SecEQP: A Secure and Efficient Scheme for SkNN Query Problem over Encrypted Geodata on Cloud

Xinyu Lei*, Alex X. Liu*, Rui Li†, Guan-Hua Tu*
* Dept. of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA
† College of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China
Email: {leixinyu, alexliu}@cse.msu.edu, ruili@dgut.edu.cn, ghtu@msu.edu

***Abstract*—Nowadays, location-based services are proliferating and being widely deployed. For example, a Yelp user can obtain a list of the recommended restaurants near his/her current location. For some small or medium location service providers, they may rely on commercial cloud services, e.g., Dropbox, to store the tremendous geospatial data and deal with a number of user queries. However, it is challenging to achieve a secure and efficient location-based query processing over encrypted geospatial data stored on the cloud. In this paper, we propose the Secure and Efficient Query Processing (SecEQP) scheme to address the secure k nearest neighbor (SkNN) query problem. SecEQP employs the projection function-based approach to code neighbor regions of a given location. Given the codes of two locations, the cloud server only needs to compare whether codes equal or not to check the proximity of the two locations. The codes are further embedded into an indistinguishable Bloom filter tree to build a secure and efficient index. The security of SecEQP is formally proved in the random oracle model. We further prototype SecEQP scheme and evaluate its performance on both real-world and synthetic datasets. Our evaluation results show that SecEQP is a highly efficient approach, e.g., top-10 NN query over 1 million datasets only needs less than 40 msec to get queried results.**

I. Introduction

A. Motivations

In location-based services, a user sends his/her current location to a location service provider, and the service provider then responds the user with the query results (such as the top five nearest restaurants). For lower cost, higher performance, and better flexibility, location service providers often host their geospatial data on public clouds. However, in this service model, security and privacy are major concerns as public clouds are typically not fully trusted. The confidential geospatial data and querier location information may be leaked or inferred by the cloud service providers. These storage clouds may have financial incentives (e.g., delivering advertisements to users) to collect or infer their customer sensitive information by analyzing the stored data and user queries. Moreover, these public storage clouds may be compromised and all of the stored information is further leaked by hackers. For example, it is reported that Dropbox is hacked and more than 68 million Dropbox account information is now for sale on the DarkNet marketplace [1].

In this paper, we focus on secure k nearest neighbor (SkNN) query. The location-based kNN search is one of the most widely used location-based services. The state-of-the-art solutions are either not sufficiently efficient or non-strong-provable-secure to perform the location-based kNN searches over the encrypted geospatial data on cloud. Therefore, it is crucial to develop a scheme that provides strong provable security against the untrusted clouds, while still preserving the cloud's ability to efficiently perform location-based kNN queries over the encrypted geospatial data.

B. Problem Formulation

• **Threat Model.** We consider a service model which consists of a data owner, a cloud, and multiple users. The data owner will store the geospatial data on the cloud. The cloud will serve the users' location-based queries. The adversary we consider is the cloud, which is assumed to be honest-but-curious. More specifically, the cloud provides reliable data and query services as the protocol specification, but it is curious about data it stores and queries it receives. Therefore, to protect data privacy and users' location privacy, the data owner needs to encrypt data before outsourcing and the data users need to encrypt the queries before submitting to the cloud.

• **Geospatial Data.** We consider that the data owner stores geospatial data items. Each data item consists of spatial information (e.g., the location of a restaurant) and non-spatial information (e.g., the rating of a restaurant). Data items can be represented and indexed by their spatial information. Formally, they are represented by points $p_1, \cdots, p_n$ in the two-dimensional geographical space.

• **Approximate kNN.** The secure kNN problem is modeled as how the cloud finds the top-k nearest points of $q \in U$ given by a user, as well as provides both the data owner and the user with the security guarantee. It should be ensured that the honest-but-curious cloud cannot deduce any useful information from the data it stores. Meanwhile, when the data user submits its current location to the cloud to launch a kNN query, the honest-but-curious cloud cannot learn the data user's location. In SecEQP, we use the Euclidean distance as the distance metric. To reduce query latency, SecEQP does not aim to discover strict accurate results but acceptable approximate results (e.g., the error is limited to 10%). Note that an approximate answer of kNN with a small error is still very useful in some use scenarios. For example, a user wants to find the top five nearest restaurants within 1 km for lunch. SecEQP may return five nearest restaurants within 1.1 km. The approximate results can still help the user to find a nearby restaurant (s)he likes.

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00065

TABLE I
THE COMPARISON AMONG PREVIOUS SCHEMES AND SECEQP.

Features	Wong et al. [2]	Hu et al. [3]	Yi et al. [4]	Elmehdwi et al. [5]	Wang et al. [6]	Yao et al. [7]	Our SecEQP
Strong security	×	×	×	√	×	√	√
Sublinear query latency	×	√	×	×	√	√	√
Result accuracy	Accurate	Accurate	Accurate	Accurate	Accurate	Accurate	`Approximate`#
kNN or 1NN	k	k	k	k	k	1	k
High-dimensional data	√	√	×	√	√	×	×*
No local index	√	×	√	√	√	×	√
Single server	√	√	√	×	√	√	√
Rounds of interaction	1	$O(\log n)$	1	1	1	2	1

#: SecEQP can achieve high result accuracy by well-developed strategies.
$*$: Handling data with dimensionality more than two is not required for location-based services.

C. Service Model and Design Goals

We propose SecEQP scheme to address the aforementioned secure kNN problem. The service model and design goals of SecEQP scheme are elaborated below.

• **Service Model.** The proposed SecEQP service model is depicted in Figure 1. In SecEQP scheme, data owner delegates the query service to authenticated data users by sharing the secret keys with them. Each Geospatial data item hosted by the data owner consists of location information (spatial attributes) and other information (non-spatial attributes). In order to preserve the ability to query and retrieve the data efficiently, the data owner extracts the spatial attributes of each data item and builds a secure index and then encrypts the entire data items by using the shared keys. Because queries are processed on the secure index, the data items can be encrypted by any encryption algorithms including the standard encryption algorithms with strongest security assurance (e.g., AES). Each secure index item should contain the identifier information (i.e., a pointer) to record the association between the secure index item and the encrypted data item. Afterward, the data owner outsources both the secure index items and the encrypted data items to the powerful cloud, which provides both storage and search services. After the cloud receives the secure index and encrypted data items, the authorized data users can use the shared keys to generate valid search tokens and search for the corresponding SkNN results.

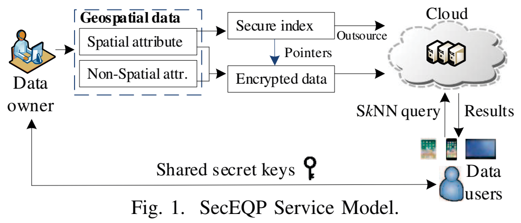


Fig. 1. SecEQP Service Model.

• **Design Goals.** There are three design goals: security, efficiency, and accuracy, which are described in detail as follows.

- *Security.* SecEQP should preserve the following three types of privacy. (1) Data privacy: from the encrypted data items, the adversary cannot reveal any useful information about the data. (2) Index privacy: from the secure index, the adversary cannot learn any useful information about the spatial information of the data items. (3) Token privacy: from the encrypted search token, the adversary cannot infer any information about the query point's location.
- *Efficiency.* SecEQP should satisfy two types of efficiency requirements. (1) Low query latency: the data user can get the result within a reasonable amount of time. (2) Low interaction: the protocol should be non-interactive, or it just requires a small constant number of interactions between the data user and the cloud server.
- *Accuracy.* Let o_i be the ith nearest point returned by SecEQP scheme, and let o_i^* be the ground truth, i.e., the actual ith nearest point. We can compute their distances between the query point q, denoted as $\|q, o_i\|$ and $\|q, o_i^*\|$, respectively. SecEQP should keep that $\|o_i, q\|$ is as close as possible to $\|o_i^*, q\|$ (for all $i = 1, \cdots, k$). A formal definition of accuracy metric can be found in Section V.

D. Comparison with Prior Arts

We compare our proposed SecEQP with other six state-of-the-art SkNN schemes [2]–[7] based on features that a secure kNN scheme is expected to satisfy, such as the support of strong security (i.e., the data privacy and users' location privacy will not be disclosed or inferred), the support of sublinear query processing time (i.e., the query running time is in $O(k \log n)$), etc. The results are summarized in Table I. Among these features, the two important ones are the support of strong security and the support of sublinear query processing time. The major limitation for most of the previous secure kNN schemes is that it is hard to achieve both of them simultaneously. Wang et al. [6] proposed a secure kNN scheme based on order-preserving encryption (OPE) [8], which is a deterministic encryption scheme whose encryption function preserves numerical ordering of the plaintexts. A similar method called distance-recoverable encryption (DRE) is leveraged in [2] and [3] to support secure kNN search. The DRE enables anyone to recover the distance between two points by running a function over their encrypted data. The OPE and DRE are two cases of property-preserving encryptions, which only provide weak privacy protection. They are vulnerable to various serious attacks, as analyzed in [9]. Elmehdwi et al. [5] proposed a novel protocol over encrypted data based on a twin-cloud model [10] and Paillier cryptosystem [11]. This protocol employs too many heavy cryptographic operations, so its query latency is too long, rendering it impractical for large datasets. The private information retrieval (PIR)-based schemes [4] mainly consider how to protect query privacy but not data privacy. Besides, the inefficiency of PIR significantly increases the total search time. Yao et al. [7] designs a solution that can support secure nearest neighbor search by exploiting Voronoi diagram [12] for space partition. Voronoi-based schemes require each data user to download and maintain a copy of the large-size index locally for query processing, which seriously impedes its real-world applications. Besides, the generation of order-k Voronoi diagram for kNN is very computational intensive, as analyzed in [13].

Different from the previous works, SecEQP scheme can support the two most important features (i.e., strong security and sublinear query processing time). However, we would like to point out that SecEQP still has two downsides: (1) returning approximate results to a query instead of accurate ones and (2) only supporting 2-dimensional data, which may not fit all use scenarios (e.g., requiring strict accurate results or 3-dimensional data).

E. Technical Challenges and Proposed Solutions

There are three technical challenges SecEQP shall deal with.

• **C1:** *How to achieve strong data privacy while still supporting efficient kNN query processing?* To measure the proximity of two encrypted points, the straightforward approach is to compute their distance over the encrypted data. A dilemma arises: on one hand, to ensure low query latency requires the data is weakly encrypted (e.g., using order-preserving encryption); on the other hand, if strong encryption (e.g., fully homomorphic encryption (FHE) [14]) is used, the query latency will be prohibitively long. To address the dilemma, we propose the projection-based space encoding method to build a secure index. In SecEQP, the geospatial data can be formally encrypted by standard encryption methods to achieve strong data privacy (e.g., CPA-secure [15]). The secure index enables SecEQP to circumvent heavy computation over encrypted data while still supports secure kNN query processing (Section II).

• **C2:** *How to design a secure index for sublinear query latency while preserving the index privacy?* Building a secure index is not enough for secure kNN query processing. Without any index optimization, the cloud may linearly scan each encrypted data item in the database to evaluate its distance with the queried location. The linear query latency is prohibitively slow for a large dataset (e.g., a million locations are stored in the cloud). To tackle this challenge, we first propose the prefix-free encoding technique to turn the kNN query processing problem to be the keywords query problem. Then, we exploit the indistinguishable Bloom filter (IBF) tree data structure for the secure index building, which can ensure the protocol to be secure and sublinear (Section IV).

• **C3:** *How to develop effective strategies to improve the result accuracy of SecEQP?* SecEQP can only return approximate query results. How to satisfy the high query result accuracy demands is not an easy task. In order to solve this problem, we leverage the observed successive inclusion property to develop an effective strategy to improve the result accuracy (Section V-D).

F. SecEQP Scheme Overview

Figure 2 sketches the SecEQP scheme. We first turn the kNN problem into the equality checking problem via the projection-based space encoding technique (§ II). We further translate the equality checking problem into the keywords query problem by the prefix-free encoding technique (§ IV-A). Finally, we employ the indistinguishable Bloom filter (IBF) tree data structure for index building (§ IV-C) which can ensure our SecEQP to be secure and efficient (i.e., the results to kNN queries are given in sublinear time). The details of provable security are provided in (§ IV-E); we prove that SecEQP is secure in the random oracle model. Moreover, the accuracy of SecEQP results is adaptive and configurable. By leveraging the observed successive inclusion property (§ III), the accuracy of query results can be further improved (§ V-D).

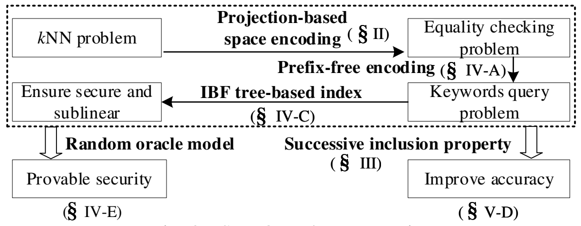

kNN problem
Projection-based space encoding (§ II)
Equality checking problem
Prefix-free encoding (§ IV-A)
Ensure secure and sublinear
IBF tree-based index (§ IV-C)
Keywords query problem
Random oracle model
Successive inclusion property (§ III)
Provable security (§ IV-E)
Improve accuracy (§ V-D)

Fig. 2. SecEQP scheme overview.

G. Main Contributions

This paper makes the following main contributions.

- We leverage a projection-based approach to realize space encoding over two-dimensional space. Space encoding enables the cloud to perform the proximity test between the queried location and the locations in the database by just equality checking operations.
- We design a prefix-free encoding method to embed the codes into an indistinguishable Bloom filter tree to build a secure index. By using the binary search over the indistinguishable Bloom filter tree, our protocol can ensure a sublinear search time.
- We formalize the information leakage functions and formally prove that the SecEQP scheme achieves strong provable security in the random oracle model.

H. Paper Organization

The remainder of this paper proceeds as follows. Section II introduces how to realize space encoding via projection functions. In Section III, we describe how to process kNN queries in plaintext domain. In section IV, we introduce how to transform the designed kNN protocol to be a secure and sublinear protocol. The related analysis is also well presented. Section V conducts the performance evaluation. Section VI overviews the related work. Finally, some conclusions are drawn in Section VII.

II. Space Encoding

In this section, we propose the space encoding technique, which can be used to build a secure index for secure kNN query processing. In the following, we first introduce our customized primitive projection function. Then, we introduce how to encode/stipulate a searching region with infinite space by a single primitive projection function and how to encode/stipulate a searching region with finite space by projection function composition (i.e., multiple primitive projection functions). Moreover, we will introduce how to perform proximity testing between two locations by using the generated codes.

A. Projection Function Introduction

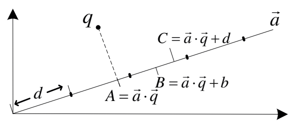
The projection function is defined as follows.

***Definition** 1:* (**Primitive Projection Function**) The primitive projection function $h : \mathbb{R}^2 \to \mathbb{Z}$ maps a two-dimensional vector $\vec{q}$ to an integer,

$$h(\vec{q}) = \lfloor \frac{\vec{a} \cdot \vec{q} + b}{d} \rfloor, \tag{1}$$

where $\vec{a} = (\theta, r)$ denotes a two-dimensional vector in polar coordinate form, where the angle θ is chosen uniformly from the range $[0, 2\pi)$ and the radius $r = 1$. The parameter b is chosen uniformly from the range $[0, d)$.

• **Geometric Interpretation.** The primitive projection function has a simple geometric interpretation. As shown in Figure 3, suppose that $\vec{a}$ crosses the origin and its slope is identical with the straight line in the figure. So the projection of a point q is a point A onto the line $\vec{a}$. By viewing the vector along $\vec{a}$ as a new coordinate axis, A can be represented by its distance from the origin, i.e., $A = \vec{a} \cdot \vec{q}$. The point B is also on the line by shifting A a distance of b. Then, the straight line is divided by discrete intervals of length d. The projected value is the ID of the interval containing B. The farthest bound that B can reach is C, where $C = \vec{a} \cdot \vec{q} + d$, i.e., $B \in [A, C)$ along the line.



q
$\vec{a}$
$C = \vec{a} \cdot \vec{q} + d$
$B = \vec{a} \cdot \vec{q} + b$
$A = \vec{a} \cdot \vec{q}$
d

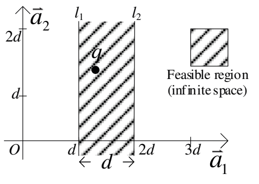
Fig. 3. Geometric illustration of the primitive projection function.

• **Comparison with LSH.** The primitive projection function in Equation (1) has a similar form with locality sensitive hashing (LSH) defined in [16], where the LSH is defined to map a high-dimensional data to an integer. The parameter $\vec{a}$ is a high-dimensional vector with entries chosen independently from a p-stable distribution. The traditional usage of LSH is to reduce the dimensionality of high-dimensional data for accelerating similarity search without security considerations. Different from traditional usage, SecEQP exploits multiple primitive projection functions to project two-dimensional data to high-dimensional data (i.e., the data has a high-dimensional vector representation) for secure kNN search.

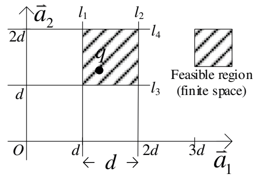
B. Space Encoding via a Single Primitive Projection Function

We now illustrate how to use a single primitive projection function to encode an infinite geometric region.

***Definition** 2:* (**Feasible Region**) Given a point q in the two-dimensional space and a projection function h, we define the feasible region of $\vec{q}$ with respect to the projection function h as consisting of all the possible points $\vec{p}$, such that $h(\vec{q}) = h(\vec{p})$, denoted as $\mathbf{FR}(h(\vec{q}))$.



(a) The feasible region with respect to a single primitive projection function.



(b) The feasible region with respect to two orthogonal projection functions.

Fig. 4. Two examples to illustrate the feasible region.

Figure 4(a) shows an example to illustrate the feasible region with respect to a single primitive projection function. Consider a projection function $h(\vec{q}) = \lfloor \frac{\vec{a_1} \cdot \vec{q} + b}{d} \rfloor$ with $b = 0$. Given a point $q \in \mathbb{R}^2$, suppose that $d \leq \vec{a_1} \cdot \vec{q} < 2d$, so we have $h(\vec{q}) = 1$. As shown in Figure 4(a), for any point in the shadowed area will be projected to be 1. Therefore, the feasible region of q is an infinite region between two parallel lines l_1 and l_2. The distance of l_1 and l_2 is exactly d. In more general cases, the properties of the feasible region are identified by the following Theorem.

***Theorem** 1:* Given a point $q \in \mathbb{R}^2$, the feasible region of q is between two parallel lines l_1 and l_2 that are perpendicular to $\vec{a}$, as shown in Figure 4(a). Define the width of the feasible region wid as the distance of l_1 and l_2, we have $wid = d$, which is independent of the location of q and the choice of b.

The projected code value enables us to test whether two points p, q are in the same infinite d-width space by checking $h(\vec{q}) \overset{?}{=} h(\vec{p})$. For example, as shown in Figure 4(a), if $h(\vec{q}) = h(\vec{p})$, then point q must locate in the feasible region (shadowed area). Note that the proximity testing by using a single projected code is not accurate because the encoded space is infinite. Two far away points may have the same projected code.

C. Projection Function Composition Introduction

We use two kinds of compositions: AND-composition and OR-composition, which are defined as follows.

***Definition** 3:* (**AND-composition** and **OR-composition**)

- AND-composition: Consider there are v projection functions $h_1, \cdots, h_v$. A new composite projection function g can be constructed as the AND-composition of them, denoted as $g = AND(h_1, \cdots, h_v)$. *Equal criterion*: given any two points q and p, $g(\vec{q}) = g(\vec{p})$ if and only if $h_i(\vec{q}) = h_i(\vec{p})$ for all $i \in [v]$, where $[v]$ denotes the set $\{1, \cdots, v\}$.
- OR-composition: Consider there are t projection functions $h_1, \cdots, h_t$. A new composite projection function g can be constructed as the OR-composition of them, denoted as $g = OR(h_1, \cdots, h_t)$. *Equal criterion*: given any two points q and p, $g(\vec{q}) = g(\vec{p})$ if and only if $h_i(\vec{q}) = h_i(\vec{p})$ for at least one $i \in [t]$, where $[t]$ denotes the set $\{1, \cdots, t\}$.

The outputs of a composite projection function can be represented in many ways (e.g., it can be represented by a hierarchical table, as shown in Table II). We give the following example to illustrate how to determine if two composite projection functions equal or not.

• **An Example to Illustrate Equal Criterion.** Table II shows an example to illustrate equal criterion for the composite projection function. Consider there are 4 primitive projection functions $h_{1,1,1}, h_{1,1,2}, h_{1,2,1}, h_{1,2,2}$. Suppose that $g_{1,1}$ is constructed by AND-composition of $h_{1,1,1}, h_{1,1,2}$ denoted as $g_{1,1} = AND(h_{1,1,1}, h_{1,1,2})$. Likewise, suppose that $g_{1,2}$ is AND-composition of $h_{1,2,1}, h_{1,2,2}$ denoted as $g_{1,2} = AND(h_{1,2,1}, h_{1,2,2})$. Let f_1 be constructed by OR-composition of $g_{1,1}, g_{1,2}$, i.e., $f_1 = OR(g_{1,1}, g_{1,2})$. In the example, given two points q and p, since $h_{1,1,1}(\vec{q}) = h_{1,1,1}(\vec{p}), h_{1,1,2}(\vec{q}) = h_{1,1,2}(\vec{p})$, we have $g_{1,1}(\vec{q}) = g_{1,1}(\vec{p})$. Because $h_{1,2,1}(\vec{q}) \neq h_{1,2,1}(\vec{p})$, we have $g_{1,2}(\vec{q}) \neq g_{1,2}(\vec{p})$. Moreover, it holds that $f_1(\vec{q}) = f_1(\vec{p})$, because either $g_{1,1}(\vec{q}) = g_{1,1}(\vec{p})$ or $g_{1,2}(\vec{q}) = g_{1,2}(\vec{p})$ will lead to $f_1(\vec{q}) = f_1(\vec{p})$.

TABLE II
AN EXAMPLE TO ILLUSTRATE EQUAL CRITERION.

point q	$f_1(\vec{q}) = OR(g_{1,1}(\vec{q}), g_{1,2}(\vec{q}))$			
	$g_{1,1}(\vec{q}) = AND(h_{1,1,1}(\vec{q}), h_{1,1,2}(\vec{q}))$		$g_{1,2}(\vec{q}) = AND(h_{1,2,1}(\vec{q}), h_{1,2,2}(\vec{q}))$	
	$h_{1,1,1}(\vec{q}) = 1$	$h_{1,1,2}(\vec{q}) = 2$	$h_{1,2,1}(\vec{q}) = 1$	$h_{1,2,2}(\vec{q}) = 2$
point p	$f_1(\vec{p}) = OR(g_{1,1}(\vec{p}), g_{1,2}(\vec{p}))$			
	$g_{1,1}(\vec{p}) = AND(h_{1,1,1}(\vec{p}), h_{1,1,2}(\vec{p}))$		$g_{1,2}(\vec{p}) = AND(h_{1,2,1}(\vec{p}), h_{1,2,1}(\vec{p}))$	
	$h_{1,1,1}(\vec{p}) = 1$	$h_{1,1,2}(\vec{p}) = 2$	$h_{1,2,1}(\vec{p}) = 3$	$h_{1,2,2}(\vec{p}) = 4$

D. Space Encoding via Projection Function Composition

In this section, we illustrate how to use projection function composition to encode a finite space.

• **Space Encoding by only AND-composition.** Given a point $q \in \mathbb{R}^2$, we now study the feasible region of q with respect to a projection function g, where g is AND-composition of v primitive projection functions. Taking the simplest case $v = 2$ as an example, let

$$g(\vec{q}) = AND(h_1(\vec{q}), h_2(\vec{q})), \tag{2}$$

where $h_1(\vec{q}) = \lfloor \frac{\vec{a}_1 \cdot \vec{q} + b_1}{d} \rfloor$ and $h_2(\vec{q}) = \lfloor \frac{\vec{a}_2 \cdot \vec{q} + b_2}{d} \rfloor$, $b_1 = b_2 = 0$, $\vec{a}_1$ and $\vec{a}_2$ are orthogonal vectors (i.e., $\vec{a}_1 \perp \vec{a}_2$), $d \leq \vec{a}_1 \cdot \vec{q} < 2d$, and $d \leq \vec{a}_1 \cdot \vec{q} < 2d$. As shown in Figure 4(b), the feasible region of q with respect to h_1 is an infinite region between l_1 and l_2. Likewise, the feasible region of q with respect to h_2 is an infinite region between l_3 and l_4. Therefore, the feasible region of q with respect to $g = AND(h_1, h_2)$ is the intersection region (i.e., a d-width square), as shown in Figure 4(b) (shadowed area).

• **Space Encoding by first AND-composition and then OR-composition.** Given a point $q \in \mathbb{R}^2$, where is the feasible region of q with respect to a projection function which is constructed by first AND-composition and then OR-composition? We consider there are two projection functions $g_1(\vec{q})$ and $g_2(\vec{q})$, which are constructed by AND-composition in the same way as Equation (2). Let $f = OR(g_1, g_2)$. Because each of $g_1(\vec{q})$ and $g_2(\vec{q})$ specifies a square feasible region of q as shown in Figure 4(b). Therefore, the feasible region of q with respect to f is the union of two d-width square feasible regions. For instance, Figure 5 shows feasible regions

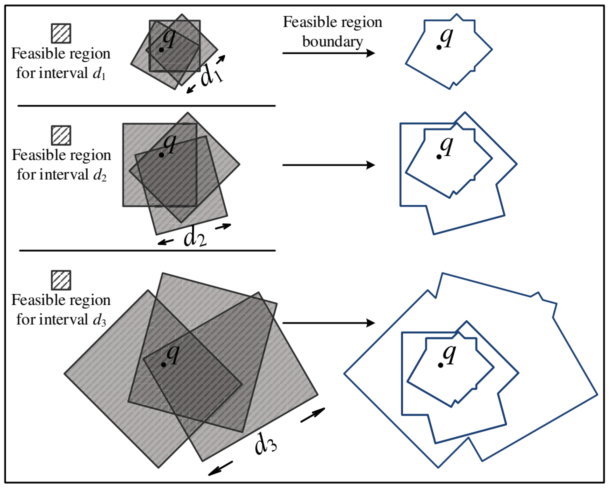

Feasible region for interval d_1
Feasible region boundary
q
d_1
Feasible region for interval d_2
q
d_2
Feasible region for interval d_3
q
d_3

Fig. 5. An example to illustrate successive inclusion property ($\mathbf{FR}(f_1(q)) \subset \mathbf{FR}(f_2(q)) \subset \mathbf{FR}(f_3(q))$).

generated by the union of three square feasible regions with different choices of parameter d.

In analogy to the single primitive projection function-based space encoding, the composite projection function code values enable us to perform proximity testing over a finite two-dimensional space. More concretely, we can test whether a point p is in the feasible region $\mathbf{FR}(f(\vec{q}))$ of q by checking $f(\vec{q}) \overset{?}{=} f(\vec{p})$. The proximity testing over a finite space can get a much more accurate result than proximity testing over an infinite space.

III. kNN PROTOCOL FOR PLAINTEXT DOMAIN

In this section, we describe how to process kNN queries in plaintext domain (i.e., no data encryption is enforced) and then elaborate on how to transform it to secure kNN protocol in Section IV.

Our kNN protocol design is developed on the top of an essential property: successive inclusion property. It can be used to generate a series of gradually enlarged feasible regions for a point. The cloud can search from the smallest feasible region to the largest one and gradually find the k nearest points. In the following, we will introduce the successive inclusion property, present our kNN protocol design, and discuss two critical parameters used in our protocol.

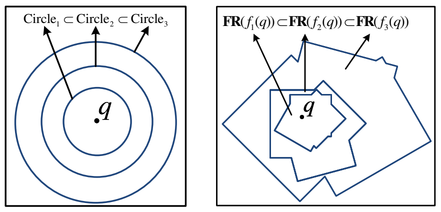
• **Successive Inclusion Property.** We construct three projection functions f_1, f_2, f_3, each of which is constructed by first AND-composition of two primitive projection functions (as in Equation (2)) and then three OR-composition. In the construction of f_1, f_2, and f_3, three parameters d_1, d_2, and d_3 are used to generate the corresponding primitive projection function, respectively. Suppose that $d_1 < d_2 < d_3$, Figure 5 shows an example of the feasible regions of q with respect to f_1, f_2, and f_3, respectively. It is shown in Figure 5 that $\mathbf{FR}(f_1(\vec{q})) \subset \mathbf{FR}(f_2(\vec{q})) \subset \mathbf{FR}(f_3(\vec{q}))$. In general, consider there is a series of composite projection function $f_1, \cdots, f_L$ (with the same first AND-composition and then OR-composite patterns), which are constructed by a series of interval lengths $(d_1, \cdots, d_L)$ (with $d_1 < \cdots < d_L$), respectively. If the gap between two successive values in $d_1, \cdots, d_L$ is sufficiently large, it holds that

$$\mathbf{FR}(f_1(\vec{q})) \subset \mathbf{FR}(f_2(\vec{q})) \subset \cdots \subset \mathbf{FR}(f_L(\vec{q})). \qquad (3)$$

We call the property exhibited in Formula (3) as successive inclusion property.

1) kNN Protocol Design: We next elaborate on our kNN protocol high-level design rationale and present its main algorithms in detail.

• **High-level Design Rationale.** Consider the service model as depicted in Figure 1. The data owner hosts a dataset of n data items in plaintext, then (s)he extracts the spatial attributes, denoted as $p_1, \cdots, p_n$, to build an index for kNN search. The data owner chooses a series of composite projection functions with successive increasing interval lengths $(d_1, \cdots, d_L)$. Given composite projection functions and a data point p_i, the data owner computes a series of feasible regions with successive increasing interval lengths $(d_1, \cdots, d_L)$. Each feasible region is represented by its composite projection function codes, which is outsourced to the cloud to serve as the index. For a query point q, the data user computes a series of the above chosen composite projection functions outputs, and send the codes to cloud for results. Upon reception of the query from the data user, the cloud evaluates the proximity of q and p_i by comparing whether their corresponding composite projection function output codes equal. The cloud searches from the smallest feasible region of q to the largest one until k points are found. Figure 6(b) shows an example of three feasible regions of q that satisfy the successive inclusion property (i.e., $\mathbf{FR}(f_1(\vec{q})) \subset \mathbf{FR}(f_2(\vec{q})) \subset \mathbf{FR}(f_3(\vec{q}))$)). Each feasible region is generated by first two AND-composition and then three OR-composition, as shown in Figure 5. For the query point q, the cloud searches from the smallest feasible region $\mathbf{FR}(f_1(\vec{q}))$ to the largest feasible region $\mathbf{FR}(f_3(\vec{q}))$ to gradually find the closest points.



Circle$_1$ ⊂ Circle$_2$ ⊂ Circle$_3$
q
FR($f_1(q)$)⊂FR($f_2(q)$)⊂FR($f_3(q)$)
q

(a) Circle-based accurate kNN search process. (b) Projection-based approximate kNN search process.

Fig. 6. Comparison between accurate and approximate kNN search process.

• **kNN Protocol in Detail.** The kNN protocol design involves in choosing a group of composite projection functions. We first define the following mathematical notations to facilitate our description. Then, we describe kNN protocol in detail.

Definition 4: (**Projection Function Family**)

- $\mathcal{H}^{d_i}_{1,1}$: We define $\mathcal{H}^{d_i}_{1,1}$ to be the primitive projection function family which contains all of primitive projection functions generated by Equation (1) (with $d = d_i$). Let $h \leftarrow \mathcal{H}^{d_i}_{1,1}$ be the process of randomly sampling a projection function h from $\mathcal{H}^{d_i}_{1,1}$, where the randomness comes from the random choices of the vector $\vec{a}$ and b in Equation (1).
- $\mathcal{H}^{d_i}_{v,1}$: We define $\mathcal{H}^{d_i}_{v,1}$ to be AND-composite projection function family which contains all of composite projection functions generated by the AND-composition of v randomly chosen primitive projection functions $h_1, \cdots, h_v$, where $h_i \leftarrow \mathcal{H}^{d_i}_{1,1}$ for all $i \in [v]$. Let $g \leftarrow \mathcal{H}^{d_i}_{v,1}$ be the process of randomly sampling a composite projection function g from $\mathcal{H}^{d_i}_{v,1}$.
- $\mathcal{H}^{d_i}_{v,t}$: We define $\mathcal{H}^{d_i}_{v,t}$ to be the Or-composite projection function family which contains all of composite projection functions generated by the OR-composition of t randomly chosen AND-composite projection functions $g_1, \cdots, g_t$, where $g_i \leftarrow \mathcal{H}^{d_i}_{v,1}$ for all $i \in [t]$. Let $f \leftarrow \mathcal{H}^{d_i}_{v,t}$ be the process of randomly sampling a composite projection function f from $\mathcal{H}^{d_i}_{v,t}$.

With the above notations, the proposed kNN protocol is described as follows. First, the data owner setups several global parameters including v, t, and L successive increasing interval lengths $(d_1, \cdots d_L)$. Second, the data owner invokes Algorithm 1 (`Index-Building`) to compute and store the projection function output values of each point in the index matrix $\mathbb{I}'$. Afterward, the index $\mathbb{I}'$ is sent to the cloud for storage. Then, the data user calls the Algorithm 2 (`Token-Generation`) to compute and store the projection function output values of the query point in the token array $\mathbb{T}'$. Recall that whether two points are in the same feasible region or not can be deduced by comparing their projection function output values, the cloud calls Algorithm 3 (`Query-Processing`) to check whether $p_i (i \in [n])$ is in the smallest feasible region $\mathbf{FR}(f_1(\vec{q}))$ of query point q via checking $f_1(\vec{q}) \overset{?}{=} f_1(\vec{p_i})$ (i.e., $\mathbb{T}'(1) \overset{?}{=} \mathbb{I}'(i, 1)$) (step 3 in Algorithm 3). According to the successive inclusion property, the cloud searches from the smallest feasible region of q (i.e., $\mathbf{FR}(f_1(\vec{q}))$) to the largest one (i.e., $\mathbf{FR}(f_L(\vec{q}))$) and stops until at least k distinct points are found. Suppose that the search stops when $k'(k \geq k)$ points are found, the data user computes their accurate distance to the query point q and sorts them to figure out the top-k closest points as the query results.

Algorithm 1: `Index-Building`

```
Input: $v$, $t$, $L$, $(d_1, \cdots, d_L)$, $p_1, \cdots, p_n$
Output: $\mathbb{I}'$
1 for ($i = 1; i \leq L; i++$) do
2     $f_i \leftarrow \mathcal{H}^{d_i}_{v,t}$;
3 for ($i = 1; i \leq n; i++$) do
4     for ($j = 1; j \leq L; j++$) do
5         compute $\mathbb{I}'(i, j) = f_j(p_i)$;
          /* $\mathbb{I}'(i, j)$ represents the composite projection
             function output values for data point $p_i$
             with $d = d_j$                                  */
```

2) Analysis of kNN Protocol Parameters: In our kNN protocol, there are two critical parameters: v (the number of AND-composition) and t (the number of OR-composition). We

Algorithm 2: `Token-Generation`

```
Input: $q$ and $f_1, \cdots, f_L$
Output: $\mathbb{T}'$
1 for ($j = 1; j \le L; j++$) do
2     compute $\mathbb{T}'(j) = f_j(q)$;
      /* $\mathbb{T}'(j)$ represents the composite projection
         function output values for query point $q$ with
         $d = d_j$                                         */
```

Algorithm 3: `Query-Processing`

```
Input: $k$, $L$, $\mathbb{I}'$, $\mathbb{T}'$ and $p_1, \cdots, p_n$
Output: $\mathbb{R}'$
/* $\mathbb{R}'$ represents the set of returned points          */
1 Initialization: $\mathbb{R}' = Null$; $i = j = 1$; $result\_num = 0$;
2 while ($result\_num < k$ && $j \le L$) do
3     if (Is_equal($\mathbb{T}'(j), \mathbb{I}'(i,j)$) == True) && ($p_i \notin \mathbb{R}'$) /* search
         for the data point $p_i$ in $\mathbf{FR}(f_j(\vec{q}))$        */
4     then
5         $\mathbb{R}' = \mathbb{R}' \cup p_j$, $result\_num++$;
6     if ($i == n$) /* if $\mathbf{FR}(f_j(\vec{q}))$ have been searched, then
         search in $\mathbf{FR}(f_{j+1}(\vec{q}))$                       */
7     then
8         $j++$, $i = 1$;
9     else
10        $i++$;
          /* search for the next data point $p_{i+1}$ in
             $\mathbf{FR}(f_j(\vec{q}))$                                 */
```

discuss how they influence the performance of kNN protocol as follows.

- **Geometric Analysis of v.** Setting up a larger v implies an improvement of the proximity measurement precision. In order to precisely measure the proximity between two points in two-dimensional space, it is desirable that points p and q are projected to more random directions on the two-dimensional plane and then compare their projection function output values. As a result, a larger v implies an improvement of the proximity measurement precision.
- **Geometric Analysis of t.** Setting up a larger t implies an improvement of the result accuracy. For the query point q, if the cloud searches from a series of concentric circle regions (centered at the point q), then the cloud always gets the accurate results. Figure 6(a) shows an example of the ideal accurate kNN search process. For the query point q, the cloud first searches from the smallest circle $\texttt{Circle}_1$ to the largest circle $\texttt{Circle}_3$ to gradually find the closest points. In comparison, Figure 6(b) shows the projection-based approximate kNN search process. In order to increase the result accuracy, it is desirable that the feasible regions are close to circles with point q at the center. Figure 7 shows that increasing the number of OR-composition t can make the feasible region closer to a circle. Therefore, a larger t implies an improvement of the result accuracy.

• **An Optimization in Projection Function Generation.** In the AND-composite function g generation, SecEQP chooses t random primitive projection functions with t random directions for a point to project. In order to better measure the proximity of two points p, q in the space, it is expected that p, q are projected in many different directions over the space.

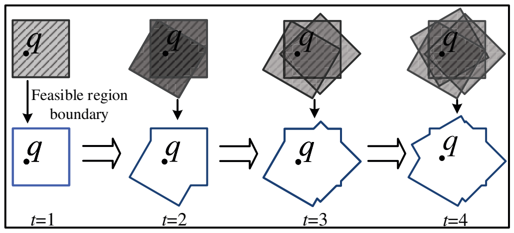


Fig. 7. The feasible region is getting closer to a circle by increasing the number of OR-composition t.

The difference between two directions $\vec{a}_1$ and $\vec{a}_2$ can be represented by their angle, denoted as $\widehat{\vec{a}_1, \vec{a}_2}$. Random projection direction choices may lead to many pairs of similar directions (e.g., $\widehat{\vec{a}_1, \vec{a}_2}$ is very small). To increase the difference between projected directions, we refine our direction choice process by first choosing a random direction $\vec{a}_1 = (\theta, 1)$, and then choose the remaining $v-1$ vectors to equally divide the space. For example, if there are three chosen directions $\vec{a}_1, \vec{a}_2$, and $\vec{a}_3$, an optimized solution is to keep $\widehat{(\vec{a}_1, \vec{a}_2)} = \widehat{(\vec{a}_2, \vec{a}_3)} = \pi/3$.

IV. Transforming kNN to Secure kNN

In this section, we describe how to transform the above kNN protocol to be a secure and sublinear protocol. Our approach is to leverage searchable symmetric encryption (SSE) for keyword query [17]. It allows data users to have secure keyword query processing on the cloud. In order to harness SSE, there are three technical issues need to be addressed.

- *How to generate keywords?* Our solution is to design a prefix-free encoding method (§ IV-A) to encode the projection function output values to generate keywords.
- *How to perform search operations over the index?* Our solution is to do the operation transformation (§ IV-B) which transforms the `Is_equal` evaluation in Algorithm 3 to be `Is_exist` evaluation.
- *How to build a secure index?* Our solution is to use the indistinguishable Bloom filter (IBF) tree based secure index (§ IV-C) which provides the sublinear search time as well as a strong security guarantee.

In the following, we will elaborate on our remedies to the above technical issues in detail and finally present how to apply them to the SkNN protocol (i.e., SecEQP) design (§ IV-D).

A. Prefix-free Encoding

In Algorithm 3 (Query-Processing), for two points p and q, in order to know whether p locates in the feasible region $\mathbf{FR}(f_i(\vec{q}))$ of q, we need to evaluate the logic expression

$$\texttt{Is\_equal}(f_i(\vec{q}), f_i(\vec{p})), \tag{4}$$

where $f_i = OR(g_{i,1}, \cdots, g_{i,t})$. Then, the logic expression (4) can be translated to

$$\texttt{Is\_equal}(g_{i,1}(\vec{q}), g_{i,1}(\vec{p})) \vee \cdots \vee \texttt{Is\_equal}(g_{i,t}(\vec{q}), g_{i,t}(p)). \tag{5}$$

Consider $g_{i,j} = AND(h_{i,j,1}, \cdots, h_{i,j,v})$, we let $str(g_{i,j}(\vec{q})) = h_{i,j,1}(\vec{q})|| \cdots ||h_{i,j,v}(\vec{q}))$, where "||" denotes the string concatenation. In order to circumvent `Is_equal` evaluation, we construct two sets by prefix encoding as

$$\begin{aligned} Q_i &= \{i||1||str(g_{i,1}(\vec{q})), \cdots, i||t||str(g_{i,t}(\vec{q}))\}, \\ P_i &= \{i||1||str(g_{i,1}(\vec{p})), \cdots, i||t||str(g_{i,t}(\vec{p}))\}. \end{aligned} \tag{6}$$

For each component in the coding, we reserve a fix number of bit to ensure that the code is prefix-free. The prefix-free encoding ensures if one element in the set Q_i equals to another element in set P_i, then their each coding component must equal to each other. For example, suppose that $h_{1,1,1}(\vec{q}) = 1, h_{1,1,2}(\vec{q}) = 11, h_{1,1,1}(\vec{p}) = 11$, and $h_{1,1,2}(\vec{p}) = 1$, a direct encoding will lead to $1||1||str(g_{i,j}(\vec{q}))$ = "1" + "1" + "11" + "1" = "11111" and $1||1||str(g_{i,j}(\vec{p}))$ = "1" + "1" + "1" + "11" = "11111". This leads to $str(g_{i,j}(\vec{q})) = str(g_{i,j}(\vec{p}))$ despite $g_{i,j}(\vec{q}) \neq g_{i,j}(\vec{p})$. However, if we fix 2 digits to encode each component, then we have $01||01||str(g_{i,j}(\vec{q}))$ = "01" + "01" + "11" + "01" = "01011101" and $01||01||str(g_{i,j}(\vec{p}))$ = "01" + "01" + "01" + "11" = "01010111", so $str(g_{i,j}(\vec{q})) \neq str(g_{i,j}(\vec{p}))$. Therefore, prefix-free encoding preserves the equal relationship after coding. In the above example, we choose 2 digits for each coding component. However, in real applications, the data owner should choose a number which is not less than the maximum number of digits for each coding component.

B. Operation Transformation

With the prefix-free encoding, the following Theorem holds immediately.

***Theorem** 2:* Logic expression (4) and (5) are True $\Longleftrightarrow$ $Q_i \cap P_i \neq \emptyset$, where "$\Longleftrightarrow$" denotes logical equivalence.

Let us reuse the settings in Table II as an example. According to prefix-free encoding described in Equation (6), we have $Q_1 = \{01010102, 01020102\}$ and $P_1 = \{01010102, 01020304\}$, where we fix 2 bits for each component in coding. Because $Q_1 \cap P_1 = \{01010102\} \neq \emptyset$, we have $f_1(\vec{q}) = f_1(\vec{p})$. Based on Theorem 2, we can employ `Is_exist` evaluation to replace `Is_equal` evaluation. That is, we can know whether query point p is located in the feasible region $\mathbf{FR}(f_i(\vec{q}))$ of q by checking $Q_i \cap P_i \overset{?}{=} \emptyset$. In order to check $Q_i \cap P_i \overset{?}{=} \emptyset$, we can traverse every element in Q_i and then test whether it exists in P_i.

C. Indistinguishable Bloom Filter Tree based Secure Index

The secure index used in SecEQP is built based on a data structure called indistinguishable Bloom filter (IBF) tree. In the following, we will provide the primer of indistinguishable bloom filter and then introduce how to construct an IBF tree for the secure index. Finally, we will discuss why IBF-based index is secure and efficient.

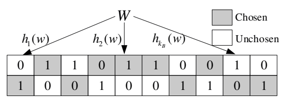
• **Indistinguishable Bloom Filter.** The indistinguishable Bloom filter (IBF) is a data structure that is extended from Bloom filter [18]. It can be used to test whether an element is a member of a set or not. IBF is defined as follows.

***Definition** 5:* (IBF [19]) An IBF is an array B of m twins, k_B different hash functions h_1, h_2, $\cdots$, h_{k_B}, and a random oracle H. Each twin consists of two cells where each cell stores either 0 or 1 and the two cells should be different. The two cells in a twin are named as 0-cell and 1-cell, respectively. For each twin, the oracle H determines which cell is chosen in a random fashion. For every twin, the chosen cell is initialized to 0 and the unchosen cell is set to 1. Given one keyword w, we hash it to k_B twins $B[h_1(w)]$, $B[h_2(w)]$, $\cdots$, $B[h_{k_B}(w)]$, and for each of these k_B twins, we set its chosen cell to 1 and the unchosen cell to 0.

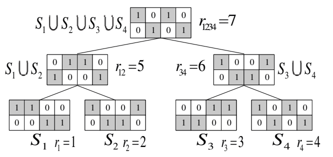
Figure 8(a) shows an example of IBF. Let us describe how to embed a keyword w_i into an IBF. We assume that the data owner and data users share k_B+1 secret keys $K_1, \cdots, K_{k_B+1}$. We construct k_B hash functions using the keyed hash message authentication code (HMAC), where $h_i(\cdot) = \texttt{HMAC}_{K_i}(\cdot) \bmod m$, for $i \in [k_B]$. We construct another hash function as $h_{i+1}(\cdot) = \texttt{HMAC}_{K_B+1}(\cdot)$. The random oracle is instantiated as $H(\cdot) = \texttt{SHA1}(\cdot) \bmod 2$. An IBF can be viewed as a two-dimensional array B with two rows and m columns. Let $B[i][j]$ be the value in the ith row and jth column of the IBF B. To embed a keyword w_i into the IBF B, we set

$$\begin{aligned} B[H(h_{k_B+1}(h_j(w_i)) \oplus r_B)][h_j(w_i)] = 1, \\ B[1 - H(h_{k_B+1}(h_j(w_i)) \oplus r_B)][h_j(w_i)] = 0, \end{aligned} \tag{7}$$

for all $j \in [k_B]$, where r_B is a random number associated with IBF B. To test whether a keyword w_i is in the IBF B, we just need to compute the corresponding hashes and test whether the positions indicated by these hashes are all 1. If all positions are 1, then w_i is in the IBF, otherwise not.



(a) An simplified example of indistinguishable Bloom filter.



(b) An simplified example of indistinguishable Bloom filter tree.

Fig. 8. Indistinguishable Bloom filter and indistinguishable Bloom filter tree examples.

• **Indistinguishable Bloom Filter Tree.** IBFs can be organized into a binary tree structure to achieve sublinear search time. Figure 8(b) shows an example of IBF tree. An IBF tree is constructed as follows. Suppose that B_v is the father IBF of two children IBFs: B_l (left child) and B_r (right child), then B_v is constructed as follows: for each $i \in [m]$, the value of B_v's ith twin is the logical OR of B_l's ith twin and B_r's ith twin. That is

$$\begin{aligned} & B_v[H(h_{k_B+1}(i) \oplus r_{B_v})][i] = \\ & B_l[H(h_{k_B+1}(i) \oplus r_{B_l})][i] \vee B_r[H(h_{k_B+1}(i) \oplus r_{B_r})][i]. \end{aligned} \tag{8}$$

By this way, the IBF tree can be constructed from a number of leaf nodes until there is one root node. As shown in Figure 8(b), if B_l is an IBF representing set S_1 and B_r is an IBF representing set S_2, then we have that B_v is an IBF representing set $S_1 \cup S_2$ while the random numbers, r_1, r_2, and r_{12}, for S_1, S_2, and $S_1 \cup S_2$ are 1, 2, and 5, respectively. More examples and the illustration about how to build a Bloom filter tree can be found in [20], [21]. This property enables us to perform a binary search from the root IBF in the tree to the leaf IBF to test whether a keyword is embedded in a leaf IBF in $O(\log(n))$ time.

The security intuition behind IBF tree-based index is that the positions of 0-cell and 1-cell are determined by the random oracle H, so an IBF tree is a completely random data structure consists of an equal number of 1s and 0s. Moreover, a node-specific random number r_B (see Equation (7)) is adopted while each IBF node is generated. With this design, even if there are two points at the same location, their IBF nodes are likely to be different unless their random numbers are equivalent. This approach thus prevents the cloud from inferring the projected values or the closeness of locations in geospatial database by analyzing their IBF nodes in the IBF tree-based index. Therefore, intuitively, the IBF tree-based index can achieve index privacy (the formal index indistinguishability proof is elaborated in Step 2 of Theorem 3).

D. SkNN Protocol (SecEQP) Design

We next introduce the SecEQP design which employs the aforementioned prefix-free encoding, operation transformation, and IBF-tree based security index techniques.

- *Index-Building.* For each data point in the database, the data owner computes its projection function output values (exactly the same process as described in Algorithm 1). Then, the data owner employs the prefix-encoding (according to the method described in Equation (6)) to generate a set of codes for each point. The set of codes are grouped by a series of sets P_i, for $i \in [L]$. Each point's codes are embedded into a distinct IBF (in the way as described by Equation (7)). All IBFs generated by data points now serve as the leaf nodes to construct a balanced IBF tree (in the way as described by Equation (8)). Each IBF node in the IBF tree is associated with a random number as shown in Equation (7) and Equation (8). The IBF tree along with the random number for each IBF node in the tree serve as the secure index, which is outsourced and stored in the cloud.
- *Token-Generation.* Given the query point q, the data user computes projection function values and employs the prefix-encoding to generate a set of codes for each point (according to the method described in Equation (6)). The set of codes are grouped by a series of sets Q_i, for $i \in [L]$. The set of codes serve as a series of keywords. Note that the keywords in Q_{i_1} are put before keywords in Q_{i_2} if $i_1 < i_2$. For a keyword w_i, the data user computes k_B locations $h_j(w_i)$, for $j \in [k_B]$. For each location $h_j(w_i)$, the data user computes hash $h_{K_B+1}(h_j(w_i))$. The search token t_{w_i} of keyword w_i is a k_B-pair of hashes and locations: $\{h_{K_B+1}(h_j(w_i), h_j(w_i)\}$, for $j \in [k_B]$. The data user generates search tokens in the above way for all keywords in Q_i ($i \in [L]$) and sends these search tokens to the cloud for results. Because these hash functions are one-way, it is hard for the cloud to deduce the useful information of the query point by viewing these search tokens.
- *Query-processing.* On receipt of a search token t_{w_i} for keyword w_i from the data user, the cloud performs the query processing, which is described as follows. Let $t_{w_i}[j]$ denote the jth ordered pair in t_{w_i}, i.e., $t_{w_i}[j] = \{h_{K_B+1}(h_j(w_i), h_j(w_i))\}$. Let $t_{w_i}[j].f$ and $t_{w_i}[j].s$ be the first and second hash in $t_{w_i}[j]$, respectively. For an IBF B that the cloud checks against t_{w_i}, if there exist $j \in [k_B]$ such that $B[H(t_{w_i}[j].f \oplus r_B)][t_{w_i}[j].s] = 0$, then t_{w_i} does not match any of the items embedded in the IBF. If the cloud determines that t_{w_i} does not match the IBF B, the query processing terminates. Otherwise, the cloud processes t_{w_i} against the left and right children of the IBF B. The search begins from the root IBF until the cloud reaches the leaf IBF and get the corresponding encrypted data item. The cloud searches from the first keyword-generated tokens to the last keyword-generated tokens and stops until at least k distinct IBF leaf nodes are found. Last, the cloud returns the corresponding encrypted data item to the data user for further processing.

E. Security Analysis

In this section, we first describe the adopted security model and related notations. Then, we define leakage functions and perform security proof for SecEQP.

• *Secure Model and Notations.* We adopt the widely used adaptive indistinguishability under chosen-keyword attack (IND-CKA) secure model [17]. Let $\mathbf{D} = \{d_1, \cdots, d_n\}$ denote the set of data items. Let $\mathbb{I}$ and $\mathbf{T}$ denote the index and search token, respectively. Suppose that SecEQP employes a CPA-secure encryption scheme [15] to encrypt each data items.

• *Leakage Functions.* Before we carry out the formal security proof, we introduce two leakage functions. (1) $\mathcal{L}_1(\mathbb{I}, \mathbf{D})$: Given the index $\mathbb{I}$ and dataset $\mathbf{D}$, this function outputs the size of each IBF m, the number of data items n in $\mathbf{D}$, the data item identifiers $ID = (id_1, \cdots, id_n)$, and the size of each encrypted data item. (2) $\mathcal{L}_2(\mathbb{I}, \mathbf{D}, q_i)$: This function takes as input the index $\mathbb{I}$, the set of data items $\mathbf{D}$, and a query q_i. It outputs two types of information: the *search pattern*, which is the information about whether the same search was performed before or not, and the *access pattern*, which is the information about which data item identifiers that match query q_i.

***Theorem** 3:* SecEQP scheme is adaptive IND-CKA $(\mathcal{L}_1, \mathcal{L}_2)$-secure in the random oracle model.

Proof: In the proof, we first describe a simulator $\mathcal{S}$ that can simulate a view $A_v^* = (\mathbb{I}^*, \mathbf{T}^*, \mathbf{c}^*)$ with the help of information accessible in the leakage functions $\mathcal{L}_1$ and $\mathcal{L}_2$. Next, we show that a probabilistic polynomial-time (PPT) adversary cannot distinguish between the simulated view $A_v^* = (\mathbb{I}^*, \mathbf{T}^*, \mathbf{c}^*)$ and the real adversary view $A_v = (\mathbb{I}, \mathbf{T}, \mathbf{c})$.

• Step (1): Simulate $\mathbf{c}^*$ (which captures the requirement for data privacy). To simulate the encrypted data items $D = \{d_1, \cdots, d_n\}$, the simulator first learns the value n and the size of each encrypted data item from the leakage function $\mathcal{L}_1$. Then, the simulator generates the simulated ciphertext with randomly selected plaintext and the known CPA-secure encryption algorithm. The simulator needs to ensure that the simulated ciphertext has the same size as the real ciphertext.

Because the CPA-secure encryption algorithm achieves ciphertext indistinguishability, a PPT adversary cannot distinguish the simulated ciphertext with the real ciphertext.

• Step (2): Simulate $\mathbb{I}^*$ (which captures the requirement for index privacy). To simulate the IBF tree T, the simulator $\mathcal{S}$ constructs an identically structured IBF tree first. Then, for each node v in T, the simulator $\mathcal{S}$ sets up an IBF B_v with the same size as in the IBF in the index $\mathbb{I}$. Note that the simulator $\mathcal{S}$ can learn the IBF size from the leakage function $\mathcal{L}_1$. In the ith twin of B_v, the simulator $\mathcal{S}$ stores either 0 at $B_v[0][i]$ and 1 at $B_v[1][i]$, or vice versa. For each twin, how to assign 0-cell and 1-cell is decided by fairly tossing a coin. Next, for each IBF node, the simulator $\mathcal{S}$ generates a random number to associate with it. Finally, the simulator $\mathcal{S}$ outputs the IBF tree T and its associated random number as the simulated index $\mathbb{I}^*$ to the adversary. The simulated index $\mathbb{I}^*$ has exactly the same structure with the real index $\mathbb{I}$. The IBF nodes in either $\mathbb{I}^*$ or $\mathbb{I}$ have the same size and equally distributed 0-cell and 1-cell. Hence, a PPT adversary cannot distinguish between the simulated index $\mathbb{I}^*$ and the real index $\mathbb{I}$.

• Step (3): Simulate $\mathbf{T}^*$ (which captures the requirement for token privacy). Suppose that the simulator $\mathcal{S}$ receives a query q_i. From the leakage function $\mathcal{L}_2$, the simulator $\mathcal{S}$ knows whether this query has been searched before or not. If it has been searched before, the simulator $\mathcal{S}$ outputs the previous searched token t_{q_i} to the adversary. Otherwise, the simulator $\mathcal{S}$ generates a new search token t_{q_i} as follows. The search token for a query is the set of k_B-pair of hashes and locations. Because the simulator $\mathcal{S}$ can learn access pattern from the leakage function $\mathcal{L}_2$, the simulator $\mathcal{S}$ knows which leaf IBF node in the index matches the search token t_{q_i}. For the leaf IBF node that matches the search token t_{q_i}, the simulator $\mathcal{S}$ can program the bit output by the random oracle $H(\cdot)$ to select k_B-pair of hashes and locations and ensure that the selected k_B-pair of hashes and locations match the leaf IBF node v. For the leaf IBF node that does not match the search token t_{q_i}, the simulator $\mathcal{S}$ is able to ensure that the simulated search token does not match the IBF node by programming the bit output by the random oracle. By this way, the simulator $\mathcal{S}$ can output the generated k_B-pair of hashes and locations as the simulated search token $\mathbf{T}^*$. Since the search token is k_B-pair of hashes and locations which are produced by the random hash functions, the simulated search token $\mathbf{T}^*$ is indistinguishable from the real search token $\mathbf{T}$ by a PPT adversary.

In summary, the simulated view $A_v^* = (\mathbb{I}^*, \mathbf{T}^*, \mathbf{c}^*)$ and the real view $A_v = (\mathbb{I}, \mathbf{T}, \mathbf{c})$ are indistinguishable by a PPT adversary. Therefore, SecEQP scheme is adaptive IND-CKA $(\mathcal{L}_1, \mathcal{L}_2)$-secure in the random oracle model. ■

V. Performance Evaluation

In this section, we first introduce parameter settings, datasets, performance metrics, and implementation. Then, we evaluate the performance of SecEQP and compare it with other two schemes (Elmehdwi et al. [5] and Yao et al. [7]) with the strong security assurance. Last, we describe a strategy to improve the result accuracy to meet a variety of location service demands.

A. Parameters Settings

Table IV summarizes the default parameter settings in the experiment. Among these parameters, the choices of $(d_1, \cdots, d_L)$ are not straightforward, because they affect the size of the feasible region. If they are set to be too small, too few or even no points are inside the feasible region. if they are set to be too large, then too many points are inside the feasible region, this would make the post-processing inefficient. Accordingly, we design a parameter training algorithm (run by the data owner) for choosing appropriate $(d_1, \cdots, d_L)$. The design rationale of the parameters training algorithm is that it uses the knowledge of the dataset to adjust $(d_1, \cdots, d_L)$ to be appropriate values to ensure that an appropriate number of points can be returned in the search. We skip the details of the parameters training algorithm due to space limitations.

B. Datasets, Metrics, and Implementation

• **Datasets.** (1) **NY** is a real-world dataset contains 1 million spatial data in the state of New York (NY) from OpenStreetMap Project [22], which collects geographical data from volunteered mobile device carriers. (2) **CA** is a real-world dataset contains 1 million spatial data in California (CA) from OpenStreetMap Project. (3) **UF** is a synthetic dataset contains 1 million spatial data generated from uniform (UF) distribution. More specifically, each data is denoted as (X_{UF}, Y_{UF}), where $X_{UF} \sim U[0, 10^9]$ and $Y_{UF} \sim U[0, 10^9]$.

• **Metrics.** Three metrics are used: (1) query latency, (2) query result accuracy, and (3) query cost. The query latency is defined as the time for the cloud to respond to an SkNN query. The result accuracy of an SkNN query can be reflected by *Overall Approximation Ratio* (OAR) [23], which is defined as $\frac{1}{k}\sum_{i=1}^{k}\frac{\|o_i,q\|}{\|o_i^*,q\|}$, where q is the query point, o_i is the ith nearest point in the search results and o_i^* is the ground truth (i.e., the actual ith nearest point in the dataset). Theoretically, the high result accuracy means OAR should be close to 1. The query cost consists of the communication cost and the size of the secure index maintained in the cloud.

• **Implementation.** The SecEQP implementations are achieved by C++. We carry out the experiments on a cluster node (serves as the cloud) equipped with 128 GB RAM and two 2.5Ghz 10-core Intel Xeon E5-2670v2 CPU. In our experiments, unless otherwise stated, when we vary the value of one parameter in concern, we keep all other parameters at their default values, which are displayed in Table IV.

C. Experiment Results

We first evaluate the performance of SecEQP in terms of query latency, result accuracy, and query cost. Then we compare SecEQP with other two schemes (Elmehdwi et al. [5] and Yao et al. [7]) with the strong security assurance.

• **Query Latency.** The query latency as a function of n (i.e., the number of points in a dataset) and k (i.e., the number of nearest points required in a query) is shown in Figure 9 and

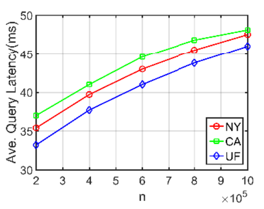


Fig. 9. SecEQP query latency by varying the parameter n.

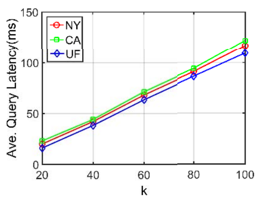


Fig. 10. SecEQP query latency by varying the parameter k.

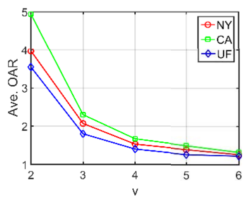


Fig. 11. SecEQP OAR by varying the parameter v.

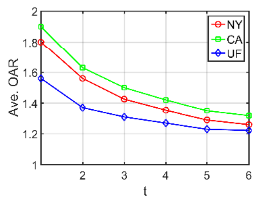


Fig. 12. SecEQP OAR by varying the parameter t.

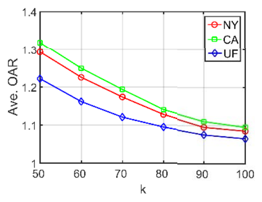


Fig. 13. SecEQP OAR by varying the parameter k.

TABLE III
COMPARE SECEQP WITH OTHER SCHEMES (NA: NOT APPLICABLE).

Scheme	**Query latency**						**Query cost**				**Accuracy (OAR)**
	$k=1$			$k=50$			Communication volume	Index size			
	$n=10^4$	$n=10^5$	$n=10^6$	$n=10^4$	$n=10^5$	$n=10^6$	(token size + data items size)	$n=10^4$	$n=10^5$	$n=10^6$	
SecEQP	9 ms	21 ms	31 ms	12 ms	32 ms	47 ms	6.93 KB + data item size	0.44 GB	3.13 GB	15.8 GB	≈1.3
Yao et al. [7]	5 ms	9 ms	12 ms	Na	Na	Na	8 byte + data items size	14.8 MB	17.4 MB	20.3 MB	1
Elmehdwi et al. [5]	0.15 sec	1.44 sec	14.3 sec	0.12 min	1.18 min	11.78 min	16 byte + data items size	Na	Na	Na	1

TABLE IV
PARAMETER SETTINGS.

Notations	Meanings	Default Values
n	the number of points	100,000
k	the number of nearest points required in a query	50
m	the number of twins in the root node of IBF tree	$10Ltn$
k_B	the number of hash functions in an IBF	7
v	the number of AND-composition	6
t	the number of OR-composition	6
L	the number of interval lengths	5

Figure 10. It can be observed that the query latency grows sublinear with n and a slightly faster than linear with k. While $k = 50$, the query latency for a dataset contains 1 million points with is less than 50 msec.

• **Result Accuracy.** Let n' be the total number of inserted items in the root node of the IBF. Figure 11 and 12 exhibit OAR as a function of v and t, respectively. The OAR is monotonically decreasing with increasing v and t. Hence, increasing v and t can improve the result accuracy. As shown in Figure 11, if $v = 6$ and $t = 6$, the average OAR is about 1.3. This means that the average distance between the queried point and returned results is 1.3 times longer than the ground truth. Note that a strategy is developed to further improve the result accuracy (i.e., OAR can be improved to be 1.1) in Section V-D.

• **Query Cost.** For the query cost, we consider the communication volume and the size of the secure index which helps to accelerate the query processing in the cloud. The communication volume consists of the transmission of the search token and the encrypted data items. Since the size of encrypted data items is independent of the adopted SecEQP scheme, we only consider the communication volume of the search token. Table III shows the token size in an SkNN query for different schemes. The token size of SecEQP is computed based on the parameter settings in Table IV. It is shown that all of the schemes have a constant token size for an SkNN query. The search token only takes 6.93 KB, indicating a very small communication volume. For index size, SecEQP employs the existing IBF tree compression algorithms [19] to compress the index size. The index size varies with dataset sizes. While the dataset contains 10^4, 10^5, and 10^6 points, the index will take 0.44 GB, 3.13 GB, and 15.8 GB, respectively.

• **Comparison with other schemes.** We compare SecEQP with two schemes with strong security assurance. The experiment results for query latency are average over the three datasets. The results are summarized in Table III. We have five observations.

First, while $k = 1$ (i.e., 1NN), Yao et al. [7] has the shortest query latency over three database sizes (10^4, 10^5, 10^6), ranges from 5 ms to 12 ms, whereas SecEQP has a comparable query latency (i.e., from 9 ms to 31 ms). Second, while $k = 50$, SecEQP has the shortest query latency, ranges from 12 ms to 47 ms, which is not significantly increased with k, whereas Elmehdwi et al. [5] do (e.g., 14.3 sec$\rightarrow$11.78 min). Note that Yao et al. [7] does not support the use scenarios while $k > 1$. Third, the average OAR for both Yao et al. [7] and Elmehdwi et al. [5] is 1, whereas SecEQP is about 1.3. Forth, all of three schemes do not create large tokens. The token size ranges from 8 bytes (Yao et al. [7]) to 6.9 KB (SecEQP). Fifth, SecEQP's index is the largest one compared with other schemes. In a dataset contains 10^6 points, SecEQP's index takes 15.8 GB, whereas Yao et al. [7] and Elmehdwi et al. [5] use 20.3 MB and 0 MB, respectively. There are two causes. First, Elmehdwi et al. [5] does not employ the index mechanism for the query acceleration. Second, SecEQP supports the use scenarios while $k > 1$; however, Yao et al. [7] does not.

D. Improve Result Accuracy

In this section, we first illustrate the top nearest accuracy property and then describe how to use it to develop an effective strategy to improve the result accuracy.

• **Top Nearest Accuracy Property.** It is observed in the experiments that the closer the point, the less probability it is missed in the searching. We call this top nearest accuracy property. Theoretically, this property is caused by the successive inclusion property as exhibited in Formula (3). The following example is helpful for understanding. Suppose that the query processing stops after searching $\mathbf{FR}(f_5(q))$. For the points in $\mathbf{FR}(f_1(q))$, they are searched for 5 times; ... ; for the points in $\mathbf{FR}(f_5(q))$, they are searched for only 1 time. This repeated filtering process leads to top nearest accuracy property.

• **Improve Accuracy Strategy.** The top nearest accuracy property indicates a strategy to improve the result accuracy of SecEQP scheme. The strategy is that if we want to get kNN, we can query k'NN, where $k' > k$, and figure out the top-k nearest points as the query results. To evaluate this strategy, we conduct an experiment as follows. We query kNN, where $k = 50, \cdots, 100$, and select the top 50 nearest points as the final query results for 50NN. The experiment results are plotted in Figure 13. We observe that the result accuracy is improved by increasing k. If we let $k = 100$, the average OAR is less than 1.1, which demonstrates that this strategy is effective in improving the result accuracy.

VI. Related Work

• Location Obfuscation Approach. Schemes based on location obfuscation [24], and data transformation [2], [25] do not use strong standard encryption algorithm. Therefore, they suffer from weak privacy.

• Private Information Retrieval Approach. The Private Information Retrieval (PIR)-based solutions [4] mainly consider protecting query privacy but not data privacy. Besides, PIR-based solutions suffer from long query latency for large-scale dataset.

• Fully Homomorphic Encryption Approach. Fully homomorphic encryption (FHE) [14] enables cloud to perform kNN computation directly over the encrypted data. However, current FHE solutions still lack efficiency.

• Property-preserving Encryption Approach. Distance-recoverable encryption (DRE)-based schemes [2], [3] and Order-preserving encryption (OPE)-based SkNN schemes [2], [6] achieve weak security, as analyzed in [9].

• Voronoi Diagram Approach. Voronoi-based scheme [7] requires each data user to download and maintain a copy of the large-size index locally for query processing, which seriously impedes its real-world applications.

VII. Conclusions

In this paper, we proposed a novel SecEQP scheme which supports practical SkNN query processing over encrypted geospatial data in cloud computing. The key novelty of our scheme is in applying projection function composition to encode two-dimensional data to test the proximity of two points by only equality checking operations. We formulated the related theory and explained it via various illustrative graphic examples. We implemented and evaluated SecEQP scheme on both real-world and synthetic datasets. It is shown that SecEQP scheme can achieve strong security, high-efficiency, and high result accuracy. We hope that our study will invite more research in the current era of big data meeting security.

Acknowledgement

This work is partially supported by the National Science Foundation under Grant Numbers CNS-1837146, CNS-1815636, and CNS-1814551. Note that any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors only and do not necessarily reflect those of the National Science Foundation. Moreover, this work is also partially supported by the National Natural Science Foundation of China under Grant Numbers 61872082 and 61672156.

References

[1] "Dropbox hack," http://www.troyhunt.com/the-dropbox-hack-is-real/.
[2] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *SIGMOD*, 2009, pp. 139–152.
[3] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *ICDE*, 2011, pp. 601–612.
[4] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical k nearest neighbor queries with location privacy," in *ICDE*, 2014, pp. 640–651.
[5] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *ICDE*, 2014, pp. 664–675.
[6] B. Wang, Y. Hou, and M. Li, "Practical and secure nearest neighbor search on encrypted large-scale data," in *INFOCOM*, 2016, pp. 1–9.
[7] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *ICDE*, 2013, pp. 733–744.
[8] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *SIGMOD*, 2004, pp. 563–574.
[9] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *CCS*, 2015, pp. 644–655.
[10] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider, "Twin clouds: An architecture for secure cloud computing," in *WCSC*, 2011.
[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *ICTACT*, 1999.
[12] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 2009.
[13] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino, "Secure knn query processing in untrusted cloud environments," *IKDE*, pp. 2818–2831, 2014.
[14] C. Gentry, "Fully homomorphic encryption using ideal lattices." in *STOC*, 2009, pp. 169–178.
[15] J. Katz and Y. Lindell, "Introduction to modern cryptography: principles and protocols. cryptography and network security," 2008.
[16] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *SCG*, 2004, pp. 253–262.
[17] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *CCS*, 2006, pp. 79–88.
[18] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, 1970.
[19] R. Li and A. X. Liu, "Adaptively secure conjunctive query processing over encrypted data for cloud computing," in *ICDE*, 2017, pp. 697–708.
[20] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, p. 216, 2003.
[21] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," *PVLDB*, pp. 1953–1964, 2014.
[22] "Openstreetmap," http://www.openstreetmap.org/.
[23] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and efficiency in high dimensional nearest neighbor search," in *SIGMOD*, 2009, pp. 563–576.
[24] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: query processing for location services without compromising privacy," in *VLDB*, 2006, pp. 763–774.
[25] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *SSTD*, 2007, pp. 239–257.