to obtain $\widehat{T}_1 = (r_j a_0, r_j a_1, \cdots, r_j a_{n_1}, -r_j)$ and $\widehat{T}_2 = (r_j b_0, r_j b_1, \cdots, r_j b_{n_1}, -r_j)$.
   3) Permute $\widehat{T}_1$ and $\widehat{T}_2$ to obtain $T_1 = \pi_1(\widehat{T}_1)$ and $T_2 = \pi_1(\widehat{T}_2)$.
   4) For $j = 1$ to $n_1 + 2$ and $i = \{1, 2\}$, if $S[j] = 1$, $\mathcal{SU}$ sets both $T_i'[j]$ and $T_i''[j]$ to be $T_i[j]$. If $S[j] = 0$, $\mathcal{SU}$ randomly splits the value of $T_i[j]$ into $T_i'[j]$ and $T_i''[j]$.
   5) Compute $\Gamma_i = (M_1^{-1} T_i', M_2^{-1} T_i'')$, where $i = \{1, 2\}$.

4: GRQ.Query$(I_i, \{\Gamma_1, \Gamma_2\}) \rightarrow \Lambda$:
   1) Compute $I_i \circ \Gamma_1$ and $I_i \circ \Gamma_2$.
   2) Set $\Lambda = 1$ if $I_i \circ \Gamma_1 \geq 0$ and $I_i \circ \Gamma_2 \leq 0$; otherwise set $\Lambda = 0$.

A. Detailed Construction of Geometric Range Query

GRQ consists of four algorithms, namely, GRQ.KeyGen, GRQ.IndexBuild, GRQ.TrapGen, and GRQ.Query. As shown in Fig. 3, we introduce the overview of GRQ processing from step ① to step ③, and we give the detailed implementation of GRQ in Algorithm 1. For clear description, we further describe the construction of GRQ as follows.

1) GRQ.KeyGen(1^λ) $\rightarrow$ sk_1: Given a secure parameter λ, the algorithm GRQ.KeyGen outputs a secret key $sk_1 = (S, M_1, M_2, \pi_1, M_1^{-1}, M_2^{-1})$, where S is an $(n_1 + 2)$-dimensional binary vector used to split the location vectors and query vectors for both $\mathcal{DO}$ and $\mathcal{SU}$, M_1 and M_2 are two $(n_1 + 2) \times (n_1 + 2)$ invertible matrices that are utilized to encrypt the location vectors for $\mathcal{DO}$, π_1 is a random permutation, i.e., $\pi_1 : \mathbb{R}^{n_1+2} \rightarrow \mathbb{R}^{n_1+2}$, and M_1^{-1} and M_2^{-1} are the inverse matrices of M_1 and M_2, which are utilized to encrypt the geometric range for $\mathcal{SU}$.

2) GRQ.IndexBuild(M_1, M_2, S, L_i) $\rightarrow$ I_i: Given two $(n_1 + 2) \times (n_1 + 2)$ invertible matrices M_1 and M_2, the random permutation π_1, the random binary vector S, and a location L_i, the algorithm GRQ.IndexBuild performs extension, permutation, segmentation, and encryption of the location L_i that are collected by the IoT devices and transmitted to the gateway. Each location is denoted as $L_i = (x_i, y_i)$ for $i \in [1, m]$, where m denotes the size of the spatial database DB. First, L_i is extended to an $(n_1 + 2)$-dimensional vector $\widehat{R}_i = (r_i, r_i x_i, r_i x_i^2, \ldots, r_i x_i^{n_1}, r_i y_i)$, where r_i is a one-time random positive value selected by $\mathcal{DO}$. After that, $\widehat{R}_i$ is permuted as $R_i = \pi_1(\widehat{R}_i)$. Then, R_i is randomly split by the $(n_1 + 2)$-dimensional binary vector $S = (S[1], S[2], \ldots, S[n_1 + 2])$, and the splitted vector is denoted as (R_i', R_i''). For each $j = 1, \ldots, n_1 + 2$, the vector split operation is shown as follows:

$$\begin{cases} R_i'[j] = R_i''[j] = R_i[j], & S[j] = 0 \\ R_i'[j] + R_i''[j] = R_i[j], & S[j] = 1. \end{cases}$$

When the split operation is finished, R_i' and R_i'' are encrypted by the two invertible matrices M_1 and M_2. Finally, the index build algorithm GRQ.IndexBuild returns the encrypted IoT spatial location $I_i = (M_1^T R_i', M_2^T R_i'')$, and $\mathcal{DO}$ outsources $\{I_i\}_{i=1}^m$ to $\mathcal{CS}$.

3) GRQ.TrapGen (M_1^{-1}, M_2^{-1}, $S, \theta_1^*, \theta_2^*$) $\rightarrow$ {Γ_1, Γ_2}: Given two invertible matrices M_1^{-1} and M_2^{-1}, the random permutation π_1, and the random binary vector S, GRQ.TrapGen performs permutation, segmentation, and encryption to the geometric range. To be specific, $\mathcal{SU}$ first exploits the curve fitting technique to build $\theta_1^*(x) = a_0 + a_1 x + \cdots + a_{n_1} x^{n_1}$ and $\theta_2^*(x) = b_0 + b_1 x + \cdots + b_{n_1} x^{n_1}$ to fit the desired locations. Then, $\mathcal{SU}$ extracts the coefficients of the curves $\theta_1^*(x)$ and $\theta_2^*(x)$ to generate the query trapdoor as $(\widetilde{T}_1, \widetilde{T}_2)$, where $\widetilde{T}_1 = (a_0, a_1, \ldots, a_{n_1}, -1)$ and $\widetilde{T}_2 = (b_0, b_1, \ldots, b_{n_1}, -1)$. Similarly, $\mathcal{SU}$ selects a one-time random positive value r_j and generates two vectors $\widehat{T}_1$ and $\widehat{T}_2$ for the GRQ, i.e., $\widehat{T}_1 = (r_j a_0, r_j a_1, \ldots, r_j a_{n_1}, -r_j)$ and $\widehat{T}_2 = (r_j b_0, r_j b_1, \ldots, r_j b_{n_1}, -r_j)$. After that, $\widehat{T}_1$ and $\widehat{T}_2$ are permuted as $T_1 = \pi_1(\widehat{T}_1)$ and $T_2 = \pi_1(\widehat{T}_2)$. Then, $\mathcal{SU}$ randomly splits trapdoors T_1 and T_2 by the $(n_1 + 2)$-dimensional binary vector S, and the splitted vectors can be denoted as (T_1', T_1'') and (T_2', T_2''). For each $j = 1, \ldots, n_1 + 2$, the trapdoors can be split as follows:

$$\begin{cases} T_1'[j] + T_1''[j] = T_1[j], \ T_2'[j] + T_2''[j] = T_2[j], \ S[j] = 0 \\ T_1'[j] = T_1''[j] = T_1[j], \ T_2'[j] = T_2''[j] = T_2[j], \ S[j] = 1. \end{cases}$$

After the split operation, the trapdoors are encrypted as $\Gamma_1 = (M_1^{-1} T_1', M_2^{-1} T_1'')$ and $\Gamma_2 = (M_1^{-1} T_2', M_2^{-1} T_2'')$. Finally, $\mathcal{SU}$ submits $\{\Gamma_1, \Gamma_2\}$ to $\mathcal{CS}$.

4) GRQ.Query(I_i, {Γ_1, Γ_2}) $\rightarrow$ Λ = {0,1}: After receiving the encrypted trapdoors, $\mathcal{CS}$ calculates the inner products of I_i and $\{\Gamma_1, \Gamma_2\}$ and returns the corresponding query results. The detailed process is described as follows:

$$\begin{aligned} I_i \circ \Gamma_1 &= \left(M_1^T R_i', M_2^T R_i''\right) \circ \left(M_1^{-1} T_1', M_2^{-1} T_1''\right) \\ &= R_i' \circ T_1' + R_i'' \circ T_1'' \\ &= r_i r_j \left(a_0 + a_1 x_i + \cdots + a_{n_1} x_i^{n_1} - y_i\right) \\ &= r_i r_j \left(\theta_1^*(x_i) - y_i\right). \end{aligned} \tag{2}$$

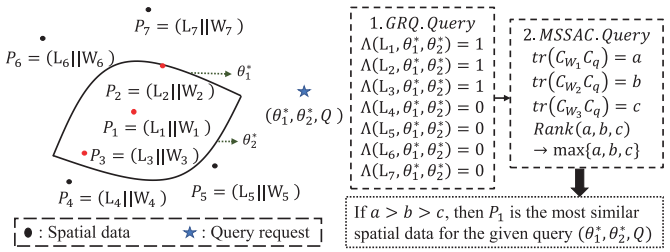


Fig. 4. Example of query types.

Similarly, we have $I_i \circ \Gamma_2 = r_i r_j (\theta_2^*(x_i) - y_i)$, where $I_i \circ \Gamma_2$ denotes the inner product between I_i and Γ_2. As shown in (1), since r_i and r_j are positive values, if both $\theta_1^*(x_i) - y_i \geq 0$ and $\theta_2^*(x_i) - y_i \leq 0$ are satisfied, the query algorithm GRQ.Query outputs 1, which means the location L_i is located in the geometric range (θ_1^*, θ_2^*), as shown in Fig. 4. Otherwise, GRQ.Query outputs 0. By doing so, $\mathcal{CS}$ can efficiently find the correct IoT spatial data for a given geometric range.

Remark: For a given query request, all of the locations in the geometric range are considered to be the desired spatial data. Based on GRQ, we only needs to calculate the Euclidean distances between the keyword vectors of spatial data located in the geometric range and the keyword vector of the query request in the ciphertext. Finally, $\mathcal{CS}$ returns the IoT spatial data with the related keywords in the geometric range to $\mathcal{SU}$. As shown in Fig. 4, we give an example of query types, i.e., GRQ.Query and MSSAC.Query, for the spatial keyword search. We introduce a detailed construction of MSSAC in Section V-B.

B. Multidimensional Spatial Keyword Similarity Search With Access Control

The main idea of MSSAC is to obtain the IoT spatial data located in the geometric range that are similar to the query request. To find those similar spatial data, $\mathcal{CS}$ needs to execute compute-then-compare operations over the keyword vectors of the spatial data in the geometric range and the keyword vector of the query request. Fig. 3 shows the overview of MSSAC, namely, the spatial keyword search from steps ④ to ⑧. Generally speaking, MSSAC consists of four algorithms, i.e., MSSAC.KeyGen, MSSAC.IndexBuild, MSSAC.TrapGen, and MSSAC.Query, which are described as follows.

1) MSSAC.KeyGen$(1^\lambda) \rightarrow sk_2$: Given a secure parameter λ, $\mathcal{DO}$ sets the dimension of the keyword vectors of the spatial data to be n_2 and generates two random $(n_2 + 3 + 2k) \times (n_2 + 3 + 2k)$ invertible matrices D_1 and D_2, which are utilized to encrypt the keyword vectors of the spatial data. Then, $\mathcal{DO}$ randomly generates a permutation π_2, i.e., $\pi_2 : \mathbb{R}^{n_2+3+2k} \rightarrow \mathbb{R}^{n_2+3+2k}$, and computes D_1^{-1} and D_2^{-1}. The secret key can be denoted as $sk_2 = (D_1, D_2, \pi_2, D_1^{-1}, D_2^{-1})$.

2) MSSAC.IndexBuild$(sk_2, W_i) \rightarrow C_{W_i}$: Consider the amount of the spatial data that are located in the geometric range is d, i.e., $\{P_1, P_2, \ldots, P_d\}$, and $d < m$. For any $i \in [1, d]$, the spatial data P_i consist of a location vector and a keyword vector. The n_2-dimensional keyword vector can be denoted as $W_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,n_2})$, where $i \in [1, d]$. Given two invertible matrices D_1 and D_2, and the random permutation π_2, for $i \in [1, d]$, $\mathcal{DO}$ encrypts the keyword vector W_i as follows.

- Step 1: First, $\mathcal{DO}$ extends the n_2-dimensional keyword vector W_i to an $(n_2 + 3 + 2k)$-dimensional vector $\widehat{W_i} = (\alpha w_{i,1}, \alpha w_{i,2}, \ldots, \alpha w_{i,n_2}, \alpha(-[1/2]\sum_{j=1}^{n_2} w_{i,j})^2, 1, \widehat{z_0}, \ldots, \widehat{z_{2k}})$, where α is a one-time random positive number, and $\{\widehat{z_j}\}_{j=0}^{2k}$ represent the coefficients of the role-based polynomial function $g_i(s)$.
- Step 2: Next, $\mathcal{DO}$ permutes the extended vector $\widehat{W_i}$ to an $(n_2 + 3 + 2k)$-dimensional vector $\overline{W_i}$, i.e., $\overline{W_i} = \pi_2(\widehat{W_i})$.
- Step 3: After that, $\mathcal{DO}$ transforms vector $\overline{W_i}$ to a corresponding diagonal square matrix A_{W_i} with the diagonal entries being $\overline{W_i}$.
- Step 4: Finally, $\mathcal{DO}$ generates a random $(n_2+3+2k) \times (n_2+3+2k)$ lower triangular matrix B_{W_i} that is secret to $\mathcal{CS}$. The main diagonal elements of B_{W_i} are set to be 1, and the remainder nonzero elements are one-time random values that are utilized to perturb the diagonal matrix A_{W_i}. Then, $\mathcal{DO}$ encrypts A_{W_i} as $C_{W_i} = D_1 B_{W_i} A_{W_i} D_2$.

When the encryption of the keyword indices is finished, $\mathcal{DO}$ uploads the encrypted indices $\{C_{W_i}\}_{i=1}^d$ to $\mathcal{CS}$.

3) MSSAC.TrapGen$(sk_2, Q) \rightarrow C_q$: Given two invertible matrices D_1^{-1} and D_2^{-1}, the random permutation π_2, and $\mathcal{SU}$'s query request $Q = (q_1, q_2, \ldots, q_l)$, where q_i $(i \in [1, l])$ denotes $\mathcal{SU}$'s query keyword, $\mathcal{SU}$ adds $n_2 - l$ dummy keywords $\{q_{l+1}, \ldots, q_{n_2}\}$ to the query request Q to make the number of keywords in the query request consistent with the number of spatial keywords. Note that each dummy keyword is completely different from the keyword dictionary and, thus, it does not affect the search results. Then, $\mathcal{SU}$ generates the search trapdoor as follows.

- Step 1: First, $\mathcal{SU}$ extends the n_2-dimensional query vector Q to an (n_2+3+2k)-dimensional query vector $\widehat{Q} = (\beta q_1, \beta q_2, \ldots, \beta q_{n_2}, \beta, \gamma, 1, s, s^2, \ldots, s^{2k})$, where β and γ are one-time random positive numbers and s denotes $\mathcal{SU}$'s role.
- Step 2: Next, $\mathcal{SU}$ permutes the extended query vector $\widehat{Q}$ to an $(n_2 + 3 + 2k)$-dimensional vector $\overline{Q}$, i.e., $\overline{Q} = \pi_2(\widehat{Q})$.
- Step 3: After that, $\mathcal{SU}$ transforms $\overline{Q}$ to a diagonal square matrix A_q, where the diagonal entries of A_q are equal to the entries of $\overline{Q}$.
- Step 4: Finally, $\mathcal{SU}$ generates a random $(n_2 + 3 + 2k) \times (n_2 + 3 + 2k)$ lower triangular matrix B_q, whose main diagonal elements are set to be 1 and the rest of nonzero elements are one-time random values. Also, B_q is secret to $\mathcal{CS}$. Then, $\mathcal{SU}$ encrypts the diagonal matrix A_q as $C_q = D_2^{-1} A_q B_q D_1^{-1}$.

When the trapdoor generation phase is over, $\mathcal{SU}$ submits the search trapdoor C_q and a positive integer t $(t < d)$ to $\mathcal{CS}$, where t denotes the number of spatial data that need to be returned from $\mathcal{CS}$.

4) MSSAC.Query$(\{C_{W_i}\}_{i=1}^d, C_q) \rightarrow C_{W_i}$: Upon receiving the search trapdoor C_q, $\mathcal{CS}$ first verifies whether $\mathcal{SU}$ can access

the spatial data by checking whether $\mathcal{SU}$'s role is a root of the role-based polynomial function, i.e., $g_i(s) = \sum_{i=0}^{2k} \widehat{z_i} s^i \stackrel{?}{=} 0$. If not, $\mathcal{CS}$ denies the current query request. Otherwise, $\mathcal{CS}$ authenticates the query user and calculates $C_{W_i}C_q$ and $C_{W_j}C_q$, respectively. As shown in Fig. 4, $\mathcal{CS}$ computes the matrix traces $\mathrm{tr}(C_{W_i}C_q)$ and $\mathrm{tr}(C_{W_j}C_q)$ for the given query keyword trapdoor C_q. Here, C_{W_i}, C_{W_j}, and C_q denote the ciphertexts of spatial keyword vectors W_i, W_j, and the query keyword vector Q, respectively. According to the matrix traces $\mathrm{tr}(C_{W_i}C_q)$ and $\mathrm{tr}(C_{W_j}C_q)$, $\mathcal{CS}$ compares both of them and outputs C_{W_i} if $\mathrm{tr}(C_{W_i}C_q) > \mathrm{tr}(C_{W_j}C_q)$; otherwise, $\mathcal{CS}$ outputs C_{W_j}.

Finally, $\mathcal{CS}$ ranks all the matrix traces $\mathrm{tr}(C_{W_i}C_q)$ for $i \in [1, d]$, and $\mathcal{CS}$ selects the top-t spatial data that have the maximal matrix traces and returns them to $\mathcal{SU}$. The spatial data with the maximal matrix traces are the most similar ones to the query request, which is demonstrated in Theorem 2. The overall process of MSSAC is shown in Algorithm 2.

C. Correctness Analysis

To demonstrate the correctness of GRQ and MSSAC, we have the following theorems (Theorems 1 and 2).

Theorem 1 (Correctness of GRQ): For a given geometric range, GRQ is correct if and only if the GRQ.Query$(I_i, \{\Gamma_1, \Gamma_2\})$ returns the correct locations that belong to the geometric range.

Proof: To proof the correctness of GRQ, we analyze two situations in the GRQ, namely, $\Lambda = 1$ and $\Lambda = 0$. In the first situation, if $\Lambda = 1$, there exists at least one location (x_i, y_i) $(i \in [1, m])$ that belongs to the geometric range, such that $\theta_1^*(x_i) - y_i \geq 0$ and $\theta_2^*(x_i) - y_i \leq 0$ hold. Thus, the query algorithm GRQ.Query$(I_i, \{\Gamma_1, \Gamma_2\})$ outputs 1. In the second situation, if $\Lambda = 0$, which means the locations $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_m, y_m)\}$ satisfy $\theta_1^*(x_i) - y_i < 0$ or $\theta_2^*(x_i) - y_i > 0$ for $i \in [1, m]$. That is, none of the locations is located in the geometric range. Hence, the query algorithm GRQ.Query$(I_i, \{\Gamma_1, \Gamma_2\})$ outputs 0. Therefore, the correctness of GRQ is demonstrated. ∎

To analyze the correctness of MSSAC, we first present the following lemma, which is widely used in linear algebra.

Lemma 1: Given a square matrix A, an invertible matrix D, two diagonal matrices A_F and A_q with the main diagonal being two vectors F and Q, respectively, $\mathrm{tr}(A) = \mathrm{tr}(DAD^{-1})$ and $\mathrm{tr}(A_F A_q) = F \circ Q$ are satisfied.

Based on the above lemma, we have the correctness of MSSAC in Theorem 2.

Theorem 2 (Correctness of MSSAC): Given two keyword vectors W_i and W_j of spatial data, and a keyword vector Q of query request, MSSAC.Query$(C_{W_i}, C_{W_j}, C_q) \rightarrow C_{W_i}$ is correct if and only if $\mathrm{tr}(C_{W_i}C_q) > \mathrm{tr}(C_{W_j}C_q)$.

Proof: Note that $C_{W_i}C_q = D_1 B_{W_i} A_{W_i} A_q B_q D_1^{-1}$. Based on Lemma 1, we have $\mathrm{tr}(C_{W_i}C_q) = \mathrm{tr}(B_{W_i}A_{W_i}A_qB_q)$. In linear algebra, the product of a lower triangular matrix and a diagonal matrix is also a lower triangular matrix. Thus, $B_{W_i}A_{W_i}$ is a lower triangular matrix, whose main diagonal is equal to $\overline{W_i}$. The reason is that $\overline{W_i}$ is the main diagonal of A_{W_i}, and the diagonal entries of B_{W_i} are set to be 1. Similarly, A_qB_q is also a lower triangular matrix, whose main diagonal is $\overline{Q}$.

Algorithm 2 MSSAC

Input: W_i, W_j, Q, $g_i(s)$.

Output: C_{W_i} or C_{W_j}.

1: MSSAC.KeyGen$(1^\lambda) \rightarrow sk_2$:
   1) Set the parameters as (n_2, k).
   2) Randomly generate two $(n_2+3+2k) \times (n_2+3+2k)$ invertible matrices D_1, D_2 and a permutation π_2.
   3) Calculates the inversions D_1^{-1} and D_2^{-1}.
   4) Set $sk_2 = (D_1, D_2, \pi_2, D_1^{-1}, D_2^{-2})$.

2: MSSAC.IndexBuild$(sk_2, W_i) \rightarrow C_{W_i}$:
   1) Generate a random positive number α and extract coefficients $\{\widehat{z_o}, \cdots, \widehat{z_{2k}}\}$ from the function $g_i(s)$.
   2) Extend the spatial keyword vector W_i to an $(n_2 + 3 + 2k)$-dimensional vector $\widehat{W_i} = (\alpha w_{i,1}, \cdots, \alpha w_{i,n_2}, \alpha(-\frac{1}{2}\sum_{j=1}^{n_2} w_{i,j})^2, 1, \widehat{z_0}, \cdots, \widehat{z_{2k}})$.
   3) Permute $\widehat{W_i}$ to obtain $\overline{W_i} = \pi_2(\widehat{W_i})$.
   4) Transform $\overline{W_i}$ to a diagonal matrix A_{W_i} with diagonal of A_{W_i} being $\overline{W_i}$.
   5) Generate a random $(n_2 + 3 + 2k) \times (n_2 + 3 + 2k)$ lower triangular matrix B_{W_i} with diagonal being 1.
   6) Compute $C_{W_i} = D_1 B_{W_i} A_{W_i} D_2$.
   7) Submit the record C_{W_i} to $\mathcal{CS}$.

3: MSSAC.TrapGen$(sk_2, Q) \rightarrow C_q$:
   1) Generate two random numbers β and γ.
   2) Extend vector Q to an $(n_2 + 3 + 2k)$-dimensional vector $\widehat{Q} = (\beta q_1, \cdots, \beta q_{n_2}, \beta, \gamma, 1, s, \cdots, s^{2k})$.
   3) Permute $\widehat{Q}$ to obtain $\overline{Q} = \pi_2(\widehat{Q})$.
   4) Transform $\overline{Q}$ to a diagonal matrix A_q with diagonal being $\overline{Q}$.
   5) Generate a random $(n_2 + 3 + 2k) \times (n_2 + 3 + 2k)$ lower triangular matrix B_q with diagonal fixed as 1.
   6) Compute $C_q = D_2^{-1} A_q B_q D_1^{-1}$.
   7) Send the query C_q to $\mathcal{CS}$.

4: MSSAC.Query$(\{C_{W_i}, C_{W_j}\}, C_q) \rightarrow C_{W_i}$ or C_{W_j}:
   1) Set $\mathcal{SU}$ Authenticated if $g_i(s) = 0$; otherwise Deny the query request of $\mathcal{SU}$.
   2) Calculate $tr(C_{W_i}C_q)$ and $tr(C_{W_j}C_q)$.
   3) Output C_{W_i} if $tr(C_{W_i}C_q) > tr(C_{W_j}C_q)$; otherwise output C_{W_j}.

Since $B_{W_i}A_{W_i}$ and A_qB_q are both lower triangular matrices, the diagonal of $B_{W_i}A_{W_i}A_qB_q$ is equal to the diagonal of $A_{W_i}A_q$, whose trace is equal to $\overline{W_i} \circ \overline{Q}$. Since both $\overline{W_i}$ and $\overline{Q}$ are generated from the same permutation π_2, we have $\overline{W_i} \circ \overline{Q} = \widehat{W_i} \circ \widehat{Q}$. Therefore, we have $\mathrm{tr}(C_{W_i}C_q) = \mathrm{tr}(A_{W_i}A_q) = \overline{W_i} \circ \overline{Q} = \widehat{W_i} \circ \widehat{Q}$.

From the construction of $\widehat{W_i}$ and $\widehat{Q}$, we have $\mathrm{tr}(C_{W_i}C_q) = \alpha\beta \sum_{f=1}^{n} w_{i,f} q_f - (1/2)\alpha\beta \sum_{f=1}^{n} w_{i,f}^2 + \gamma + g_i(s)$. We use UB to denote the upper bound of $\alpha\beta \sum_{f=1}^{n} w_{i,f} q_f - (1/2)\alpha\beta \sum_{f=1}^{n} w_{i,f}^2 + \gamma$ for $1 \leq i \leq d$. According to the access control strategy in Definition 1, $\mathcal{SU}$ can access the IoT spatial data if and only if $g_i(s) = 0$ and $\mathrm{tr}(C_{W_i}C_q) \leq$ UB. Thus, we have $\mathrm{tr}(C_{W_i}C_q) = \alpha\beta \sum_{f=1}^{n} w_{i,f} q_f - (1/2)\alpha\beta \sum_{f=1}^{n} w_{i,f}^2 + \gamma$. Otherwise, $g_i(s)$ is larger than N and $\mathrm{tr}(C_{W_i}C_q) \gg$ UB, which means $\mathcal{SU}$ cannot access the IoT spatial data.

Let $d(\cdot, \cdot)$ represents the Euclidean distance between two keyword vectors. Then, $d(W_i, Q)$ and $d(W_j, Q)$ can be

compared as follows:

$$\begin{aligned} d^2(W_i, Q) - d^2(W_j, Q) &= \sum_{f=1}^{n}\left(w_{i,f} - q_f\right)^2 - \sum_{f=1}^{n}\left(w_{j,f} - q_f\right)^2 \\ &= \frac{2}{\alpha\beta}\sum_{f=1}^{n}\left(\alpha\beta w_{j,f}q_f - \frac{1}{2}\alpha\beta w_{j,f}^2 + \gamma\right) \\ &\quad - \frac{2}{\alpha\beta}\sum_{f=1}^{n}\left(\alpha\beta w_{i,f}q_f - \frac{1}{2}\alpha\beta w_{i,f}^2 + \gamma\right) \\ &= \frac{2}{\alpha\beta}\left(\mathrm{tr}\left(C_{W_j}C_q\right) - \mathrm{tr}\left(C_{W_i}C_q\right)\right). \end{aligned}$$

Due to $\alpha > 0$ and $\beta > 0$, we have

$$\mathrm{tr}\left(C_{W_i}C_q\right) - \mathrm{tr}\left(C_{W_j}C_q\right) > 0 \Leftrightarrow d(W_i, Q) < d\left(W_j, Q\right).$$

Therefore, the correctness of algorithm MSSAC.Query is guaranteed. ∎

VI. Security Definition and Analysis

In this section, we formally present the security definitions and theoretically prove the security of both GRQ and MSSAC.

A. Security Definitions

Before giving the formal security definition, we first define three leakage functions as follows.

1) *Size Pattern:* $\mathcal{CS}$ obtains the total size of spatial database DB and the dimension of spatial data outsourced by $\mathcal{DO}$ and the dimension of the search trapdoor sent by $\mathcal{SU}$.
2) *Access Pattern:* $\mathcal{CS}$ knows which encrypted spatial data record matches a search trapdoor.
3) *Search Pattern:* $\mathcal{CS}$ knows whether a search trapdoor is repeated.

The above leakages can be denoted as $\mathcal{L}$. The leakage function is commonly used in most searchable encryption schemes [16], [34], [44]. We formally give the following definitions to simulate the security game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Definition 2 (Data Confidentiality): Let $\prod$ =(KeyGen, IndexBuild, TrapGen, Query) be a multidimensional spatial keyword similarity search scheme over a security parameter λ. A security game between $\mathcal{A}$ and $\mathcal{C}$ is defined as follows.

Initialization: $\mathcal{A}$ submits two spatial databases $\mathsf{DB}_0 = (P_{0,1}, P_{0,2}, \ldots, P_{0,m})$ and $\mathsf{DB}_1 = (P_{1,1}, P_{1,2}, \ldots, P_{1,m})$ with the same dimension to $\mathcal{C}$, where $P_{i,j}$, $i \in \{0,1\}, j \in [1,m]$ is spatial data that consist of 2-D location vector $L_{i,j}$ and n_2-dimensional keyword vector $W_{i,j}$.

Setup: $\mathcal{C}$ runs KeyGen to generate the secret key, which is kept secret from $\mathcal{A}$.

Phase 1: $\mathcal{A}$ adaptively submits a series of requests to $\mathcal{C}$. Subsequently, $\mathcal{C}$ responses with a ciphertext and a trapdoor through IndexBuild and TrapGen, respectively. Each request is one of the following two types.

1) *Ciphertext Request:* On the jth ciphertext request, $\mathcal{A}$ outputs a spatial data set DB'_j, where $\mathsf{DB}'_j = (P'_{j,1}, P'_{j,2}, \ldots, P'_{j,m})$, and $P'_{j,t} = (L'_{j,t}||W'_{j,t}) = (p'_{j,1}, p'_{j,2}, \ldots, p'_{j,n_2+2})$, where $t \in [1,m]$, $p'_{j,i} \in \mathbb{R}$ for $j \in \{0,1\}$, $i \in [1, n_2+2]$, and $\mathcal{A}$ uploads DB'_j to $\mathcal{C}$. Then, $\mathcal{C}$ responses with a ciphertext $C_{P_{j,t}}$ through IndexBuild.
2) *Trapdoor Request:* On the jth trapdoor request, $\mathcal{A}$ outputs a query request $Q_j = \{\theta^*_{j,1}, \theta^*_{j,2}, W^*_j\}$, where $\theta^*_{j,1}$ and $\theta^*_{j,2}$ are two curvilinear equations, and W^*_j is a query keyword vector. Then, $\mathcal{A}$ uploads Q_j to $\mathcal{C}$. After that, $\mathcal{C}$ responses with a search trapdoor C_{Q_j} through TrapGen, where Q_j is subjected to $\mathcal{L}(C_{P_{0,t}}, Q_j) = \mathcal{L}(C_{P_{1,t}}, Q_j)$.

Challenge: With DB_0 and DB_1 selected in Initialization, $\mathcal{C}$ flips a coin $b \in \{0,1\}$, and calculates $C_{P_{b,i}}$ via IndexBuild. Then, $\mathcal{C}$ returns $\mathsf{EDB}_b \leftarrow \{C_{P_{b,j}}\}_{j=1}^m$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ adaptively selects a series of requests and submits them to $\mathcal{C}$, which are still subjected to the same restrictions in Phase 1.

Guess: $\mathcal{A}$ takes a guess b' of b.

$\prod$ is secure against IND-SCPA on data confidentiality if for any polynomial-time adversary $\mathcal{A}$ in the above game, $\mathcal{A}$ has at most a negligible advantage

$$\mathrm{Adv}^{\mathrm{IND-SCPA-Data}}_{\prod,\mathcal{A}}\left(1^\lambda\right) = \left|\Pr\left(b' = b\right) - \frac{1}{2}\right| \le \mathrm{negl}(\lambda)$$

where $\mathrm{negl}(\lambda)$ represents a negligible function.

Definition 3 (Query Confidentiality): Let $\prod$ =(KeyGen, IndexBuild, TrapGen, Query) be a multidimensional spatial keyword similarity search scheme over a security parameter λ. A security game between $\mathcal{A}$ and $\mathcal{C}$ is defined as follows.

Initialization: $\mathcal{A}$ submits two spatial data queries $Q_0 = \{\theta^*_{0,1}, \theta^*_{0,2}, W^*_0\}$ and $Q_1 = \{\theta^*_{1,1}, \theta^*_{1,2}, W^*_1\}$ with the same dimension to $\mathcal{C}$.

Setup: $\mathcal{C}$ runs KeyGen to generate a secret key, which is kept secret from $\mathcal{A}$.

Phase 1: $\mathcal{A}$ adaptively submits a series of requests to $\mathcal{C}$. Then, $\mathcal{C}$ also responses with a ciphertext and a trapdoor via IndexBuild and TrapGen, respectively. Each request is one of the following two types.

1) *Ciphertext Request:* On the jth ciphertext request, $\mathcal{A}$ outputs a spatial database $\mathsf{DB}'_j = (P'_{j,1}, P'_{j,2}, \ldots, P'_{j,m})$, where $P'_{j,t} = (L'_{j,t}||W'_{j,t}) = (p'_{j,1}, p'_{j,2}, \ldots, p'_{j,n_2+2})$, where $t \in [1,m]$, $p'_{j,i} \in \mathbb{R}$ for $j \in \{0,1\}$, $i \in [1, n_2+2]$, and $\mathcal{A}$ sends it to $\mathcal{C}$. Then, $\mathcal{C}$ responses with a ciphertext $C_{P_{j,t}}$ via IndexBuild, where $C_{P_{j,t}}$ is subject to $\mathcal{L}(C_{P_{j,t}}, Q_0) = \mathcal{L}(C_{P_{j,t}}, Q_1)$.
2) *Trapdoor Request:* On the jth trapdoor request, $\mathcal{A}$ outputs a query request $Q_j = \{\theta^*_{j,1}, \theta^*_{j,2}, W^*_j\}$, where $\theta^*_{j,1}$ and $\theta^*_{j,2}$ are two curvilinear equations, and W^*_j is a query keyword vector. Then, $\mathcal{C}$ responses with a search trapdoor C_{Q_j} through TrapGen.

Challenge: With Q_0 and Q_1 selected in Initialization, $\mathcal{C}$ flips a coin $b \in \{0,1\}$, and calculates $C_{Q_{b,j}}$ via TrapGen. Then, $\mathcal{C}$ returns $C_{Q_{b,j}}$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ adaptively selects a series of requests and submits them to $\mathcal{C}$, which are still subjected to the same restrictions in Phase 1.

Guess: $\mathcal{A}$ takes a guess b' of b.

$\prod$ is secure against IND-SCPA on query confidentiality if for any polynomial-time adversary $\mathcal{A}$ in the above game, $\mathcal{A}$

has at most a negligible advantage

$$\mathrm{Adv}_{\prod,\mathcal{A}}^{\mathrm{IND-SCPA-Query}}(1^{\lambda}) = \left|\Pr(b' = b) - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda).$$

B. Security of Proposed Schemes

Theorem 3 (Security Against IND-SCPA of GRQ): GRQ is secure against the IND-SCPA model.

Proof: According to the above security game defined in Definition 2, we should prove that adversary $\mathcal{A}$ cannot distinguish the ciphertexts $I_{0,j}$ and $I_{1,j}$, even if the adversary $\mathcal{A}$ has oracle access to GRQ.IndexBuild.

Considering that $\mathcal{DO}$'s location vector $L_i = (x_i, y_i)$ is first extended to an $(n_1 + 2)$-dimensional vector $\widehat{R_i} = (r_i, r_i x_i, r_i x_i^2, \ldots, r_i x_i^{n_1}, r_i y_i)$, where r_i is a one-time random positive value. Then, vector $\widehat{R_i}$ is permuted by the permutation π_1. After that, the permuted vector R_i is encrypted into I_i by the random binary vector S and the matrices M_1 and M_2. Since $\mathcal{A}$ has no idea about the random value r_i, the permutation π_1, and the split vector S, he cannot recover the secret keys M_1 and M_2 with the ciphertexts I_i.

Due to the fact that the encrypted indices are constructed as $I_i = (M_1^T R_i', M_2^T R_i'')$, $\mathcal{A}$ cannot obtain M_1 and M_2 from the ciphertext I_i. There exists $2n_1 + 4$ equations in the index construction $I_i = (M_1^T R_i', M_2^T R_i'')$, however, it contains $2n^2 + 8n_1 + 9$ unknowns. Even if $\mathcal{A}$ obtains several plaintext-ciphertext pairs, it is difficult for $\mathcal{A}$ to launch a linear analysis attack.

In Phase 1 and Phase 2 of Definition 2, $\mathcal{A}$ can adaptively select different locations $\{L_{i,j}'\}_{j=1}^{m}$ and observe the corresponding ciphertexts $\{I_{i,j}'\}_{j=1}^{m}$. However, since $\widehat{R_i}$ is a one-time random vector that is determined by $\mathcal{C}$, the ciphertexts are random to $\mathcal{A}$. That is, for a given ciphertext that is encrypted by utilizing the location selected by $\mathcal{A}$, $\mathcal{A}$ cannot distinguish which location is actually encrypted. Hence, even though adversary $\mathcal{A}$ in the above game has oracle access to GRQ.IndexBuild, it has a random guess b' of b with a negligible advantage

$$\mathrm{Adv}_{\mathcal{A},\mathrm{GRQ}}^{\mathrm{IND-SCPA}} = \left|\Pr(b' = b) - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda).$$

Since the trapdoor generation in GRQ is similar to the index construction, query confidentiality can also be protected. Therefore, both the confidentiality of data and query in GRQ can be guaranteed under the IND-SCPA model. ∎

Theorem 4 (Security Against IND-SCPA of MSSAC): MSSAC is secure against the IND-SCPA model.

Proof: To prove the IND-SCPA data confidentiality of MSSAC, we also simulate the security game defined in Definition 2. Note that the n_2-dimensional keyword vector W_i of the spatial data is first extended to an $(n_2 + 3 + 2k)$-dimensional vector $\widehat{W_i}$ with a one-time random number α. Then, $\mathcal{DO}$ permutes the vector $\widehat{W_i}$ to an $(n_2 + 3 + 2k)$-dimensional vector $\overline{W_i}$ with the random permutation π_2. After that, the vector $\overline{W_i}$ is transformed to a diagonal matrix A_{W_i}, and then A_{W_i} is encrypted into matrix C_{W_i} by a random triangular matrix B_{W_i} and secret key D_1, D_2. Finally, the encrypted index is constructed as $C_{W_i} = D_1 B_{W_i} A_{W_i} D_2$. Obviously, since $\mathcal{A}$ cannot observe the one-time random value α, the random permutation π_2, and the random triangular matrix B_{W_i}, it is difficult for $\mathcal{A}$ to obtain D_1 and D_2. Similarly, for the query keyword trapdoor C_q, $\mathcal{A}$ also cannot obtain D_1 and D_2.

Furthermore, the index confidentiality of MSSAC is also guaranteed by the one-time randomness. The one-time randomness is semantic secure, and it is also similar to the one-time pad encryption. Assume that the keyword vector is $W_0 = (w_{0,1}, w_{0,2}, \ldots, w_{0,n_2})$. In Section V-B, the encryption of the keyword vector is processed by the extension, permutation, and encryption. In MSSAC.IndexBuild, the keyword vector W_0 is first extended to a vector $\widehat{W_0} = (\alpha w_{0,1}, \alpha w_{0,2}, \ldots, \alpha w_{0,n_2}, \alpha(-[1/2]\sum_{j=1}^{n_2} w_{0,j}^2), 1, \widehat{z_0}, \widehat{z_1}, \ldots, \widehat{z_{2k}})$, where α is a one-time random value, and $\widehat{z_i}$ for $i \in [0, 2k]$ is the corresponding coefficient of the role-based polynomial function $g_i(s)$. After that, vector $\widehat{W_0}$ is permuted to $\overline{W_0}$, and then $\overline{W_0}$ will be transformed to a diagonal matrix A_{W_0}. Finally, the ciphertext of W_0 is generated as $C_{W_0} = D_1 B_{W_0} A_{W_0} D_2$, where B_{W_0} is a random triangular matrix, whose main diagonal entries are set to 1 and the rest of nonzero entries are one-time random numbers $b_{i,j}$, where $i, j \in [1, n_2 + 2k + 3]$. Given W_i for $i = \{0, 1\}$ chosen by $\mathcal{A}$, since α and $b_{i,j}$ are one-time random numbers determined by $\mathcal{C}$, the ciphertexts of W_i are random to $\mathcal{A}$. Namely, for any two keyword vectors W_0 and W_1, and a ciphertext C_{W_b}, where $b \in \{0, 1\}$, $\mathcal{A}$ cannot distinguish which keyword vector is actually encrypted. Similarly, $\mathcal{A}$ cannot obtain the sensitive information from the ciphertexts of query requests. With a similar analysis in Theorem 3, given a ciphertext $C_{P_{b,j}}$ ($b \in \{0, 1\}$) encrypted by using the plaintext chosen by $\mathcal{A}$, $\mathcal{A}$ cannot distinguish which plaintext is actually encrypted, even though $\mathcal{A}$ has oracle access to MSSAC.IndexBuild. Therefore, we have

$$\mathrm{Adv}_{\mathcal{A},\mathrm{MSSAC}}^{\mathrm{IND-SCPA}}(1^{\lambda}) = \left|\Pr(b' = b) - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda).$$

Similarly, based on the security game defined in Definition 3, $\mathcal{A}$ cannot obtain the sensitive information from the ciphertexts of query keywords under the IND-SCPA model. ∎

Theorem 5: GRQ and MSSAC can achieve trapdoor unlinkability.

Proof: The trapdoor unlinkability of GRQ can be realized by the one-time randomness, random permutation, and random segmentation. Given two GRQs $\widetilde{T_{q1}}$ and $\widetilde{T_{q2}}$ from the same search trapdoor, $\mathcal{SU}$ picks up two one-time random values r_{j1} and r_{j2} and embeds them to $\widetilde{T_{q1}}$ and $\widetilde{T_{q2}}$, respectively. Thus, the trapdoors are random to $\mathcal{A}$. In addition, $\mathcal{A}$ selects two random permutations π_1 and π_1' to obtain T_{q1} and T_{q2}. Hence, the trapdoor unlinkability is guaranteed by the random permutation, and we have $\Pr[T_{q1} = T_{q2}] = [1/(2^{n_1+2})]$, where n_1 is the highest degree of the curvilinear equation. Furthermore, when $S[j] = 0$, both $T_{q1}[j]$ and $T_{q2}[j]$ are randomly split into two random values. Suppose $\Pr[\{T_{q1}'[j], T_{q1}''[j]\} = \{T_{q2}'[j], T_{q2}''[j]\}] = \gamma$ for any dimension j where $S[j] = 0$, we have $\Pr[\{T_{q1}', T_{q1}''\} = \{T_{q2}', T_{q2}''\}] = \gamma^{h_0}$, where h_0 is the number of "0" in the bit vector S. Since $\gamma \to 0$ and $h_0 \to n_1 + 2$, we have $\Pr[T_{q1} = T_{q2}] \to 0$. Therefore, $\mathcal{A}$ cannot infer the relationship between two GRQs. Namely, the trapdoor unlinkability in GRQ is guaranteed.

In MSSAC, since the keyword trapdoor is encrypted as $C_q = D_2^{-1} A_q B_q D_1^{-1}$, it is computationally infeasible for $\mathcal{A}$ to compare whether two keyword trapdoors are generated from the same query request Q, because π_2 is a random permutation generated for perturbing each keyword trapdoor, and B_q is a random lower triangular matrix selected for encrypting each keyword trapdoor. To randomize query results, MSSAC introduces some one-time random values, such as β, γ, and $b_{i,j} \in B_q$, that are embedded to each keyword trapdoor. With different keyword trapdoors generated by the same keyword query, different query results will be calculated even for the same spatial data. Therefore, the trapdoor unlinkability in MSSAC is guaranteed. ■

VII. Performance Evaluation

To evaluate the performance of GRQ and MSSAC, we conduct the experiments on a desktop with 32.0-GB RAM and an Intel Core E3-1225 v5, CPU @ 3.30 GHz. All of the algorithms are implemented in JAVA. We measure the computational costs of GRQ and MSSAC over a real data set and a synthetic data set. The real data set is downloaded from the UCI repository,[1] named the Shuttle data set, which contains 57 000 spatial data with the data dimension being 10. The synthetic data set has 5500 random spatial data, whose vector dimension varies from 10 to 100. Additionally, we extract 100 textual keywords to construct a keyword dictionary from the Enron data set.[2] For the GRQ, the two curvilinear equations with the highest degree being 10 are selected to cover the spatial locations, i.e., $n_1 = 10$. This statement is reasonable since the curves are overfitting when the highest degree is larger than 10, which is demonstrated in [37].

A. Performance Analysis

Before providing the experimental results, we analyze the computational cost and communication overhead of the proposed schemes. The notations used in the performance analysis are described in Table III. For a clear description, we assume that the computational cost of each vector multiplication in GRQ is $\mathsf{T_V}$, which is equal to $O(n_1 + 2)$, and the size of a vector is $|\mathsf{V}|$, i.e., $|\mathsf{V}| = (n_1 + 2)|\lambda|$. In addition, the computational cost of a matrix–matrix multiplication in MSSAC is $\mathsf{T_M}$, which is equal to $O((n_2 + 3 + 2k)^3)$, and the size of a matrix is $|\mathsf{M}|$, i.e., $|\mathsf{M}| = (n_2 + 3 + 2k)^2|\lambda|$.

1) Communication Overhead: We compare the communication overhead of our schemes with the existing works [9], [12], [21] and summarize the results in Table IV. We use $|\mathsf{PRF}|$ to represent the size of PRF output, and $|\mathsf{XOR}|$ to denote the size of the XOR output. In our GRQ, the communication overheads of both GRQ.IndexBuild and GRQ.Query are $m|\mathsf{V}|$ bits, and the communication cost of GRQ.TrapGen is $4|\mathsf{V}|$ bits. In addition, the communication cost of MSSAC.IndexBuild and MSSAC.Query also have the same number of bits, i.e., $d|\mathsf{M}|$, and the communication cost of MSSAC.TrapGen is $2|\mathsf{M}|$.

[1] http://archive.ics.uci.edu/ml

[2] http://www.enron-mail.com

TABLE III
Notations for Performance Analysis

Notations	Descriptions
n_1	The highest degree of the curvilinear equations
n_2	Dimension of the keyword vector
n'	Number of non-leaf nodes in PBRQ
m	Size of the spatial database
b	Size of the bloom filter in GRSE
B	Size of bitmap in PBRQ
G	Size of the grid in FastGeo and PBRQ
d	Number of the spatial data in the geometric range
$\lvert\lambda\rvert$	Bit-length of the security parameter
$\mathsf{T_e}$	Computational cost to compute a exponentiation
$\mathsf{T_p}$	Computational cost to comupte a pairing
$\mathsf{T_{PRF}}$	Computational cost to compute a pseudorandom function
$\mathsf{T_{Enc}}$	Computational cost to perform a symmetric encryption
$\mathsf{T_{Dec}}$	Computational cost to perform a symmetric decryption
$\mathsf{T_{XOR}}$	Computational cost to perform an exclusive-or operation
$\mathsf{T_V}$	Computational cost of each vector multiplication
$\mathsf{T_M}$	Computational cost of a matrix-matrix multiplication

TABLE IV
Comparison of Communication Overhead

Scheme	IndexBuild	TrapGen	Query
GRSE [9]	$m(6b+2)\lvert\lambda\rvert$	$8b\lvert\lambda\rvert$	$m(2b+2)\lvert\lambda\rvert$
FastGeo [12]	$m(6G+2)\lvert\lambda\rvert$	$8dG\lvert\lambda\rvert$	$d(2G+2)\lvert\lambda\rvert$
PBRQ-L [21]	$m\lvert\mathsf{PRF}\rvert$	$\lvert\mathsf{PRF}\rvert$	$m\lvert\mathsf{XOR}\rvert$
PBRQ-Q [21]	$m(\lvert\mathsf{PRF}\rvert+\lvert\mathsf{XOR}\rvert)$	$\lvert\mathsf{PRF}\rvert+\lvert\mathsf{XOR}\rvert$	$m\lvert\mathsf{XOR}\rvert$
GRQ	$m\lvert\mathsf{V}\rvert$	$4\lvert\mathsf{V}\rvert$	$m\lvert\mathsf{V}\rvert$
MSSAC	$d\lvert\mathsf{M}\rvert$	$2\lvert\mathsf{M}\rvert$	$d\lvert\mathsf{M}\rvert$

2) Computational Cost: The computational cost comparison among our schemes and the existing works is shown in Table V. To encrypt the locations and keywords, $\mathcal{DO}$ needs to perform two matrix–vector multiplications in GRQ.IndexBuild and three matrix–matrix multiplications in MSSAC.IndexBuild, which cost $2(n_1 + 2)\mathsf{T_V}$ and $3\mathsf{T_M}$, respectively. Similarly, in the trapdoor generation phase, $\mathcal{SU}$ performs four matrix-vector multiplications in GRQ.TrapGen to generate a search trapdoor and three matrix–matrix multiplications are needed in MSSAC.TrapGen on $\mathcal{SU}$, which is equal to $4(n_1 + 2)\mathsf{T_V}$ and $3\mathsf{T_M}$, respectively. In the phase of Query, $\mathcal{CS}$ first performs GRQ to obtain the locations located in the geometric range. Then, $\mathcal{CS}$ calculates matrix traces between the encrypted keyword trapdoor and encrypted keyword indices, which contain d spatial keyword matrices. Thus, the computational cost of GRQ.Query is $2\mathsf{T_V}$, and the computational cost of MSSAC.Query is $d\mathsf{T_V}$. Note that GRSE [9] and FastGeo [12] are designed based on the pairing operation and SSW-based encryption mechanism. Both of them are public-key-based schemes. Besides, PBRQ [21] relies on the cryptographic operations such as symmetric-key hidden vector encryption (SHVE). The PBRQ scheme may incur a large size of Gray code to achieve arbitrary GRQs, while our schemes are designed by utilizing the techniques of randomizable matrix multiplication and curve fitting without any expensive pairing or exponentiation operations. Therefore, our schemes can significantly reduce the computational cost.

TABLE V
COMPARISON OF COMPUTATIONAL COST

Scheme	IndexBuild	TrapGen	Query
GRSE [9]	$O(m)(6b+2)\mathsf{T_e}$	$O(d)8b\mathsf{T_e}$	$O(m)(2b+2)\mathsf{T_p}$
FastGeo [12]	$O(m)(6G+2)\mathsf{T_e}+\mathsf{T_{PRF}}$	$O(d)(8G\mathsf{T_e}+\mathsf{T_{PRF}})$	$O(d)(2G+2)\mathsf{T_p}$
PBRQ-L [21]	$O(m)(2^{log_2 G}+B)\mathsf{T_{PRF}}$	$O(d)(2^{log_2 G}+B)\mathsf{T_{PRF}}+\mathsf{T_{Enc}}$	$O(m)(2^{log_2 G}+B)\mathsf{T_{XOR}}+\mathsf{T_{Dec}}$
PBRQ-Q [21]	$O(m)(2^{log_2 G}+B)\mathsf{T_{PRF}}+$ $O(n')(2^{log_2 G}(5\mathsf{T_{PRF}}+\mathsf{T_{XOR}})+\mathsf{T_{Enc}}+B\mathsf{T_{PRF}})$	$(O(d)2^{log_2 G}+B)(\mathsf{T_{PRF}}+\mathsf{T_{XOR}})+$ $(O(d)+1)\mathsf{T_{Enc}}+B\mathsf{T_{PRF}}$	$O(dlogm)((2^{log_2 G}+B)\mathsf{T_{XOR}}+$ $\mathsf{T_{Dec}})$
GRQ	$2(n_1+2)\mathsf{T_V}$	$4(n_1+2)\mathsf{T_V}$	$2\mathsf{T_V}$
MSSAC	$3\mathsf{T_M}$	$3\mathsf{T_M}$	$d\mathsf{T_V}$

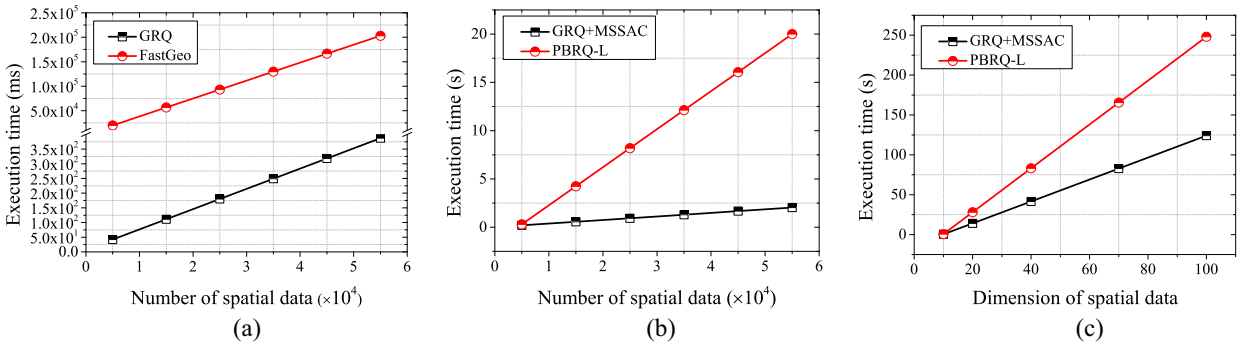


Fig. 5. Computational cost of *IndexBuild*. (a) $n_2 = 10$. (b) $n_2 = 10$. (c) $m = 5500$.

B. Experimental Evaluations

In our experiments, the number of the roles is fixed at 10, i.e., $k = 10$, in the access control strategy. We evaluate the performance of IndexBuild, TrapGen, and Query. In the proposed schemes, GRQ is used to obtain the geometric range for $\mathcal{SU}$, and MSSAC is used to find the matched keywords based on the GRQ. In the following parts, we compare the proposed schemes with the FastGeo [12] and PBRQ [21] in terms of the computational cost, and we set the length of the Gray code to be 100. The FastGeo achieves the GRQ based on pairing operations. In PBRQ, they realize the spatial keyword search with arbitrary geometric ranges by leveraging the techniques of Gray code and Quadtree, and they achieve location privacy protection by using SHVE.

1) IndexBuild: In the index build phase, $\mathcal{DO}$ first builds indices for GRQ. Then, $\mathcal{DO}$ constructs indices for spatial keyword search based on the GRQ.

In the phase of IndexBuild, we evaluate the running time of GRQ.IndexBuild and compare it with FastGeo. In addition, to evaluate the computational cost of keyword encryption, we also test the running time of MSSAC.IndexBuild and compare it with PBRQ-L. As shown in Fig. 5(a), the execution time of GRQ.IndexBuild increases linearly when $n_2 = 10$ and m varies from 5000 to 55 000, GRQ is about 525× faster than that of FastGeo. Besides, we fix n_2 at 10 and change m from 5000 to 55 000, the running time of both GRQ and MSSAC increases slowly. Fig. 5(b) shows that the execution time of (GRQ+MSSAC).IndexBuild is more efficient than that of PBRQ-L. Especially, owing to the generation of the geometric range, the search scope can be greatly reduced and the computational cost of IndexBuild in MSSAC is far less than the PBRQ-L scheme when the size of the data set is larger than 45 000. Also, we compare the running time of (GRQ+MSSAC).IndexBuild with the PBRQ-L scheme over the synthetic data set when the dimension of spatial data varies from 10 to 100. In Fig. 5(c), we can observe that MSSAC is nearly twice as fast as the PBRQ-L scheme. The reason is that our schemes are implemented based on matrix multiplication, while the size of the Gray code in PBRQ-L needs to be set large to achieve arbitrary GRQs. Therefore, for resource-constrained IoT devices, GRQ and MSSAC have the characteristics of high efficiency and practicality in both location encryption and spatial keyword encryption.

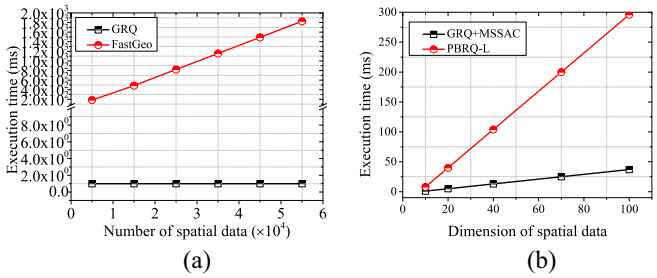


Fig. 6. Computational cost of *TrapGen*. (a) $n_2 = 10$. (b) $m = 5500$.

2) TrapGen: To generate the search trapdoor, $\mathcal{SU}$ first builds two curvilinear equations to cover the desired locations. Then, $\mathcal{SU}$ generates the trapdoor of the query keywords for searching over the encrypted spatial data located in the geometric range.

In the phase of TrapGen, $\mathcal{SU}$ only needs to encrypt a geometric range and a query keyword vector and sends the search trapdoor to $\mathcal{CS}$. Then, $\mathcal{CS}$ utilizes the search trapdoor to perform the spatial keyword search. For $\mathcal{SU}$, the computational cost of the GRQ is extremely low. Fig. 6(a) indicates that the

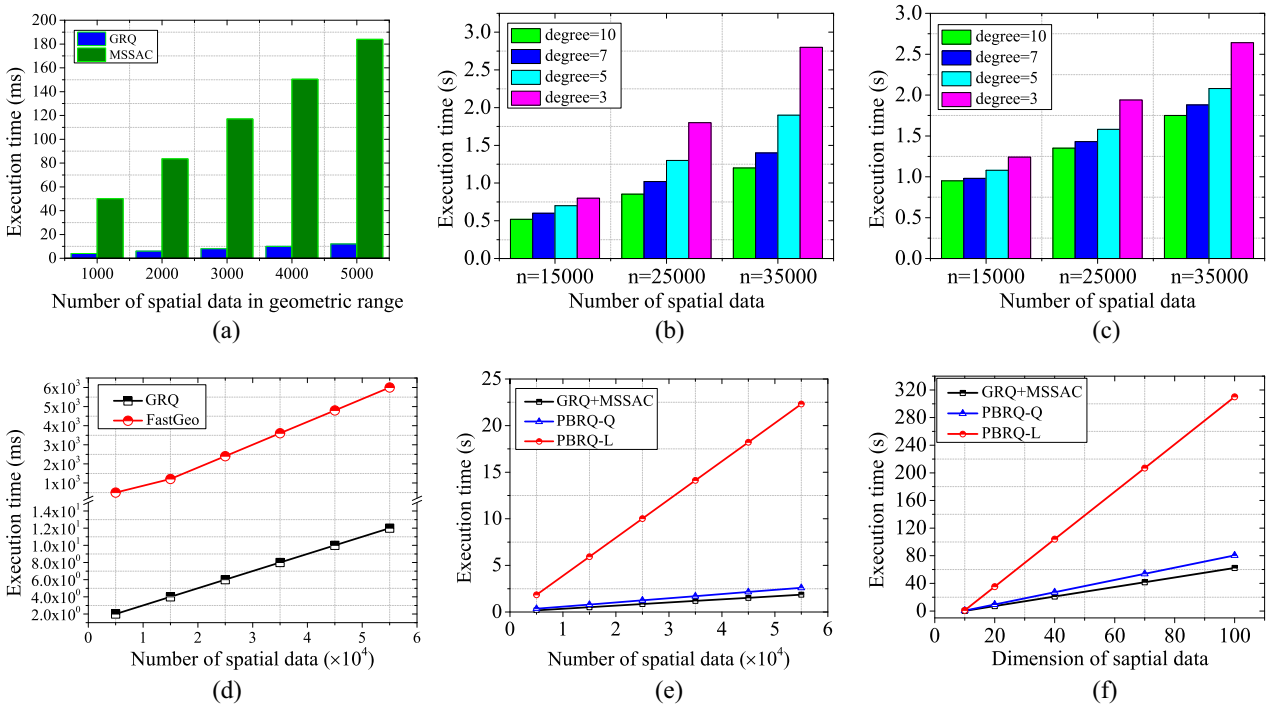


Fig. 7. Computational cost of *Query*. (a) Query processing with d. (b) Query processing for triangular range. (c) Query processing for circular range. (d) $n_2 = 10$. (e) $n_2 = 10$. (f) $m = 5500$.

running time of GRQ.TrapGen remains stable when the number of spatial data changes from 5000 to 55 000, while FastGeo increases linearly with the number of spatial data. In Fig. 6(b), we plot the running time of (GRQ+MSSAC).TrapGen and PBRQ-L.TrapGen, where $m = 5500$ and the dimension of spatial data n_2 ranges from 10 to 100. It shows that the performance of trapdoor generation of MSSAC outperforms that of PBRQ-L, even though the dimension of the query vector up to 100. The reason is that the size of the Gray code extremely affects the performance of TrapGen in PBRQ-L, while our MSSAC enables efficient trapdoor generation thanks to the adoption of matrix multiplication.

3) Query: To achieve efficient and privacy-preserving spatial keyword search, GRQ is used to reduce the search scope and enable the size of spatial data to be stable. The privacy of locations and keywords can be protected by utilizing random permutation and matrix multiplication.

In the phase of Query, as shown in Algorithm 1, $\mathcal{CS}$ first finds the locations located in the geometric range generated from the curvilinear equations $\theta_1^*(x)$ and $\theta_2^*(x)$ based on the curve fitting technique. Then, according to the corresponding keyword vectors of the spatial data in the geometric range, $\mathcal{CS}$ executes a spatial keyword search for the given query keywords as shown in Algorithm 2. To illustrate the query performance of GRQ and MSSAC, we evaluate the time cost of query processing with different spatial data in the geometric range, i.e., d, and we test the execution time of the query processing for different ranges, i.e., triangular range and circular range, in different degrees of curvilinear equations. As shown in Fig. 7(a), when the number d of spatial data in the geometric range ranges from 1000 to 5000, the query processing time of MSSAC increases linearly and the query time of GRQ increases slowly. As shown in Fig. 7(b) and (c), when changing the geometric ranges with different degrees, we can see that the time cost increases as the degree becomes smaller. This is reasonable because the smaller degree results in that more spatial data can satisfy the curve fitting condition. In addition, we compare the running time of GRQ with FastGeo, and MSSAC with PBRQ, respectively. In Fig. 7(d), we plot the query processing time of GRQ and the FastGeo scheme varying with m. In Fig. 7(d), we can see that the geometric query of FastGeo needs more time than that of GRQ. The query performance of GRQ is nearly 500× faster than that of FastGeo, which is caused by the expensive pairing operations. Fig. 7(e) illustrates that the query time of MSSAC increases slowly as the size of the spatial data set grows, and we can observe that MSSAC is more efficient than the PBRQ-L scheme. For MSSAC, it only takes 1.9 s to search the whole data set when $n_2 = 10$. Fig. 7(f) shows that the running time of query processing in PBRQ-L is much more than that of MSSAC when $m = 5500$. MSSAC only takes 62.3 s to return the correct results to $\mathcal{SU}$, which is 5× faster than that of PBRQ-L and slightly superior to PBRQ-Q, whose query complexity is faster-than-linear. The reason is that MSSAC is implemented by leveraging the techniques of matrix multiplication and GRQ, which not only avoids the cryptographic operations but also reduces the search scope in a limited geometric range. From Table VI, we can observe that our schemes are quite efficient. Specifically, the query time of our design is far less than PBRQ-L and close to PBRQ-Q, and our design is significantly efficient than the other two public-key-based GRQ schemes.

TABLE VI
QUERY TIME COMPARISON AMONG SCHEMES

m	1×10^4	2×10^4	3×10^4	4×10^4	5×10^4
GRSE [9]	4.55 h	6.36 h	8.19 h	10.02 h	11.85 h
FastGeo [12]	47.86 s	81.92 s	115.98 s	150.04 s	184.13 s
PBRQ-L [21]	5.13 s	10.57 s	16.01 s	21.45 s	26.89 s
PBRQ-Q [21]	0.26 s	0.37 s	0.95 s	1.32 s	1.75 s
GRQ	0.004 s	0.006 s	0.008 s	0.01 s	0.012 s
GRQ+MSSAC	0.13 s	0.28 s	0.69 s	1.02 s	1.36 s

Notes: h denotes hour and s denotes second.

VIII. CONCLUSION

In this article, we have designed a GRQ scheme, by which a search user can construct arbitrary geometric ranges that cover the user's desired locations. GRQ enables the cloud server to efficiently obtain the similar spatial data in the geometric range for a given query request. Based on GRQ, we have proposed an efficient and privacy-preserving spatial keyword similarity search scheme, called MSSAC, which can support lightweight access control and efficient keyword similarity search over encrypted multidimensional spatial data. Moreover, the data owner's spatial data and the search user's query request can be well protected. Also, GRQ and MSSAC are highly practical for spatial keyword search in resource-constrained IoT devices since they do not involve heavy cryptographic operations. For future work, we will design a privacy-preserving Boolean spatial keyword search scheme that enables more complex functions in cloud-based IoT systems.

REFERENCES

[1] W. Zhang, D. Yang, Y. Xu, X. Huang, J. Zhang, and M. Gidlund, "DeepHealth: A self-attention based method for instant intelligent predictive maintenance in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5461–5473, Aug. 2021.

[2] S. Zhang, G. Wang, M. Z. A. Bhuiyan, and Q. Liu, "A dual privacy preserving scheme in continuous location-based services," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4191–4200, Oct. 2018.

[3] C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[4] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[5] J. Ni, X. Lin, and X. Shen, "Toward privacy-preserving valet parking in autonomous driving era," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2893–2905, Mar. 2019.

[6] J. Liang, Z. Qin, L. Xue, X. Lin, and X. Shen, "Efficient and privacy-preserving decision tree classification for health monitoring systems," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 12528–12539, Apr. 2021, doi: 10.1109/JIOT.2021.3066307.

[7] D. Liu, J. Ni, C. Huang, X. Lin, and X. S. Shen, "Secure and efficient distributed network provenance for IoT: A blockchain-based approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7564–7574, Aug. 2020.

[8] B. Wang, M. Li, H. Wang, and H. Li, "Circular range search on encrypted spatial data," in *Proc. IEEE CNS*, Florence, Italy, 2015, pp. 182–190.

[9] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 704–719, 2016.

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

[11] H. Deng, Z. Qin, L. Sha, and H. Yin, "A flexible privacy-preserving data sharing scheme in cloud-assisted IoT," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11601–11611, Dec. 2020.

[12] 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. 2017.

[13] H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li, "An efficient privacy-preserving location-based services query scheme in outsourced cloud," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7729–7739, Sep. 2016.

[14] R. Guo, B. Qin, Y. Wu, R. Liu, H. Chen, and C. Li, "MixGeo: Efficient secure range queries on encrypted dense spatial data in the cloud," in *Proc. IEEE IWQoS*, Phoenix, AZ, USA, 2019, pp. 1–10.

[15] 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.

[16] 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 Security*, vol. 14, pp. 870–885, 2019.

[17] J. Yuan, S. Yu, and L. Guo, "SEISA: Secure and efficient encrypted image search with access control," in *Proc. IEEE INFOCOM*, Hong Kong, 2015, pp. 2083–2091.

[18] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2010, pp. 1–9.

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

[20] S. Hu, M. Li, Q. Wang, S. S. M. Chow, and M. Du, "Outsourced biometric identification with privacy," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 2448–2463, 2018.

[21] X. Wang *et al.*, "Search me in the dark: Privacy-preserving Boolean range query over encrypted spatial data," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, 2020, pp. 2253–2262.

[22] M. S. Nair and M. S. Rajasree, "Fine-grained search and access control in multi-user searchable encryption without shared keys," *J. Inf. Security Appl.*, vol. 41, pp. 124–133, Aug. 2018.

[23] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, 2016, pp. 1–9.

[24] G. Xu, H. Li, H. Ren, X. Lin, and X. S. Shen, "DNA similarity search with access control over encrypted cloud data," *IEEE Trans. Cloud Comput.*, early access, Jan. 23, 2020, doi: 10.1109/TCC.2020.2968893.

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

[26] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. ACM STOC*, 2009, pp. 169–178.

[27] L. Xue, D. Liu, J. Ni, X. Lin, and X. S. Shen, "Balancing privacy and accountability for industrial mortgage management," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4260–4269, Jun. 2020.

[28] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "VeriML: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2524–2540, Oct. 2021.

[29] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[30] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related SQL range queries with privacy preserving," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 1596–1608, 2017.

[31] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1632–1644, Jul./Aug. 2021.

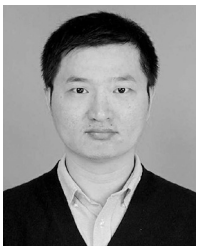
[32] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast and scalable range query processing with strong privacy protection for cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2305–2318, Aug. 2016.

[33] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using Rp-trees," in *Proc. IEEE ICDE*, Brisbane, QLD, Australia, 2013, pp. 314–325.

[34] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud," in *Proc. NDSS*, 2012, pp. 1–17.

[35] W. K. Wong, D. W.-I. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD*, 2009, pp. 139–152.

[36] K. Zhou and J. Ren, "PassBio: Privacy-preserving user-centric biometric authentication," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 3050–3063, 2018.
[37] C. Zhang, L. Zhu, C. Xu, J. Ni, C. Huang, and X. S. Shen, "Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, early access, May 17, 2021, doi: 10.1109/TMC.2021.3080714.
[38] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving efficient and privacy-preserving exact set similarity search over encrypted data," *IEEE Trans. Dependable Secure Comput.*, early access, Jun. 23, 2020, doi: 10.1109/TDSC.2020.3004442.
[39] F. Song, Z. Qin, D. Liu, J. Zhang, X. Lin, and X. Shen, "Privacy-preserving task matching with threshold similarity search via vehicular crowdsourcing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 7, pp. 7161–7175, Jul. 2021.
[40] J. Liang, Z. Qin, L. Xue, X. Lin, and X. Shen, "Verifiable and secure SVM classification for cloud-based health monitoring services," *IEEE Internet Things J.*, early access, Apr. 26, 2021, doi: 10.1109/jiot.2021.3075540.
[41] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Edge-aided computing and transmission scheduling for LTE-U-enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7881–7896, Dec. 2020.
[42] 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.
[43] P. Strobach, "Solving cubics by polynomial fitting," *J. Comput. Appl. Math.*, vol. 235, no. 9, pp. 3033–3052, 2011.
[44] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proc. IEEE S&P*, Berkeley, CA, USA, 2007, pp. 350–364.



Fuyuan Song received the B.S. degree from the College of Mathematics and Information Science, Jiangxi Normal University, Nanchang, China, in 2014. He is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China.

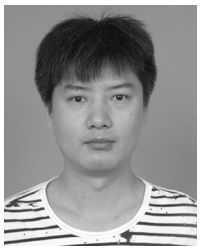
In 2019, he joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, as a Visiting Scholar. His research interests include cloud security and applied cryptography.

Zheng Qin (Member, IEEE) received the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in 2001.

He is a Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include machine learning, applied cryptography, and cloud computing.

Liang Xue (Graduate Student Member, IEEE) received the B.S. and M.S. degrees from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada.

Her research interests include applied cryptography, cloud computing, and blockchain.



Jixin Zhang (Member, IEEE) received the B.S. degree in mathematics and the M.S. degree in computer science and technology from Wuhan University of Technology, Wuhan, China, in 2011 and 2014, respectively, and the Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2019.

He is currently working with the School of Computer Science, Hubei University of Technology, Wuhan. His primary researches focus on machine learning and security.

Xiaodong Lin (Fellow, IEEE) received the Ph.D. degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1998, and the Ph.D. degree (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2008.

He is currently a Full Professor with the School of Computer Science, University of Guelph, Guelph, ON, Canada. His research interests include wireless communications and network security, computer forensics, privacy-enhancing technologies, blockchain, and applied cryptography.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular *ad hoc* and sensor networks.

Dr. Shen received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from Wireless Communications Technical Committee (2019) and the AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, and IEEE Globecom'07 and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the President Elect of the IEEE Communications Society. He was the Vice President for Technical and Educational Activities and Publications, a Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a member of IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, *IEEE Network Magazine*, and *IET Communications*. He is a Registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.