Secure Range Search Over Encrypted Uncertain IoT Outsourced Data

Cheng Guo, Ruhan Zhuang, Yingmo Jie, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Xinyu Tang

***Abstract*—Internet of Things (IoT) is an increasingly popular technological trend. The operation of IoT needs a strong data-handling capacity, where most of the data are sensor data. Limitations associated with measurement, delays in data updating, and/or the need to preserve the privacy of data can result in the sensor data being uncertain. Thus, one key challenge is "how do we ensure the privacy of data collected from IoT devices, particularly uncertain data, that are being outsourced to the cloud for analysis, storage and archival?". Searchable encryption scheme is a promising technique that allows the searching over encrypted (uncertain) data stored offshore. In this paper, we propose a secure range search for encrypted data from IoT devices. Specifically, we use homomorphic and order-preserving encryption to encrypt data published by the data owners. We then use the *k*-dimensional tree to build the data index. Our scheme is designed to ensure the privacy of the dataset, without affecting the efficiency of keyword search on the (encrypted) dataset. We also demonstrate that our scheme can preserve both data and query privacy, as well as evaluating its performance to demonstrate efficiency.**

***Index Terms*—Internet of Things (IoT), range search, secure range search, sensor data, uncertain data.**

I. Introduction

INTERNET of Things (IoT) devices, such as sensing devices (e.g., radio-frequency identification, infrared sensor, global positioning system, and laser scanners), can be used to facilitate intelligent identification, positioning, tracking, monitoring, and management. Such data (also referred to as sensor data) can be random and incomplete in nature, partly due to limitations of deployed measuring instruments or delays in data updating. In other words, the sensor data can be imprecise and uncertain. The ability to manage uncertain data efficiently is crucial in those working with databases, etc. Thus, how to efficiently process uncertain data is a topic of ongoing interest to researchers [1], [2].

Manuscript received January 15, 2018; revised March 29, 2018 and May 2, 2018; accepted June 5, 2018. Date of publication June 7, 2018; date of current version May 8, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61501080 and Grant 61572095, in part by the Cloud Technology Endowed Professorship, and in part by NSF CREST under Grant HRD-1736209. *(Corresponding author: Kim-Kwang Raymond Choo.)*

C. Guo, R. Zhuang, Y. Jie, and X. Tang are with the School of Software Technology, Dalian University of Technology, Dalian 116620, China, and also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Development Zone, Dalian University of Technology, Dalian 116620, China (e-mail: guocheng@dlut.edu.cn; clindy007@163.com; jymsf2015@mail.dlut.edu.cn; 1079062525@qq.com).

K.-K. R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA, and also with the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JIOT.2018.2845106

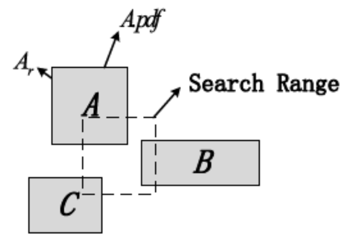


Fig. 1. Range search over sensor data.

Range search is a fundamental query performed on uncertain data, whose purpose is to retrieve data within the query range. One example application of range search in IoT is in agriculture [2], where farmers can install sensors in the field to monitor temperature changes, relative humidity, and pollution level information. Each sensor obtains a set of sensor data $U = \{u_1, \dots, u_n\}$, where U represents a sensor object, u_i represents an instance, and $i \in [1, n]$. Hence, each object contains three data values and is modeled as a 3-D object (Fig. 1).

There are three sensor objects $\{A, B, C\}$, due to factors, such as equipment failure and noise. The received sensor objects may be uncertain. As such, each object is represented by an uncertain region A_r (shaded areas in Fig. 1) and the probabilistic density function (PDF) ($A.pdf$ in Fig. 1). This implies that an object may appear in an uncertain region with the probabilities described by its PDF. Farmers cannot obtain precise data in practice. However, using range search, they can analyze and determine which range has abnormal conditions (e.g., fire hazard, waterlogging, and insect attacks). It is an effective way for them to have a real-time understanding of the conditions in the fields.

Existing research on range searches over multidimensional uncertain data with an arbitrary PDF [1], [3], [4] mainly follow the filtering and verification paradigm. By leveraging an effective index structure, some objects can be filtered at a threshold value without calculating their appearance probabilities in detail. Also, existing research generally focus on plaintext and does not consider data interaction and sharing.

An important medium for sensor data interaction and sharing in the IoT is the cloud, due to benefits that could be realized, such as cost efficiency, high-capacity, and the reduction of overhead. For example, data owners can potentially benefit from the outsourcing of the database to the cloud. A tradeoff is data owners ceding over the control of the query process. This clearly has security implications. It is also not safe for data owners to upload plaintext data. Encrypting the data prior to outsourcing is an effective security measure, although a tradeoff is reduced data utility. For example, searching on encryption datasets will be inefficient and impractical.

Searchable encryption (SE) schemes can be designed to search on encrypted data, such as the symmetric SE scheme of Song *et al.* [5] and the symmetric SE scheme in [6]. The SE schemes have been applied in a number of areas [7]–[9]. The uncertain and imprecise nature of IoT sensing data, however, complicate the design of efficient search schemes on such encrypted data.

In this paper, we are motivated by the challenge in designing a secure range search scheme to support the queries of uncertain outsourced IoT data. In [10], for example, the authors used *U-Quadtree* to organize the uncertain data in order to support the range search. They developed a cost model to build an effective quadtree. This tree would be unbalanced if the data in the dataset was uneven. The unbalanced tree would also incur significant storage and time overhead.

To solve this problem, we apply a k-dimensional tree (KD-tree) [11], or the binary space partitioning structure, to organize the sensor data. According to the data distribution, a KD-tree can split the dataset evenly and support an efficient range search. To support comparison and additive operations, we apply homomorphic and order-preserving encryption (OPE) encryption to encrypt the sensor data published by the data owners. This can be used to hide the access and search patterns and ensure data privacy. We use two cloud servers (C1 and C2) to support the range search process. Our scheme algorithm achieves a significant improvement in performance during a range search over the encrypted uncertain sensor data.

We consider the contributions in this paper to be the new SE scheme designed to facilitate secure and fast range search over uncertain sensor data, using KD-tree, OPE, and homomorphic encryption. In our scheme, we ensure the confidentiality of the dataset and the query, by hiding the search and access patterns.

The rest of this paper is organized as follows. In Section II, we introduce extant literature including related SE schemes, range search, and data privacy. We then introduce the relevant background information (i.e., OPE, homomorphic encryption, KD-tree, and uncertain sensor data) in Section III. Section IV presents the model of our scheme, the algorithms for the range search and the security analysis. The experimental analysis is given in Section V. Our conclusions are presented in Section VI.

II. Related Work

As previously discussed, SE scheme enables data owners to search on their encrypted data, say in the cloud (more specifically, search the data over a ciphertext domain). Existing SE schemes can be categorized into those based on public key-based cryptography [6], [12], [13], and those based on symmetric key-based cryptography [5], [14], [15]. Song *et al.* [5] proposed the first symmetric SE scheme, but many other SE schemes were proposed afterward [13], [16], [17]. These early works only support keyword search schemes, which are very simple in terms of functionality. As technologies advance, so does the complexity of data. For example, IoT data (e.g., sensor data), as well as the errors, or the limitations of the sensors, result in the obtained data being uncertain. However, early SE schemes are not capable of supporting searches over uncertain data.

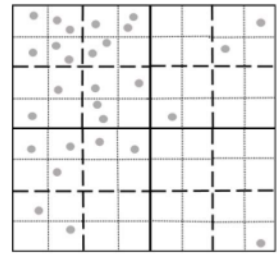


Fig. 2. Example of Quadtree.

Uncertain data management [18], [19] has gained traction among researchers, particularly due to the many practical applications in various domains. Range search is an effective way to conduct data analysis, by enabling a quick search of the most relevant data. Not surprisingly, a number of range search schemes over plaintext uncertain data have been presented in the literature in recent years [1], [3], [4]. Most existing schemes use some indexing techniques to improve the retrieval performance.

Common retrieval structures include R-tree [1], [3], U-tree [4], UI-tree [20], UP-index [21], Quadtree [10], and KD-tree [11]. The first four structures employ an "equality strategy," that is, the same amount of resources in terms of the index space usage are allocated to each uncertain object. Consequently, they cannot effectively address different uncertain region sizes during the index construction process. To overcome such a limitation, Zhang *et al.* [10] applied Quadtree to organize the uncertain data. Quadtree is a space partitioning tree data structure in which a d-dimensional space is recursively subdivided into 2^d regions. In each iteration of the partitioning process, the space will be divided into 2^d equal parts. Fig. 2 is an example of Quadtree. The data points are in a 2-D space and the space is recursively divided into four regions. In Fig. 2, the data points are uneven, which leads to some useless partitions. However, it will increase the space and time overhead. As discussed earlier, existing schemes only support range searching over plaintext uncertain data. In other words, such schemes are ineffective on encrypted uncertain data.

In this paper, we apply KD-tree, a binary space partitioning structure, to organize the uncertain sensor data. KD-tree is mainly used in multidimensional space data retrieval (e.g., range and nearest neighbor searches). It can achieve efficient retrievals by solving defects in other indexing structures.

Because data from the diverse IoT devices (e.g., sensors) is uncertain, a range search over the encrypted data should support some basic operations.

Many existing efficient encryption primitives can support different operations in the ciphertext domain. Paillier [22], for example, proposed a homomorphic encryption scheme to support additions. OPE [23], [24] can evaluate comparisons. BGN (the abbreviation for the authors' name) encryption [25], proposed by Dan *et al.*, or the more recent novel approach in [26], can support an unlimited number of additions and only one multiplication.

In this paper, we apply OPE and homomorphic encryption simultaneously to encrypt the sensor data. Data owners obtain uncertain data from the IoT devices. They then use the KD-tree to organize the data. To ensure data privacy, they will use the OPE and homomorphic encryptions to encrypt the KD-tree and the dataset. Such data can then be outsourced to the cloud. When users wish to perform a range search, they should encrypt the query and then send that query to the cloud. When the cloud receives the query, it will conduct a search over the KD-tree and return the encrypted results to the users. Users use their own secret key to decrypt the results and choose the results they want. The detailed algorithm will be presented in Section IV.

III. Preliminaries

In this section, we revisit the homomorphic encryption [22], OPE [24], and KD-tree, prior to presenting the security definitions for our scheme.

A. Homomorphic Encryption

The homomorphic encryption system [22] is an additive homomorphic and probabilistic asymmetric encryption scheme, based on the higher-order residue class problem. It contains three stages, namely: 1) key generation; 2) encryption; and 3) decryption.

Let pk be the public key given by (N, g), where N is the product of two large primes and g is in $\mathbb{Z}^*_{N^2}$. Let E_{pk} be the encryption function with public key pk and D_{sk} be the decryption function with secret key sk. Given plaintext $a, b \in \mathbb{Z}_N$, this system has the following properties, namely: homomorphic addition and homomorphic multiplication.

Homomorphic Addition:

$$E_{pk}(a + b) = E_{pk}(a) * E_{pk}(b) \bmod N^2.$$

Homomorphic Multiplication:

$$E_{pk}(a * b) = E_{pk}(a)^b \bmod N^2.$$

The Paillier encryption system has been shown to be semantically secure, where an adversary cannot infer any information about the plaintext from the given ciphertexts.

B. Order-Preserving Encryption

OPE [24] is a special type of encryption, where the orders of the encrypted data are the same as the orders of their plaintext. This property makes it possible to sort and rank the encrypted data without revealing the plaintext. The ideal security of OPE is defined with indistinguishability under chosen-plaintext attacks (IND-CPA). It has been recently achieved by [23] and [27]. Let pk be the public key and E_{pk} be the encryption function. Given plaintext $a, b \in \mathbb{Z}_N$, OPE can insure that, if $a > b$, then $E_{pk}(a) > E_{pk}(b)$.

C. KD-Tree

A KD-tree is a data structure for indexing k-dimensional point data distributed in a k-dimensional space. It can be considered a k-dimensional binary search tree [11]. It is also a good solution to the space partitioning problem. Every node in a KD-tree is a k-dimensional point. Every nonleaf node can be considered to implicitly generate a splitting hyperplane that divides the space into two parts. Points to the left of this hyperplane is represented by the left subtree of that node and the other hyperplane is represented by the right subtree.

All nodes in the tree are associated with one of the k-dimensions and the hyperplane of each dimension is perpendicular to that dimension's axis. The first step is calculating the variance of each of these dimensions based on the points' values. We choose the maximum value of the variances and define the corresponding dimension by the splitting hyperplane direction. The points will be sorted by the value of the dimension corresponding to the maximum value of the variances. For example, if the "x" axis is chosen for a particular split, all points in the subtree with a smaller x value than the node will be in the left subtree and all points with a larger x value will be in the right subtree. We can recursively run the methods to construct the KD-tree. Selecting the splitting hyperplane direction based on the variance can guarantee that all the points can be split uniformly.

Fig. 3 is an example of a KD-tree, where the uncertain sensor object set is $\{A, B, C\}$, each object has five points (instances), and the points are in a 2-D space. It has two splitting hyperplane directions: an x-axis and a y-axis. By calculating the variances of these two dimensions, we determine that the variance of the x-dimensional space is bigger. We set the splitting hyperplane direction as the x-axis, with the point (5, 6.5) as the median point. The points with a smaller x value than "5" will be in the left subtree and the points with a larger x value than 5 will be in the right subtree. We should recursively construct the left and the right subtree until there is no point to be split.

Fig. 3(a) represents the partitioning of the space, and Fig. 3(b) represents the corresponding KD-tree. Each node consists of one instance and its corresponding range.

D. Security Definition

The main security objective of our scheme is to preserve both data and query privacy from untrusted cloud servers, which can be informally explained as follows.

1) *Data Privacy:* Given two encrypted datasets, D_0 and D_1, an adversary cannot distinguish between these two datasets.
2) *Query Privacy:* Given two search tokens, Q_0 and Q_1, an adversary cannot distinguish between these two queries.

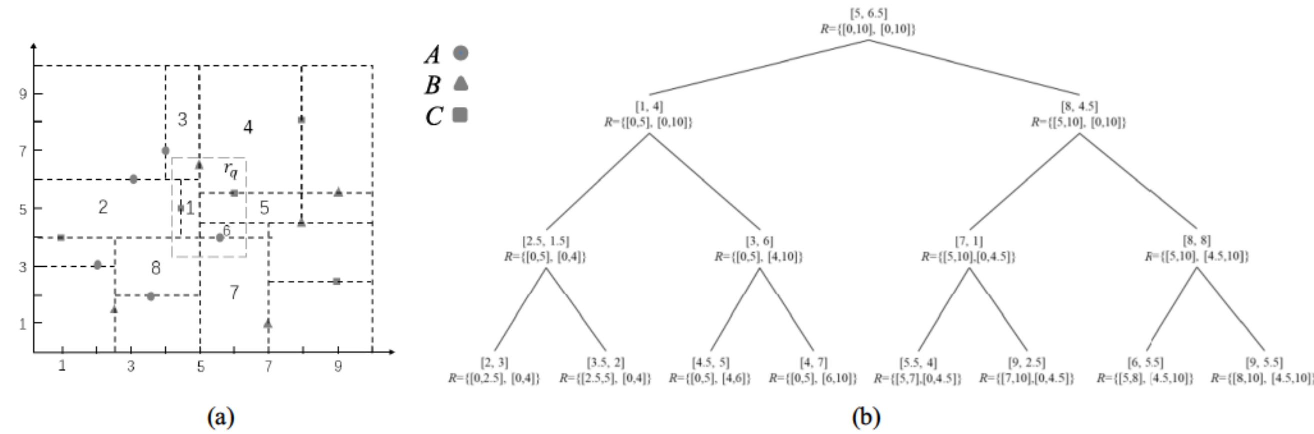


Fig. 3. Construction of the KD-tree based on three uncertain sensor objects: *A* {[2,3] [5.5,4] [4,7] [3,6] [3.5,2]}, *B* {[2.5,1.5] [5,6.5] [7,1] [9,5.5] [8,4.5]}, and *C* {[1,4] [4.5,5] [9,2.5] [8,8] [6,5.5]}. Each object has five instances and the probability of each instance is 0.2.

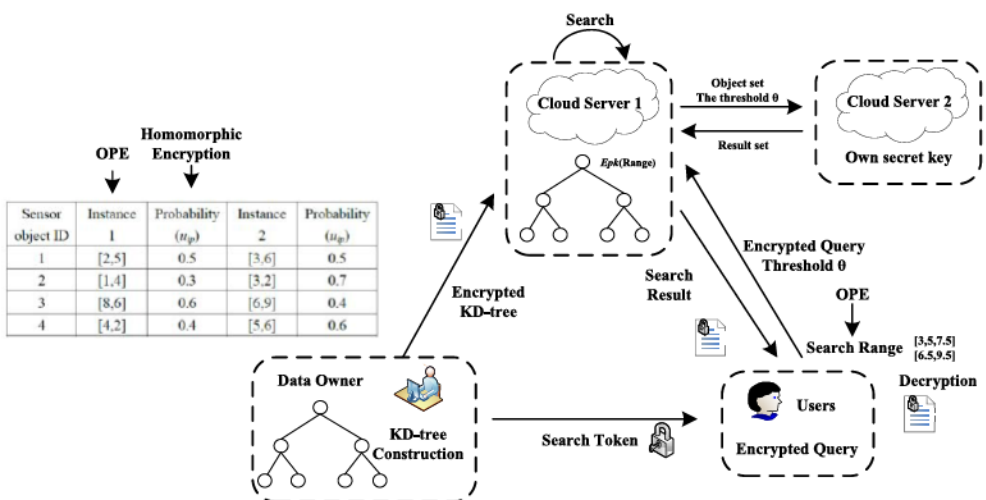


Fig. 4. Model of the secure range search over encrypted sensor data.

The rigorous definitions of our data and query privacy, with indistinguishability under IND-CPA, and its corresponding leakage function, are presented in Section V.

IV. Model of the Proposed Scheme

In this section, we describe our proposed secure range search model and then briefly introduce the general process of our scheme. Then, we provide the definition of a range search over the encrypted sensor data. The algorithm will be presented in Section V.

A. Model of the Scheme

In this paper, our scheme consists of four entities: 1) data owner; 2) cloud server 1 (C1); 3) cloud server 2 (C2); and 4) user. The model is illustrated in Fig. 4. The model illustrates that the sensor object consists of a sensor object ID, a set of instances and the probability of each instance. Each instance is a d-dimensional with its own coordinate. The data owner has a collection of data sets to be outsourced to the cloud server in the encrypted form. To enable the searching capability over encrypted data, the data owner will first build an encrypted KD-tree with the data sets. Then, the encrypted data sets and encrypted KD-tree will outsource to C1. When a user wants to do a range search, the user will encrypt the query and then send to C1. C1 and C2 will cooperate with each other to search over the encrypted KD-tree and then return the results to the user.

Our secure range search scheme consists of the following six polynomial-time protocols.

1) *GenKey*$(1^\lambda) \rightarrow \{sk_{OPE}, pk_{HE}\}$*:* Given a security parameter λ, the data owner computes and outputs: $sk_{\text{OPE}} \leftarrow$ OPE.GenKey(1^λ) and $pk_{\text{HE}} \leftarrow$ HE.GenKey(1^λ), where HE denotes homomorphic encryption.
2) *BuildTree*$(\mathcal{U}) \rightarrow \Gamma$*:* Given an object set $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, each object U has m instances, denoted by $U = \{u_1, u_2, \ldots, u_m\}$. There are $m \times n$ instances. This protocol uses all instances to construct a KD-tree. Each node consists of one instance and its corresponding range $R = \{[r_{11}, r_{12}], [r_{21}, r_{22}], \ldots, [r_{d1}, r_{d1}]\}$. The range is calculated based on the coordinate of the instance.

3) *Enc* $(sk_{OPE}, pk_{HE}, \Gamma) \rightarrow \Gamma^*$: Given a secret key sk_{OPE}, a public key pk_{HE}, and a KD-tree Γ. In the following range search process, we should compare the value of the instances' coordinates and conduct an additive operation on the probabilities. The data owner traverses the KD-tree to encrypt each node with

$$\text{OPE.Enc}([D_{i1}, D_{i2}, \ldots, D_{id}]) \rightarrow [eD_{i1}, eD_{i2}, \ldots, eD_{id}]$$
$$\text{OPE.Enc}(\{[r_{11}, r_{12}], [r_{21}, r_{22}], \ldots, [r_{d1}, r_{d1}]\}) \rightarrow eR$$

to obtain each instance's encrypted coordinate and the encrypted range, where $[D_{i1}, D_{i2}, \ldots, D_{id}]$ is each dimension value of the instance i. The data owner then runs

$$\text{HE.Enc}(u_{ip}) \rightarrow eu_{ip}$$

to obtain each instance's encrypted probability. The data owner outputs an encrypted KD-tree Γ^*.

4) *GenToken* $(sk_{OPE}, pk_{HE}, r_q) \rightarrow er_q$: Given the secret key sk_{OPE}, a range search $r_q = \{[qr_{11}, qr_{12}], [qr_{21}, qr_{22}], \ldots, [qr_{d1}, qr_{d2}]\}$ and a probabilistic threshold θ, where $[r_{j1}, r_{j2}]$ denotes the range of the jth dimension, $1 \leq j \leq d$. The user encrypts it as

$$\text{OPE.Enc}(\{[qr_{11}, qr_{12}], [qr_{21}, qr_{22}], \ldots [qr_{d1}, qr_{d2}]\})$$
$$\rightarrow \{[eqr_{11}, eqr_{12}], [eqr_{21}, eqr_{22}], \ldots, [eqr_{d1}, eqr_{d2}]\}$$
$$\text{HE.Enc}(\theta) \rightarrow e\theta$$

and outputs er_q as a search token.

5) *Search* $(\Gamma^*, er_q) \rightarrow eU_q$: Given the search token er_q and an encrypted KD-tree Γ^*, C1 starts from the root node, and traverses Γ^* to calculate the upper and lower appearance probability of each object regarding er_q and sends these to C2. The detailed search process will be given in Section V. The cloud servers cooperate with each other to output a sensor object set, which satisfies the search token.

6) *Dec*$(sk_{OPE}, pk_{HE}, eU_q) \rightarrow rU_q$: Given secret key sk_{OPE}, public key sk_{HE} and the object set returned from the cloud server. The user runs this protocol to obtain the final sensor object set, which satisfies the probabilistic threshold θ.

B. Problem Definition

In this section, we define uncertain sensor data. Table I summarizes the notations frequently used throughout this paper.

The uncertain sensor data (object) is represented by its possible points and the probability that it may appear at each point. All the points in this paper are in a d-dimensional numerical space. In particularly, an uncertain sensor object can be described either continuously or discretely. We will introduce these two conditions as follows.

In the continuous case, a sensor object U is described by its probability density function (PDF) $U.pdf$ and its uncertain region U_r. $U.pdf(x)$ denotes the probability of U appearing at point x, yielding $\int_{x \in U_r} U.pdf(x)dx = 1$. Sometimes, the PDF of the sensor object may not be available, and hence, a sensor object is represented by a set of sampled points,

TABLE I
SUMMARY OF NOTATIONS

Notation	Definition
U (Y)	Sensor objects (a set of sensor objects)
eU	Encrypted sensor object
n	Number of uncertain objects
$\{[qr_{11}, qr_{12}],\ldots,[qr_{d1},qr_{d2}]\}$	Range search region r_q
u_{ip}	The probability of U appears at instance u_i
eu_{ip}	Encrypted probability of U appears at u_i
$[D_{i1}, D_{i2},\ldots,D_{id}]$	The coordinate of instance i
$\{[r_{11},r_{12}],\ldots,[r_{d1},r_{d2}]\}$	The range of a node
θ	Probabilistic threshold
$P(U, r_q)$	The appearance probability of U regarding r_q
$eP(U, r_q)$	Encrypted appearance probability regarding r_q
$LP(U, r_q)$	The lower bounds of $P(U, r_q)$
$eLP(U, r_q)$	Encrypted lower bounds of $P(U, r_q)$
$UP(U, r_q)$	The upper bound of $P(U, r_q)$
$eUP(U, r_q)$	Encrypted upper bound of $P(U, r_q)$

which is the discrete case. A sensor object contains a set of instances (points) $U = \{u_1, u_2, \ldots, u_m\}$, u_{ip} denotes the probability of U appearing at instance u_i and $\sum_{u \in U} u_p = 1$.

For a point p and a region r, $p \in r$ means that r contains p. For any two regions r_1 and r_2, $r_2 \subseteq r_1$ if $r_1 \cup r_2 = r_1$ means r_1 contains r_2. We say $r_1$overlaps r_2 if $r_2 \not\subset r_1$ and $r_1 \cap r_2 \neq \emptyset$.

For presentation simplicity, we concentrate on the discrete case in this paper. Nevertheless, all techniques developed in this paper can be applied to the continuous case.

Below is the definition of a "probabilistic threshold range search," which is equivalent to a "range search" in the rest of paper.

Definition 1 (Probabilistic Threshold Range Search): Given a set $\mathcal{U}$ of sensor objects and a user specified probabilistic threshold θ, the probabilistic threshold range search retrieves all objects $U \in \mathcal{U}$ with $P(U, r_q) \geq \theta$ $(0 \leq \theta \leq 1)$.

In this paper, we concentrate on the problem of a probabilistic threshold range search over encrypted multidimensional sensor objects. We aim to develop an effective indexing structure to facilitate the range search process.

V. Range Search Over Encrypted Sensor Objects

A. Main Idea

As specified previously, the main process of our scheme can be summarized as follows.

The data owner obtains the sensor dataset from the sensors, where the dataset contains n uncertain sensor objects and each object contains m instances. Each instance is a triplet $(u.o, u.D, u_p)$, where $u.o$ denotes the object it belongs to, $u.D$ denotes the coordinate of this instance, and u_p denotes the probability of this instance. The data owner constructs a KD-tree based on the instances. Each node is a two-tuple $(u.o, u.R)$, where u denotes the instance in this node and $R = \{[r_{11}, r_{12}], [r_{21}, r_{22}], \ldots, [r_{d1}, r_{d1}]\}$ denotes the range of its area, as in Fig. 3(b), where d denotes the d-dimensional space. The range of each node is calculated based on the coordinate of the instance $n.u.D$.

The data owner runs the encryption function mentioned in Section IV to encrypt the KD-tree and output the encrypted KD-tree Γ^*, where the two-tuple of each node is denoted by $(n.eu, n.eR)$. The data owner then outsources the encrypted KD-tree Γ^* to the C1. If a user wants to conduct a range search, then he/she should use the secret key of OPE to encrypt the query r_q and then send the encrypted search token er_q to C1.

When C1 gets a search token, it will traverse Γ^* to calculate the encrypted lower and upper bounds of the probability for all objects and then send it to C2. C2 decrypts these probabilities and compares them to the user's probabilistic threshold θ. It chooses the objects that satisfy the requirements. Then, C2 encrypts the result object set and sends it to C1. C1 reconfirms the results and then sends the set to the user. The user decrypts the results to obtain the objects. During this time, C2 will follow the filtering-and-verification process to choose the objects that satisfy his/her requirements. A sensor U may be filtered in either of the following ways.

1) U is pruned if $UP(U, r_q)$ is smaller than the given probabilistic threshold θ.
2) U is validated if $LP(U, r_q)$ is not less than θ.

where $LP(U, r_q)$ and $UP(U, r_q)$ denote the lower and upper bounds of $P(U, r_q)$, respectively. Only the objects that survive the filtering phase need to be verified [i.e., explicitly computing $P(U, r_q)$].

B. KD-Tree-Based Range Search

Theorem 1 indicates that we can derive $LP(U, r_q)$ and $UP(U, r_q)$, based on the topological relationship between r_q and the range of each node.

Theorem 1: Given a search token er_q and an encrypted KD-tree Γ^*, let $N_1 (N_2)$ denote the node set contained (overlapped) by er_q

$$eLP\left(U, r_q\right) = \prod n.u.eu_p, \text{ where } n.eR \in N_1$$

$$eUP\left(U, r_q\right) = \prod n.u.eu_p, \text{ where } n.eR \in N_1 \cup N_2.$$

Proof: Because the probability of each instance u_p is encrypted by a homomorphic encryption, we should use the property of homomorphic addition to calculate $eLP(U, r_q)$ and $eUP(U, r_q)$. For any node n, we have $n.u \in er_q$ if the range of node n is contained by er_q. Immediately, $\mathrm{Dec}(eP(U, r_q)) \geq \mathrm{Dec}(\prod n.u.eu_p)$, where Dec() is the decryption function and $n.eR \in N_1$. Given a node n, if $n.eR$ is not contained or overlapped by er_q, then we have $n \notin er_q$. This implies $\mathrm{Dec}(eP(U, r_q)) \leq 1 - \mathrm{Dec}(\prod n.u.eu_p)$, where $n.eR \notin N_1 \cup N_2$. Because $\mathrm{Dec}(\prod_{u \in U} eu_p) = 1$, we have $\mathrm{Dec}(eP(U, r_q)) \leq \mathrm{Dec}(\prod n.u.eu_p)$, where $n.eR \in N_1 \cup N_2$. Therefore, the theorem holds. ■

Example 1: In Fig. 3(a), given a search region $r_q q$, according to Theorem 1, only the area 1 is contained in r_q The area 2, 3, 4, 5, 6, 7, and 8 is overlapped by r_q. The range of each node and the search region r_q are encrypted by OPE, so we can also compare the range over the ciphertext. We can now obtain $eLP(C, r_q) = \mathrm{Enc}(0.2), eUP(C, r_q) = \mathrm{Enc}(0.2) \times \mathrm{Enc}(0.2) \times \mathrm{Enc}(0.2) = \mathrm{Enc}(0.6)$. When the user obtains the resulting set, he/she will decrypt it to obtain the objects' $LP(U, r_q)$ and $UP(U, r_q)$. This will then be compared with his/her own probabilistic threshold θ. Consequently, C is pruned if $\theta = 0.8$ and C is validated if $\theta = 0.2$. Hence, we need to verify C if $\theta = 0.4$.

Algorithm 1 Range Search (r*,erq)

Input:
Γ^*: the encrypted KD-tree
$er_q\{[eqr_{11}, eqr_{12}], \ldots, [eqr_{d1}, eqr_{d2}]\}$ and $e\theta$
Output:
the encryptcd object sets R.
1: $R := \emptyset$; $V := \emptyset$;
2: C1 do
3: **for** each dimension $i, i \in [1, d]$ **do**
4: traverse Γ^* each node n
5: $U \leftarrow n.u.o$
6: $ep \leftarrow n.u.eu_p$
7: **if** $eqr_{i1} \leq n.er_{i1} \leq eqr_{i2}$ or $eqr_{i1} \leq n.er_{i2} \leq eqr_{i2}$ **then**
8: $eUP(U, r_q) := eUP(U, r_q) \times ep$
9: **if** $eqr_{i1} \leq n.er_{i1}$ and $eqr_{i2} \geq n.er_{i2}$ **then**
10: $eLP(U, r_q) := eLP(U, r_q) \times ep$
11: **end if**
12: **end if**
13: **end for**
14: C1:object set $\{U_1, U_2, \ldots, U_j\}, j \leq n \xrightarrow{send}$ C2
15: C2: use sk_{HE} to get $\{LP(U_j, r_q), UP(U_j, r_q)\}$
16: **for** each object U **do**
17: **if** $LP(U, r_q) \geq \theta$ **then**
18: $R := R \cup U$
19: **else**
20: **if** $UP(U, r_q) \geq 0 \geq LP(U.r_q)$ **then**
21: $V := V \cup U$
22: **end if**
23: **end if**
24: **end for**
25: C2: encrypts set V then, sends to C1
26: **for** each $U \in V$ **do**
27: **for** each dimension $i, i \in [l, d]$ each instance $u_j, j \leq m$ **do**
28: **if** $eqr_{i1} \leq eu_j.eD_i \leq eqr_{i2}$ **then**
29: $eP(U, r_q) = eP(U, r_q) \times eu_j.ep$
30: **end if**
31: **end for**
32: **end for**
33: C1:$U_j \in V, \{eP(U_j, r_q)\} \xrightarrow{send}$ C2
34: C2:$\{P(U_j, r_q)\} \xleftarrow{sk_{HE}} \{eP(U_j, r_q)\}$
35: **for** each $U \in V$ **do**
36: **if** $P(U, r_q) \geq \theta$ **then**
37: $R := R \cup U$
38: **end if**
39: **end for**
40: C2: encrypts set R and sends to C1
41: C1: return set R to user

Algorithm 1 details the range search following the filtering-and-verification paradigm. Lines 3–14 for C1 traverse Γ^* to calculate each object's $eLP(U, r_q)$ and $eUP(U, r_q)$ values. C1 sends the object set $\mathcal{U}$ to C2, who then uses sk_{HE} to decrypt $eLP(U, r_q)$ and $eUP(U, r_q)$.

According to Theorem 1, we arrive at the lower and upper bounds of the appearance probabilities of the objects. We can validate an object U if $LP(U, r_q) \geq \theta$ (line 18). We only need to verify the remaining objects set V, in which $LP(U, r_q) \geq \theta \geq UP(U, r_q)$ (line 21). C2 encrypts set V and sends it to C1, C1 calculates the $eP(U, r_q)$ value of each object in set V

(lines 26–32). And then, C1 sends set V to C2. C2 decrypts it and compares each object's $P(U, r_q)$ with θ. The objects which $P(U, r_q) \geq \theta$ (line 36) will be added to the resulting set R. C2 encrypts the resulting set R and then sends it to C1. C1 will return R to the user.

When the sensors obtain more data set, the KD-tree should be updated duly. If a little bit of data should be inserted to the KD-tree, the data owner can encrypt the data set and then upload it to C1. Each level of the KD-tree contains the splitting hyperplane direction. When C1 receives the encrypted data set, it will insert each data into Γ^*. The process of inserting data can be summarized as follows.

C1 traverses Γ^* from the root node and compares the value on the corresponding splitting hyperplane direction. If the value is smaller than the root node, it should traverse its left node until the data can be inserted to a leaf node.

The essence of the KD-tree is a balanced binary tree. Inserting plenty of the data will destroy the balance. So, the data owner should reconstruct the KD-tree with the old and new data if large volume of data will be updated.

C. Security Analysis

Prior to analyzing the security of the proposed scheme, we will provide some necessary definitions.

1) Concepts Definition:

a) Leakage function $\mathcal{L}$: In an SE scheme, a leakage function covers all the possible leakages revealed during the search process. The leakage function of a sensor object set $\mathcal{U}$ introduced by query r_q, can be denoted as $\mathcal{L}(\mathcal{U}, r_q)$.

In our scheme, the leakage function contains an access pattern (i.e., the identifiers of the encrypted data that are retrieved for each query), search pattern (i.e., whether the same encrypted result is retrieved by the two different queries), and a path pattern (i.e., the path that the search algorithm traverses in the KD-tree). The security of the data and query privacy in our scheme is defined as follows.

Definition 2 (IND-CPA Data Privacy): Let $\prod = $ (GenKey, BuildTree, Enc, GenToken, Search, Dec) be a probabilistic secure range search scheme over security parameter λ. We define a secure game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Init: $\mathcal{A}$ submits two sensor datasets $\mathcal{U}_0$ and $\mathcal{U}_1$ with the same size and isomorphic tree structure $\Gamma_0 \simeq \Gamma_1$, where $\mathcal{U}_0 = \{U_{01}, U_{02}, \ldots, U_{0n}\}$, $\mathcal{U}_1 = \{U_{11}, U_{12}, \ldots, U_{1n}\}$, for $1 \leq i \leq n$, $U_{01}, U_{02}, \ldots, U_{0n}$ and $U_{11}, U_{12}, \ldots, U_{1n}$ are all distinct, $\Gamma_0 \leftarrow \text{BuildTree}(\mathcal{U}_0)$, and $\Gamma_1 \leftarrow \text{BuildTree}(\mathcal{U}_1)$.

Setup: Challenger $\mathcal{C}$ runs GenKey(1^λ) to generate a public key pk and a secret key sk. It keeps these keys private.

Phase 1: Adversary $\mathcal{A}$ adaptively submits a few requests. Each request is one of the two following types.

1) *Ciphertext Request:* On the jth ciphertext request, adversary $\mathcal{A}$ outputs a dataset $\mathcal{U}'_j$, where $\mathcal{U}'_j = \{U'_{j,1}, U'_{j,2}, \ldots, U'_{j,n}\}$, for $1 \leq i \leq n$. $\mathcal{C}$ responds with an encrypted tree $\Gamma'^*_j = \text{Enc}(sk, pk, \Gamma'_j)$, where $\Gamma'_j \leftarrow \text{BuildTree}(\mathcal{U}'_j)$.
2) *Token Request:* On the jth token request, $\mathcal{A}$ outputs a range search r_{qj}. $\mathcal{C}$ responds with a search token $er_{qj} = \text{GenToken}(sk, pk, r_{qj})$.

Challenge: With $\mathcal{U}_0$ and $\mathcal{U}_1$, $\mathcal{C}$ flips a coin $b \in \{0, 1\}$, computes $\Gamma_b \leftarrow \text{BuildTree}(\mathcal{U}_b)$, and returns Γ^*_b to adversary $\mathcal{A}$.

Phase 2: Adversary $\mathcal{A}$ continues to submit a number of requests adaptively, which are still subjected to the same restrictions of Phase 1.

Guess: The adversary takes a guess b' of b.

We say that $\prod$ is secure against INC-CPA in relation to data privacy if, for any polynomial time adversary in the above game, it has, at most, a negligible advantage

$$\text{Adv}_{\prod,\mathcal{A}}^{\text{IND-CPA-Data}}(1^\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \text{negl}(\lambda)$$

where negl(λ) denotes a negligible function [28] in λ.

The definition of IND-CPA query privacy is similar to the previous definition; due to space limitations, we omit the detail.

2) Security Analyses: We now analyze the security of our scheme by following the preceding security games. We know that a homomorphic encryption scheme can against IND-CPA, so our scheme is IND-CPA data secure, as long as the homomorphic encryption is IND-CPA secure.

Proof: We simulate the security game defined in Definition 2 with an adversary $\mathcal{A}'$ from the ideal security game of OPE and HE. We then demonstrate that compromising the IND-CPA data privacy of our scheme is equivalent to compromising the IND-CPA of OPE.

Following Definition 2, the security game of our scheme is simulated by multiple instances of homomorphic encryption. As a result, $\mathcal{A}'$ could not distinguish between the two datasets, $\mathcal{U}_0$ and $\mathcal{U}_1$, as long as any pair of the two messages could be distinguished in the security game

$$\begin{aligned}\text{Adv}_{\prod,\mathcal{A}}^{\text{IND-CPA-Data}}(1^\lambda) &\leq q \cdot \text{Adv}_{\text{HE},\mathcal{A}'}^{\text{IND-CPA-Data}}(1^\lambda)\\ &\leq q \cdot \text{negl}(\lambda)\\ &\leq \text{negl}'(\lambda)\end{aligned}$$

where q denotes the number of homomorphic encryption instances needed in the game. This demonstrates the IND-CPA data security of our scheme. The proof for our scheme for IND-CPA query privacy is similar to the previous data privacy proof. ∎

VI. Performance Evaluation

In this section, we evaluate the performance of our scheme for different parameter settings. We implement the KD-tree, OPE, homomorphic encryption, and range search scheme in Java. Various experiments are run on a PC Intel Core 2.50 GHz CPU with 12 G memory.

In the experiment, there are four real sensor datasets that contain $20K$, $72K$, $168K$, and $336K$, respectively. The points in the first three datasets are 2-D and represent the location information in the United States (e.g., Los Angeles, CA, USA), which are available at: https://www.census.gov/geo/maps-data/data/tiger.html. There were 16000 3-D points included

TABLE II
SYSTEM PARAMETERS

Notation	Definition
n	Number of objects
m	Number of instances for each object
k	Dimension value
R_q	Range of query
θ	The probabilistic threshold

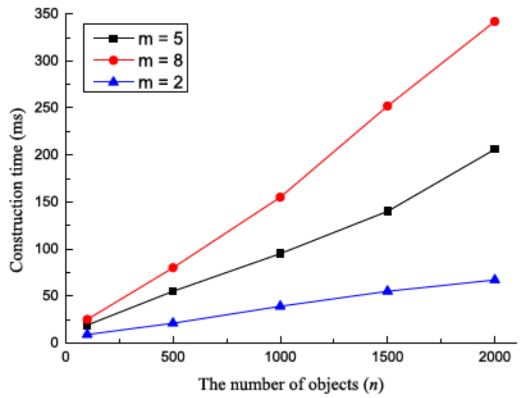


Fig. 5. Diff. n and $m = 2, 5, 8$, respectively.

in the fourth dataset, containing 2000 objects, where each object has eight instances. The data are available at https://archive.ics.uci.edu/ml/datasets.html. The dimensions represent the farm soil quality affected by three factors (i.e., air humidity, soil temperature, fertilizer application amount). We also generate a synthetic dataset to evaluate our scheme more precisely. The dimensionality varies from 2 to 6, and was named 2-D, 3-D, 4-D, 5-D, and 6-D, respectively. The object size varies from 100 to 2000 and the instance size of each object varies from 2 to 8.

All dimensions are normalized to domain [0, 400]. The query is a rectangle, or cuboid, which changes following the dimensions. The query range of each dimension varies from 10 to 50. The OPE and the homomorphic encryption key size were set to 128 bits.

Table II lists the parameters used in our performance evaluation.

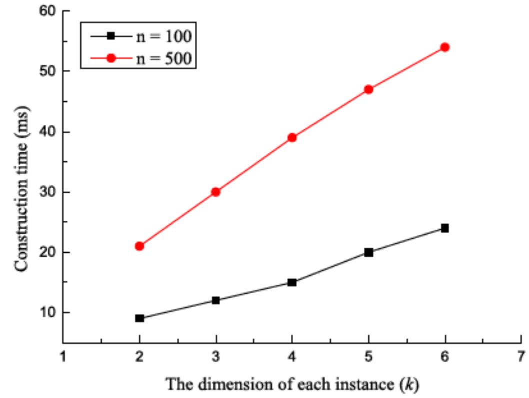


Fig. 6. Diff. k and $n = 100, 500$, respectively.

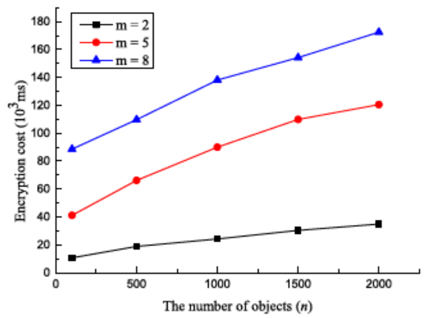


Fig. 7. Diff. n and $m = 2, 5, 8$, respectively.

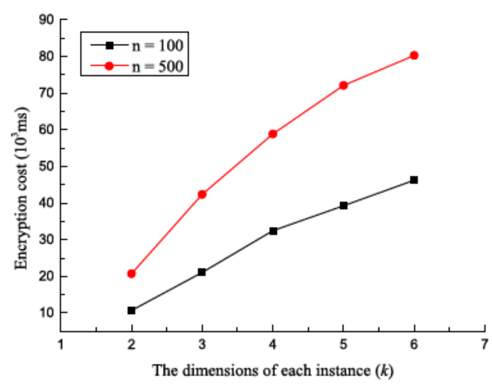


Fig. 8. Diff. k and $n = 100$ and 500, respectively.

A. Construction of the KD-Tree

For fairness, we evaluate the efficiency of the KD-tree construction process based on n, m, and k in the experiments.

Fig. 5 shows the construction time of the KD-tree for 15 datasets, where n varies from 100 to 2000 and m equals 2, 5, and 8, respectively. As expected, the construction time grows with n. When $n = 2000$, m = 800, the number of instances is 16000, and the construction time of the KD-tree is only 350 ms. The construction time increases with m, because each node only stores one instance. When the number of instances increases, the height of the tree will increase.

Fig. 6 shows the construction time of the KD-tree for ten datasets. From the results, we can see that the construction time increases linearly with the dimension k. Based on the property of the KD-tree (see Section III), we know that it will calculate each dimension's variance value in each partition. Hence, the computation cost will increase if the dimension is increased.

B. Encryption of the KD-Tree

By fixing $k = 2$, we evaluate the encryption efficiency of our scheme as n varies. Fig. 7 shows that the size of the object set varies from 100 to 2000, and the cost of the encryption increases almost linearly with n. This result reveals that when m varies from 2 to 8, the encryption time will increase, because each node represents one instance. If the number of instances grows, the height of the KD-tree grows with it.

The value of k is also an important factor that has an impact on encryption efficiency. As shown in Fig. 8, if the value of k varies from 2 to 6, the encryption cost increases linearly with

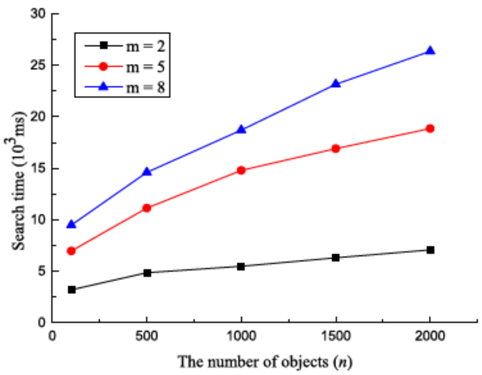


Fig. 9. Diff. n and $m = 2, 5, 8$, respectively.

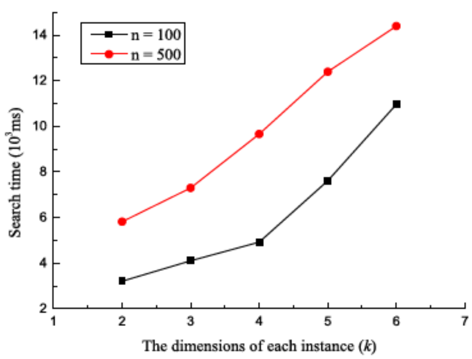


Fig. 10. Diff. k and $n = 100$ and 500, respectively.

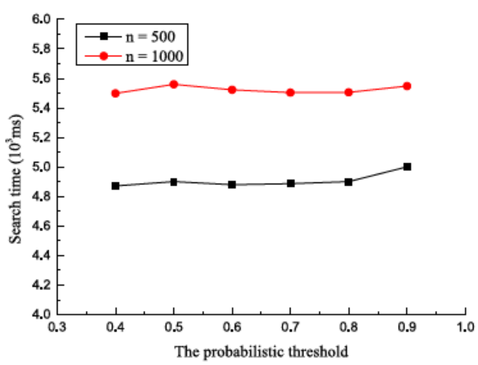


Fig. 11. Diff. θ and $n = 500$ and 1000, respectively.

it. When k is growing, OPE will be growing. The encryption cost will increase, as$k$ increases when n is bigger.

C. Evaluate Range Query

Fig. 9 reports the average query response time against the number of objects n. We fix $k = 2$, $\theta = 0.4$ and the query range R of each dimension $qr_{d2} - qr_{d1} \leq 40$. From Fig. 9, we can see that the performance of $m = 8$ is more sensitive to the growth of n, as compared with $m = 2$. This is because, when m is bigger, the number of instances will increase with an increase in n. The KD-tree will be deep and the space will be divided smaller. Hence, the search time will increase.

Fig. 10 shows that the search time will increase linearly with the value of k. The performance is more sensitive to the growth of k.

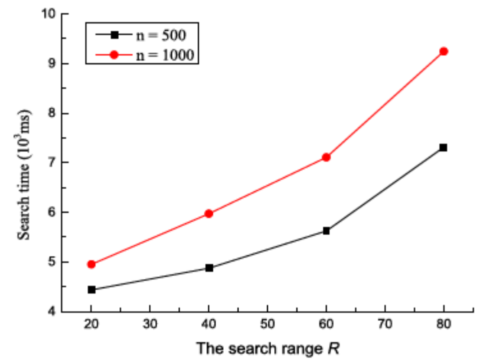


Fig. 12. Diff. search range R and $n = 500$ and 1000, respectively.

Fig. 11 shows that the performance of the algorithms is not sensitive to the probabilistic threshold θ. It is because the early calculation process costs lots of time, this process filters the most points. So, compare with θ will cost less time. We fix $k = 2$ and $m = 2$. The search range R varies from 20 to 80. Fig. 12 shows that the search time grows exponentially as R grows. This occurs because if R is bigger, the number of the nodes which are visited will increase, and hence, the number of calculations will increase.

VII. Conclusion

The diversity and range of IoT devices will grow as they are deployed in a broader range of applications, ranging from civilian (e.g., smart cities and emergency response) to military and battlefield (e.g., Internet of Military Things and Internet of Battlefield Things) and so on. This reinforces the need to efficiently manage uncertain and increasing amount of data from the IoT devices.

To ensure the security of uncertain IoT data, particularly those outsourced to the cloud or the edge, we developed an effective indexing technique to support range searches on multidimensional encrypted data. Specifically, in the proposed scheme, we used the KD-tree to organize the objects to improve the retrieval efficiency. To support operations over ciphertext, we used an OPE and homomorphic encryption scheme to encrypt the dataset. We then evaluated the security and performance of our scheme.

Future research includes implementing a prototype of the proposed scheme in a real-world environment, such as on the university campuses of the authors. This will allow us to carry out a more extensive evaluation in a real-world environment, as well as enabling us to evaluate its scalability.

References

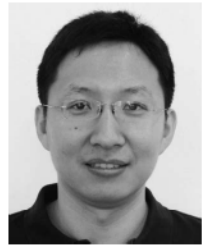
[1] H. P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz, "Probabilistic similarity join on uncertain data," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Singapore, 2006, pp. 295–309.

[2] M. Roopaei, P. Rad, and K.-K. R. Choo, "Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging," *IEEE Cloud Comput.*, vol. 4, no. 1, pp. 10–15, Jan./Feb. 2017.

[3] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch, "Indexing uncertain categorical data," in *Proc. IEEE Int. Conf. Data Eng.*, Istanbul, Turkey, 2007, pp. 616–625.

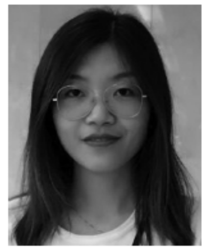
[4] Y. Tao, X. Xiao, and R. Cheng, "Range search on multidimensional uncertain data," *ACM Trans. Database Syst.*, vol. 32, no. 3, p. 15, 2007.

[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, 2000, p. 44.
[6] B. Dan, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, *Public Key Encryption With Keyword Search*. Heidelberg, Germany: Springer, 2004.
[7] C. Guo *et al.*, "Fine-grained database field search using attribute-based encryption for E-healthcare clouds," *J. Med. Syst.*, vol. 40, p. 235, Nov. 2016.
[8] Y. Liu, C. Cheng, T. Gu, T. Jiang, and X. Li, "A lightweight authenticated communication scheme for smart grid," *IEEE Sensors J.*, vol. 16, no. 3, pp. 836–842, Feb. 2016.
[9] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, "A practical privacy-preserving data aggregation (3PDA) scheme for smart grid," *IEEE Trans. Ind. Informat.*, to be published.
[10] Y. Zhang, W. Zhang, Q. Lin, X. Lin, and H. T. Shen, "Effectively indexing the multidimensional uncertain objects," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 608–622, Mar. 2014.
[11] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[12] B. Dan, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith, III, "Public key encryption that allows PIR queries," in *Proc. 27th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, Santa Barbara, CA, USA, 2007, pp. 50–67.
[13] C. Guo *et al.*, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2768045.
[14] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf Appl. Cryptography Netw. Security*, New York, NY, USA, 2005, pp. 442–455.
[15] E.-J. Goh, "Secure indexes," *IACR Cryptol. ePrint Archive*, vol. 2003, p. 216, 2003.
[16] J. Li *et al.*, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2014, pp. 1–5.
[17] C. Guo *et al.*, "Key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage," *Future Gener. Comput. Syst.*, vol. 84, pp. 190–199, Jul. 2018.
[18] D. Barbará, H. Garcia-Molina, and D. Porter, "The management of probabilistic data," *IEEE Trans. Knowl. Data Eng.*, vol. 4, no. 5, pp. 487–502, Oct. 1992.
[19] V. S. Lakshmanan *et al.*, "ProbView: A flexible probabilistic database system," *ACM Trans. Database Syst.*, vol. 22, no. 3, pp. 419–469, 1997.
[20] Y. Zhang, X. Lin, W. Zhang, J. Wang, and Q. Lin, "Effectively indexing the uncertain space," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 9, pp. 1247–1261, Sep. 2010.
[21] F. Angiulli and F. Fassetti, "Indexing uncertain data in general metric spaces," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 9, pp. 1640–1657, Sep. 2012.
[22] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1999, pp. 223–238.
[23] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, 2013, pp. 463–477.
[24] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Adv. Cryptol. (EUROCRYPT)*, Cologne, Germany, Apr. 2009, pp. 224–241.
[25] B. Dan, E. J. Goh, and K. Nissim, *Evaluating 2-DNF Formulas on Ciphertexts*. Heidelberg, Germany: Springer, 2005.
[26] D. Catalano and D. Fiore, "Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Denver, CO, USA, 2015, pp. 1518–1529.
[27] F. Kerschbaum and A. Schroepfer, "Optimal average-complexity ideal-security order-preserving encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 275–286.
[28] J. Graff, *Introduction to Modern Cryptography*. New York, NY, USA: Springer-Verlag, 2000.



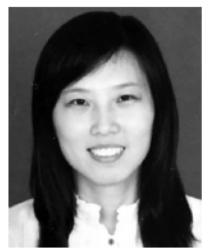
Cheng Guo received the B.S. degree in computer science from the Xi'an University of Architecture and Technology, Xi'an, China, in 2002, and the M.S. degree and Ph.D degree in computer application and technology from the Dalian University of Technology, Dalian, China, in 2006 and 2009, respectively.

From 2010 to 2012, he was a Post-Doctoral Fellow with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. Since 2013, he has been an Associate Professor with the School of Software Technology, Dalian University of Technology. His current research interests include information security, cryptology, and cloud security.



Ruhan Zhuang was born in Benxi, China, in 1994. She received the B.S. degree in software engineering from the Dalian University of Technology, Dalian, China, in 2016, where she is currently pursuing the M.S. degree in software engineering.

Her current research interests include cryptography, private data protection technology, and cloud storage technology.



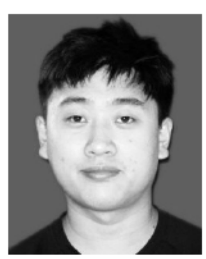
Yingmo Jie received the B.S. degree in information and computing science from the Tianjin University of Technology and Education, Tianjin, China, in 2011, and the M.S. degree in applied mathematics from the Civil Aviation University of China, Tianjin, in 2015. She is currently pursuing the Ph.D. degree at the School of Mathematical Sciences, Dalian University of Technology, Dalian, China.

Her current research interests include information security and game theory.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. degree in information security from the Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio (UTSA), San Antonio, TX, USA, and has a courtesy appointment with the University of South Australia, Adelaide, SA, Australia.

Dr. Choo was a recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the ESORICS 2015 Best Paper Award, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. In 2016, he was named the Cybersecurity Educator of the Year—APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg. He is a Fellow of the Australian Computer Society and an Honorary Commander of the 502nd Air Base Wing, Joint Base San Antonio–Fort Sam Houston.



Xinyu Tang received the B.S. degree in software engineering from the Dalian University of Technology, Dalian, China, in 2016, where he is currently pursuing the M.S. degree in software engineering.

His current research interests include cryptography and cloud security.