Secure kNN Query Processing in Untrusted Cloud Environments

Sunoh Choi, Gabriel Ghinita, Hyo-Sang Lim, and Elisa Bertino, *Fellow, IEEE*

Abstract—Mobile devices with geo-positioning capabilities (e.g., GPS) enable users to access information that is relevant to their present location. Users are interested in querying about points of interest (POI) in their physical proximity, such as restaurants, cafes, ongoing events, etc. Entities specialized in various areas of interest (e.g., certain niche directions in arts, entertainment, travel) gather large amounts of geo-tagged data that appeal to subscribed users. Such data may be sensitive due to their contents. Furthermore, keeping such information up-to-date and relevant to the users is not an easy task, so the *owners* of such data sets will make the data accessible only to paying customers. Users send their current location as the query parameter, and wish to receive as result the nearest POIs, i.e., nearest-neighbors (NNs). But typical data owners do not have the technical means to support processing queries on a large scale, so they outsource data storage and querying to a cloud *service provider*. Many such cloud providers exist who offer powerful storage and computational infrastructures at low cost. However, cloud providers are not *fully* trusted, and typically behave in an *honest-but-curious* fashion. Specifically, they follow the protocol to answer queries correctly, but they also collect the locations of the POIs and the subscribers for other purposes. Leakage of POI locations can lead to privacy breaches as well as financial losses to the data owners, for whom the POI data set is an important source of revenue. Disclosure of user locations leads to privacy violations and may deter subscribers from using the service altogether. In this paper, we propose a family of techniques that allow processing of NN queries in an untrusted outsourced environment, while at the same time protecting *both* the POI and querying users' positions. Our techniques rely on *mutable order preserving encoding (mOPE)*, the only secure order-preserving encryption method known to-date. We also provide performance optimizations to decrease the computational cost inherent to processing on encrypted data, and we consider the case of incrementally updating data sets. We present an extensive performance evaluation of our techniques to illustrate their viability in practice.

Index Terms—Location privacy, spatial databases, database outsourcing, mutable order preserving encoding

1 Introduction

The emergence of mobile devices with fast Internet connectivity and geo-positioning capabilities has led to a revolution in customized *location-based services (LBS)*, where users are enabled to access information about *points of interest (POI)* that are relevant to their interests and are also close to their geographical coordinates. Probably the most important type of queries that involve location attributes is represented by *nearest-neighbor (NN)* queries, where a user wants to retrieve the k POIs (e.g., restaurants, museums, gas stations) that are nearest to the user's current location (kNN).

A vast amount of research focused on performing such queries efficiently, typically using some sort of spatial indexing to reduce the computational overhead [1]. The issue of privacy for users' locations has also gained significant attention in the past. Note that, in order for the NNs to be determined, users need to send their coordinates to the LBS. However, users may be reluctant to disclose their coordinates if the LBS may collect user location traces and use them for other purposes, such as profiling, unsolicited advertisements, etc. To address the user privacy needs, several protocols have been proposed that withhold, either partially or completely, the users' location information from the LBS. For instance, the work in [16], [17], [18], [19] replaces locations with larger cloaking regions that are meant to prevent disclosure of exact user whereabouts. Nevertheless, the LBS can still derive sensitive information from the cloaked regions, so another line of research that uses cryptographic-strength protection was started in [7] and continued in [8], [9]. The main idea is to extend existing private information retrieval (PIR) protocols for binary sets to the spatial domain, and to allow the LBS to return the NN to users without learning any information about users' locations. This method serves its purpose well, but it assumes that the actual data points (i.e., the points of interest) are available in plaintext to the LBS. This model is only suitable for general-interest applications such as GoogleMaps, where the landmarks on the map represent public information, but cannot handle scenarios where the data points must be protected from the LBS itself.

More recently, a new model for data sharing emerged, where various entities generate or collect data sets of POI that cover certain niche areas of interest, such as specific segments of arts, entertainment, travel, etc. For instance, there are social media channels that focus on specific travel habits, e.g., eco-tourism, experimental theater productions

- *S. Choi and E. Bertino are with the Department of Computer Science, Purdue University, West Lafayette, IN 47907. E-mail: {choi39, bertino}@purdue.edu.*
- *G. Ghinita is with the Department of Computer Science, University of Massachusetts Boston, 100 William T. Morrissey Boulevard, Boston, MA 02125. E-mail: Gabriel.Ghinita@umb.edu.*
- *H.-S. Lim is with the Department of Computer and Telecommunications Engineering, Yonsei University, Wonju, Korea. E-mail: hyosang@yonsei.ac.kr.*

Manuscript received 1 June 2013; revised 11 Nov. 2013; accepted 23 Dec. 2013. Date of publication 23 Jan. 2014; date of current version 26 Sept. 2014. Recommended for acceptance by E. Pitoura. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2014.2302434

or underground music genres. The content generated is often geo-tagged, for instance related to upcoming artistic events, shows, travel destinations, etc. However, the owners of such databases are likely to be small organizations, or even individuals, and not have the ability to host their own query processing services. This category of data owners can benefit greatly from outsourcing their search services to a cloud service provider. In addition, such services could also be offered as plug-in components within social media engines operated by large industry players.

Due to the specificity of such data, collecting and maintaining such information is an expensive process, and furthermore, some of the data may be sensitive in nature. For instance, certain activist groups may not want to release their events to the general public, due to concerns that big corporations or oppressive governments may intervene and compromise their activities. Similarly, some groups may prefer to keep their geo-tagged data sets confidential, and only accessible to trusted subscribed users, for the fear of backlash from more conservative population groups. It is therefore important to protect the data from the cloud service provider. In addition, due to financial considerations on behalf of the data owner, subscribing users will be billed for the service based on a *pay-per-result* model. For instance, a subscriber who asks for kNN results will pay for k items, and should not receive more than k results. Hence, approximate querying methods with low precision, such as existing techniques [5] that return many false positives in addition to the actual results, are not desirable.

Such scenarios call for a novel and challenging category of services that provide secure kNN processing in outsourced environments. Specifically, both the POI and the user locations must be protected from the cloud provider. This model has been formulated previously in literature as “blind queries on confidential data” [18]. In this context, POIs must be encrypted by the data owner, and the cloud service provider must perform NN processing on encrypted data.

This is a very challenging task, as conventional encryption does not support processing on top of ciphertexts, whereas more recent cryptographic tools such as homomorphic encryption are not flexible enough (they support only restricted operations), and they are also prohibitively expensive for practical uses. To address this problem, previous work such as [2] has proposed privacy-preserving data transformations that hide the data while still allowing the ability to perform some geometric functions evaluation. However, such transformations lack the formal security guarantees of encryption. Other methods employ stronger-security transformations, which are used in conjunction with data set partitioning techniques [5], but return a large number of false positives, which is not desirable due to the financial considerations outlined earlier.

In this paper, we propose a family of techniques that allow processing of NN queries in an untrusted outsourced environment, while at the same time protecting both the POI and querying users’ positions. Our techniques rely on *mutable order preserving encoding (mOPE)* [6], which guarantees *indistinguishability under ordered chosen-plaintext attack* (IND-OCPA) [11], [12]. We also provide performance optimizations to decrease the computational cost inherent to processing on encrypted data, and we consider the case of incrementally updating data sets.

Inspired by previous work in [7], [9] that brought together encryption and geometric data structures that enable efficient NN query processing, we investigate the use of Voronoi diagrams and Delaunay triangulations [1] to solve the problem of secure outsourced kNN queries. We emphasize that previous work assumed that the contents of the Voronoi diagrams [7], [9] is available to the cloud provider in plaintext, whereas in our case the processing is performed entirely on ciphertexts, which is a far more challenging problem.

Our specific contributions are:

i. We propose the VD-kNN method for secure NN queries which works by processing encrypted Voronoi diagrams. The method returns exact results, but it is expensive for $k > 1$, and may impose a heavy load on the data owner.

ii. To address the limitations of VD-kNN, we introduce TkNN, a method that works by processing encrypted Delaunay triangulations, supports any value of k and decreases the load at the data owner. TkNN provides exact query results for $k = 1$, but when $k > 1$ the results it returns are only approximate. However, we show that in practice the accuracy is high.

iii. We outline a mechanism for updating encrypted Voronoi diagrams and Delaunay triangulations that allows us to deal efficiently, in an incremental manner, with changing data sets.

iv. We propose performance optimizations based on spatial indexing and parallel computation to decrease the computational overhead of the proposed techniques.

v. Finally, we present an extensive experimental evaluation of the proposed techniques and their optimizations, which shows that the proposed methods scale well for large data sets, and clearly outperform competitors.

The rest of the paper is organized as follows. In Section 2, we review related work, followed by an overview of the relevant background for the studied problem in Section 3. In Section 4, we introduce the VD-kNN method which relies on Voronoi diagrams and provides exact query results. Section 5 introduces the TkNN method which alleviates the load on the data owner, at the expense of slightly lower precision in returned results. In Section 6, we present performance optimizations. We discuss mechanisms for efficient handling of incremental updates in Section 7. We evaluate experimentally the performance of the proposed techniques in Section 8, and conclude with directions for future work in Section 9.

2 Related Work

Protecting location data is an important problem not only in the scenario of outsourced search services, but in a variety of other settings as well. For instance, two approaches for location protection have been investigated in the context of private queries to location-based services (LBS). The objective here is to allow a querying user to retrieve her nearest neighbor among a set of *public* points of interest without revealing her location to the LBS. The first approach is to use *cloaking regions (CRs)* [16], [19]. Most CR-based solutions implement the spatial k-anonymity paradigm and assume a three-tier architecture where a trusted anonymizer sits between users and the LBS server and generates rectangular regions that contain at least k user locations. This approach is fast, but not secure in the case of outliers. The second approach uses *private information retrieval* (PIR) protocols [7], [9]. PIR protocols allow users to retrieve an object X_i from a set $\mathrm{X} = \{X_1, X_2, \ldots, X_n\}$ stored by a server, without the server learning the value of i. The work in [7], [9] extends an existing PIR protocol for binary data to the LBS domain and proposes approximate and exact nearest neighbor protocols. The latter approach is provably secure, but it is expensive in terms of computational overhead.

Closer to our problem setting, location privacy has been considered in the domain of spatial database outsourcing [2], [3], [4], [5], [10]. In [10], the data points are encoded by the data owner according to a secret transformation: a Hilbert-curve mapping with secret parameters transforms 2-D points to 1-D. Users, who know the transformation key, map their queries to 1-D and the query processing is done in the 1-D space. However, the mapping can decrease the result accuracy and the transformation may be vulnerable to reverse-engineering. The work in [2] uses a secret matrix transformation to hide the data points. The data owner generates randomly a matrix M, and then transforms data points by multiplying them with M. Users transform their query points using multiplication with the inverse matrix M^{-1}. When the server receives the transformed data points from the data owner and a transformed query point from a user, it can determine which data point is nearest to the query point. In contrast with [10], the exact results are returned to the user. However, the matrix transformation is vulnerable to chosen plaintext attacks, as shown in [5].

Similar to [2], the work in [4] also uses a matrix transformation to protect both data and query privacy. Hence, it is also vulnerable to chosen plaintext attacks. In addition, given a kNN query, the server returns more than k data items to the client, and the client must filter out unnecessary data. This additional disclosure is undesirable, as the client who pays for k results should not be allowed access to more data points.

In [3], the data owner sends a shadow index to the client. The shadow index is encrypted by the data owner, and the decryption key is given to the server. The client traverses the shadow index in order to compute the distance between its query and a node of the index. The client computes encrypted distances and sends them to the server. However, the method requires the entire encrypted index to be transferred to the client. When there are a lot of data points, the size of the index grows large as well, making the method impractical.

Finally, the work in [5] shows how to stage effective attacks against methods such as [2], [3], and that solving the secure nearest neighbor problem is at least as hard as order preserving encryption (OPE) [20]. The proposed method from [5] returns a relevant partition E(G) from the entire encrypted data set, and E(G) is guaranteed to contain the answer for the NN query. However, the technique from [5] returns significantly more than k data items to the client.

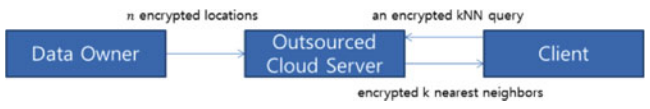


Fig. 1. System model.

3 Preliminaries

In this section, we introduce essential preliminary concepts, such as system model (Section 3.1), privacy model (Section 3.2) and an overview of the mutable order preserving encoding (mOPE) from [6] which we use as a building block in our work (Section 3.3).

3.1 System Model

The system model comprises of three distinct entities: (1) the data owner; (2) the outsourced cloud service provider (for short *cloud server*, or simply *server*); and (3) the client. The entities are illustrated in Fig. 1.

The data owner has a data set with n two-dimensional points of interest, but does not have the necessary infrastructure to run and maintain a system for processing nearest-neighbor queries from a large number of users. Therefore, the data owner outsources the data storage and querying services to a cloud provider. As the data set of points of interest is a valuable resource to the data owner, the storage and querying must be done in encrypted form (more details will be provided in the privacy model description, Section 3.2).

The server receives the data set of points of interest from the data owner in encrypted format, together with some additional encrypted data structures (e.g., Voronoi diagrams, Delaunay triangulations) needed for query processing (we will provide details about these structures in Sections 4 and 5). The server receives kNN requests from the clients, processes them and returns the results. Although the cloud provider typically possesses powerful computational resources, processing on encrypted data incurs a significant processing overhead, so performance considerations at the cloud server represent an important concern.

The client has a query point Q and wishes to find the point's nearest neighbors. The client sends its encrypted location query to the server, and receives k nearest neighbors as a result. Note that, due to the fact that the data points are encrypted, the client also needs to perform a small part in the query processing itself, by assisting with certain steps (details will be provided in Sections 4 and 5).

3.2 Privacy Model

As mentioned previously, the data set of points of interest represents an important asset for the data owner, and an important source of revenue. Therefore, the coordinates of the points should not be known to the server.

We assume an *honest-but-curious* cloud service provider. In this model, the server executes correctly the given protocol for processing kNN queries, but will also try to infer the

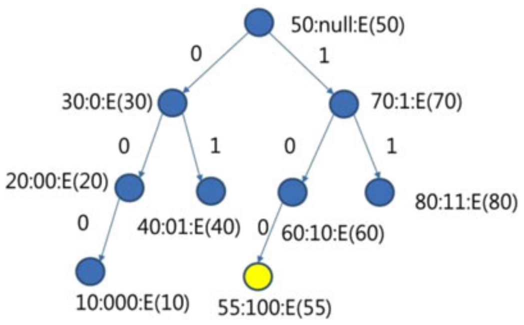
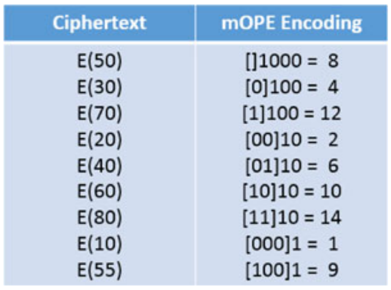


Fig. 2. mOPE tree: Inserting node E(55).



Ciphertext	mOPE Encoding
E(50)	[]1000 = 8
E(30)	[0]100 = 4
E(70)	[1]100 = 12
E(20)	[00]10 = 2
E(40)	[01]10 = 6
E(60)	[10]10 = 10
E(80)	[11]10 = 14
E(10)	[000]1 = 1
E(55)	[100]1 = 9

Fig. 3. mOPE table.

location of the data points. It is thus necessary to encrypt all information stored and processed at the server.

To allow query evaluation, a special type of encryption that allows processing on ciphertexts is necessary. In our case, we use the mOPE technique from [6]. mOPE is a provably secure order-preserving encryption method, and our techniques inherit the IND-OCPA security guarantee against the honest-but-curious server provided by mOPE.

Furthermore, we assume that there is no collusion between the clients and server, and the clients will not disclose to the server the encryption keys.

3.3 Secure Range Query Processing Method

As we will show later in Sections 4 and 5, processing kNN queries on encrypted data requires complex operations, but at the core of these operations sits a relatively simple scheme called *mutable order-preserving encryption* (mOPE) [6]. mOPE allows secure evaluation of range queries, and is the only provably secure order-preserving encoding system (OPES) known to date. The difference between mOPE and previous OPES techniques (e.g., Boldyreva et al. [11], [12]) is that it allows ciphertexts to change value over time, hence the *mutable* attribute. Without mutability, it is shown in [6] that secure OPES is not possible.

Since our methods use both mOPE and conventional symmetric encryption (AES), to avoid confusion we will further refer to mOPE operations on plaintext/ciphertexts as encoding and decoding, whereas AES operations are denoted as encryption/decryption.

The mOPE scheme in a client-server setting works as follows: the client has the secret key of a symmetric cryptographic scheme, e.g., AES, and wants to store the data set of ciphertexts at the server in increasing order of corresponding plaintexts. The client engages with the server in a protocol that builds a B-tree at the server. The server only sees the AES ciphertexts, but is guided by the client in building the tree structure. The algorithm starts with the client storing the first value, which becomes the tree root. Every new value stored at the server is accompanied by an insertion in the B-tree. Fig. 2 shows an example where plaintext values are also illustrated for clarity, although they are not known to the server (for simplicity we show a binary tree in the example).

Assume the client wants to store an element with value 55: it first requests the ciphertext of the root node from the server, then decrypts *E(50)* and learns that the new value 55 should be inserted in the tree to the right hand side of the root. Next, the client requests the right node of the root node and the server sends *E(70)* to the client. The process repeats recursively until a leaf node is reached, and 55 is inserted in the appropriate position in the sorted B-tree, as the left child of node 60. The client sends the AES ciphertext *E(55)* to the server which stores it in the tree. The *encoding* of value 55 in the tree is given by the path followed from the root to that node, where 0 signifies following the left child, and 1 the right child. In addition, the encoding of every value is padded to the same length (in practice 32 or 64 bits) as follows [6]:

$$\text{mOPE encoding} = [\text{mOPE tree path}]10\ldots0.$$

The server maintains a mOPE table with the mapping from ciphertexts to encodings, as illustrated in Fig. 3 for a tree with four levels (four-bit encoding). Clearly, mOPE is an order preserving encoding, and it can be used to answer securely range queries without need to decrypt ciphertexts.

In addition, the mOPE tree is a balanced structure. Using a B-tree, it is possible to keep the height of the tree low, and thus all search operations are efficient. In order to ensure the balanced property, when insertions are performed, it may be necessary to change the encoding of certain ciphertexts. Note that, the actual ciphertext image does not change, only its position in the tree, and thus its encoding, changes. Typically, mutability can be done very efficiently, and the complexity of the operation (i.e., the maximum number of affected values in the tree) is *O(log n)* where n is the number of stored values.

As shown in [6], mOPE satisfies IND-OCPA [6], i.e., indistinguishability under ordered chosen-plaintext attack. The scheme does not leak anything besides order, which is the intended behavior to support comparison on ciphertexts.

4 One Nearest Neighbor (1NN)

4.1 Voronoi Diagram-Based 1NN (VD-1NN)

In this section, we focus on securely finding the 1NN of a query point. We employ Voronoi diagrams [1], which are data structures especially designed to support NN queries. An example of Voronoi diagram is shown in Fig. 4. Denote the Euclidean distance between two points p and q by $dist(p,q)$, and let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n distinct points in the plane. The Voronoi diagram (or tessellation) of P is defined as the subdivision of the plane into n convex polygonal regions (called cells) such that a point q lies in the cell corresponding to a point p_i if and only if p_i is the 1NN of q, i.e., for any other point p_j it

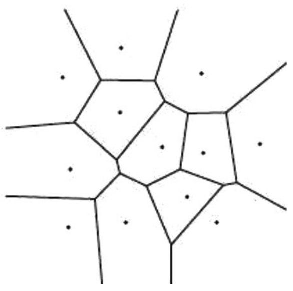


Fig. 4. Voronoi diagram.

holds that $dist(q, p_i) < dist(q, p_j)$ [1]. Answering a 1NN query boils down to checking which Voronoi cell contains the query point.

In our system model, both the data points and the query must be encrypted. Therefore, we need to check the enclosure of a point within a Voronoi cell securely. Next, we propose such a secure enclosure evaluation scheme.

4.2 Secure Voronoi Cell Enclosure Evaluation

Based on the secure range query processing method introduced in Section 3.3, we develop a secure scheme that determines whether a Voronoi cell contains the encrypted query point. Consider the sample Voronoi cell from Fig. 5. For simplicity, we consider a triangle, but the protocol we devise works for any convex polygon as a cell. The data owner sends to the server the encrypted vertices of the cell: $V_1(x_1, y_1), V_2(x_2, y_2)$ and $V_3(x_3, y_3)$.

Step 1: Filter Cells. Checking enclosure of a point within a convex polygonal region is expensive, so the server first performs a filtering step, where it checks if the query point is inside the minimum bounding rectangle (MBR) of the cell, identified by its lower-left (LL) and upper-right (UR) corners. Checking enclosure within a rectangle is much cheaper, and the polygon protocol is only performed for the cells that pass the filter. For the filtering step, the data owner needs not send any additional information to the server, since the coordinates of the MBR are already among the vertex coordinates. The data owner only has to send the indices of the four rectangle corner coordinates within the sequence of vertex coordinates, and the server will be able to compute rectangle enclosure.

By using the secure range query processing method, the server determines if the encrypted query point $Q(x_q, y_q)$ is inside the MBR or not by checking the following four conditions for every Voronoi cell:

$$x_{LL} < x_q, \tag{1}$$

$$x_q < x_{UR}, \tag{2}$$

$$y_{LL} < y_q, \tag{3}$$

$$y_q < y_{UR}. \tag{4}$$

In Fig. 5, the left side boundary of the MBR is given by coordinate x_1 and the right side by x_3. Similarly, the lower side of the MBR is given by coordinate y_3 and the upper side by y_2. If all the conditions hold, then the current cell is processed in Step 2, otherwise it is discarded. To improve performance, if a condition is not satisfied, the other conditions do not need to be checked, therefore reducing query processing time.

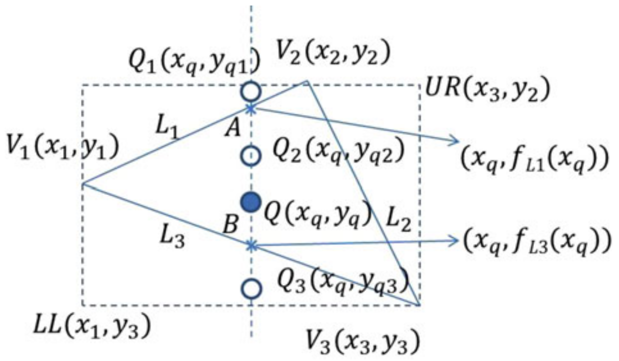


Fig. 5. Secure voronoi cell enclosure evaluation.

Step 2: Calculate Intersection Edges. For cells that passed Step 1, the server determines the intersection points of the cell edges with the vertical line that passes through the query point. Note that, since Voronoi cells are convex polygons [1], the vertical line always intersects exactly two edges of the cell. In this step, the server determines the two edges which the vertical line intersects.

In Fig. 5, the vertical line meets the Voronoi cell at points A and B. Thus, the vertical line meets edges L_1 and L_3. This can be determined using the secure range query processing method as follows:

$$x_1 < x_q < x_2 \text{ for edge } L_1, \tag{5}$$

$$x_1 < x_q < x_2 \text{ for edge } L_3. \tag{6}$$

Since $x_q < x_2$, the vertical line does not meet edge L_2, so the server need not consider edge L_2 in Step 3, but only the two edges L_1 and L_3. Recall that, all comparisons are done on encoded data, so no information about edge coordinates is learned by the server.

Step 3: Determine Polygon Enclosure. In the third step, the server determines whether the query point is "in-between" the two sides found in Step 2. Namely, the query point needs to be below one of the sides and above the other. There are two conditions to be checked, except that this time the sides may be neither horizontal, nor vertical, which makes the evaluation more complicated.

Continuing the earlier example, the server must check whether the query point is below the edge L_1 and above L_3. From Step 1, we know that the query point is within the cell MBR. Denote by f_L the line equation corresponding to side L. Then we have three possible cases for query point placement to consider: (i) $y_{q1} > f_{L1}(x_q)$ and $y_{q1} > f_{L3}(x_q)$ (illustrated by Q$_1$ in Fig. 5); (ii) $y_{q2} < f_{L1}(x_q)$ and $y_{q2} > f_{L3}(x_q)$ (illustrated by Q$_2$); and (iii) $y_{q3} < f_{L1}(x_q)$ and $y_{q3} < f_{L3}(x_q)$ (illustrated by Q$_3$). y_{q1} is the y-coordinate of Q_1, y_{q2} is the y-coordinate of Q_2 and y_{q3} is the y-coordinate of Q_3. In the first and third case, the query point is outside the cell. The second case is the only one when the query point is inside the polygon. In the following, we show how to check these cases.

For edge L_1 in Fig. 5, the line equation is

$$y = (y_2 - y_1)/(x_2 - x_1) * (x - x_1) + y_1. \tag{7}$$

When we plug x_q into Eq. (7), if y_q is less than y, then the query point is in the lower side of L_1. On the other hand, when we plug x_q into the equation of L_3, if y_q is greater than y, then the query point is in the upper side of L_3. The following condition must be satisfied if the query point is in the lower side of L_1, where $S_{i,j}$ denotes the slope of the edge between two Voronoi vertices V_i and V_j,

$$\begin{aligned} y_q &< (y_2 - y_1)/(x_2 - x_1) * (x_q - x_1) + y_1 \\ \Leftrightarrow y_q &< S_{1,2} * (x_q - x_1) + y_1. \end{aligned} \tag{8}$$

The values of x_q and y_q are variable for each query, but the Voronoi diagram does not change with the query, so, x_i, y_i, and $S_{i,j}$ remain constant. We can rewrite the equation above as follows:

$$L_{1,2} = y_q - S_{1,2} * x_q < -1 * S_{1,2} * x_1 + y_1 = R_{1,2}, \tag{9}$$

where we denote the right-hand side and the left-hand side by $R_{i,j}$ and $L_{i,j}$, respectively. $R_{i,j}$ is constant for a given query, and can be determined by the data owner when she/he uploads the database to the server. In addition, the data owner encrypts the value of slope $S_{i,j}$ with conventional encryption (e.g., AES) and sends it to the server.

```
VD-1NN protocol
1. Data Owner sends to Server the encoded Voronoi cell vertices coordinates, MBR boundaries for each cell, encoded right-hand side $R_{i,j}$, and encrypted $S_{i,j}$ for each cell edge.
2. Client sends its encoded query point to the Server.
3. Server performs the filter step, determines for each kept cell the edges that intersect the vertical line passing through the query point and sends the encrypted slope $S_{i,j}$ of the two edges to the Client.
4. Client computes the left-hand side $L_{i,j}$, encodes it and sends it to the Server.
5. Server finds the Voronoi cell enclosing the query point and returns result to Client.
```

For each of the intersecting edges determined in Step 2, the server assembles Eq. (9) and sends the encrypted value $S_{i,j}$ for each of the two edges to the client. The client decrypts $S_{i,j}$ values with the secret AES key shared with the data owner. Next, the client computes $L_{i,j}$ (Eq. (9)), encodes it, and sends it back to the server. The server is then able to check enclosure for the current cell, and thus find the final query result. The following pseudocode summarized the protocol, and Fig. 6 captures the communication pattern between parties.

4.3 Performance Analysis

The data owner computes the order-1 Voronoi diagram of the data set, determines the MBR boundaries of each Voronoi cell and encodes using mOPE the cell vertices' coordinates, as well as the right side $R_{i,j}$ of Eq. (9) for each edge of a Voronoi cell. The slopes $S_{i,j}$ are encrypted using symmetric encryption (e.g., AES).

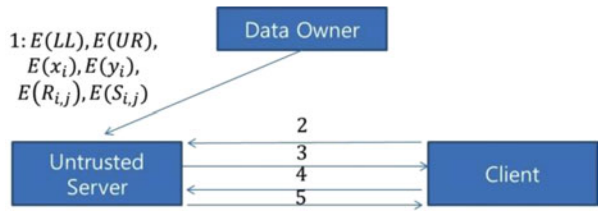


Fig. 6. VD-1NN.

Generation time for the Voronoi diagram is $O(nlogn)$ using Fortune's algorithm [1]. The number of Voronoi vertices that require mOPE encoding in a set of n data points is at most $2n - 5$ [1]. Thus, the time to encode Voronoi points is proportional to $4n$ since each Voronoi point has a x-coordinate and a y-coordinate. Furthermore, the right side $R_{i,j}$ of Eq. (9) must be encoded for each edge. The number of edges in a Voronoi diagram is at most $3n - 6$. The total number of mOPE encoding operations is proportional to $7n$. The slopes $S_{i,j}$ are encrypted using AES encryption and do not require mOPE encoding. In total, the Data Owner performs *3n* AES encryption and *7n* mOPE encoding operations.

In Line 2 of the pseudocode, the client encodes the query point with cost $O(1)$. In Line 4, the client encodes the left side $L_{i,j}$ of the two edges of the Voronoi cells whose MBR boundaries contain the query point. The number of Voronoi cells considered in this step is typically small, as we have found experimentally.

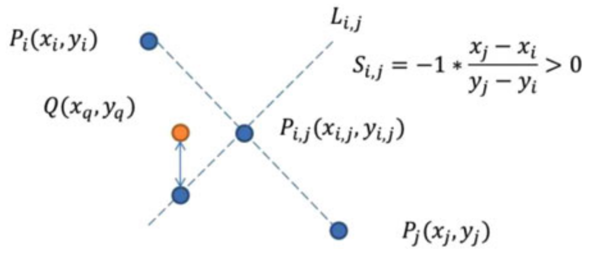
In Line 3, the server finds Voronoi cells whose boundaries enclose the query point. Since there are n Voronoi cells, the processing time is $O(n)$. When there are a lot of data points, the time to filter Voronoi cells may be high. In Section 6, we provide several optimizations to reduce this computational time.

5 k Nearest Neighbor (kNN)

To support secure kNN queries, where k is fixed for all querying users, we could extend the VD-1NN method from Section 4 by generating order-k Voronoi diagrams [1]. However, this method, which we call VD-kNN, has several serious drawbacks:

1. The complexity of generating order-k Voronoi diagrams is either $O(k^2 nlogn)$ [21] or $O(k(n-k)logn + nlog^3 n)$ [22], depending on the approach used. This is significantly higher than $O(n \cdot logn)$ for order-1 Voronoi diagrams.
2. The number of Voronoi cells in an order-k Voronoi diagram is $O(k(n-k))$, or roughly kn when $k \ll n$. That leads to high data encryption overhead at the data owner, as well as prohibitively high query processing time at the server (a k-fold increase compared to VD-1NN).

Motivated by these limitations of VD-kNN, we first introduce a secure distance comparison method (SDCM) in Section 5.1. Next, in Section 5.2 we devise Basic kNN (BkNN), a protocol that uses SDCM as building block,



$P_i(x_i, y_i)$
$L_{i,j}$
$S_{i,j} = -1 * \frac{x_j - x_i}{y_j - y_i} > 0$
$Q(x_q, y_q)$
$P_{i,j}(x_{i,j}, y_{i,j})$
$P_j(x_j, y_j)$

Fig. 7. Secure distance comparison method.

and answers kNN queries using repetitive comparisons among pairs of data points. BkNN is just an auxiliary scheme, very expensive in itself, but it represents the starting point for Triangulation kNN (TkNN), presented in Section 5.3. TkNN builds on the BkNN concept and returns exact results for $k = 1$. For $k > 1$, it is an approximative method that provides high-precision kNN results with significantly lower costs.

5.1 Secure Distance Comparison Method (SDCM)

Consider two given encrypted data points P_i and P_j and encrypted query point $Q(x_q, y_q)$. If we can securely test which data point is closer to the query point, then by repeatedly applying this test we can find all k nearest neighbors of Q. In Section 4.2, we showed how to determine whether the query point is below or above an edge of a Voronoi cell. SDCM is an extension of that scheme.

Consider the example in Fig. 7, where there are two data points and one query point. First, the data owner computes the middle point of the segment that connects the two data points, denoted by $p_{i,j}$, as well as the perpendicular bisector $L_{i,j}$ of the segment. The slope of the bisector is denoted by $S_{i,j}$. The bisector equation is

$$\begin{aligned} y = -1 * (x_2 - x_1)/(y_2 - y_1) * (x - x_{i,j}) + y_{i,j} \\ \Leftrightarrow\ y = S_{i,j} * (x - x_{i,j}) + y_{i,j}. \end{aligned} \tag{10}$$

When we plug x_q into the equation, it follows that the query point is in the upper side of the bisector, hence P_i is closer to Q than P_j, if and only if

$$\begin{aligned} y_q\ >\ S_{i,j} * (x_q - x_{i,j}) + y_{i,j} \Leftrightarrow L_{i,j} = y_q - S_{i,j} * x_q \\ >\ -1 * S_{i,j} * x_{i,j} + y_{i,j} = R_{i,j}. \end{aligned} \tag{11}$$

Similar to the case of Section 4.2, we observe that the right-hand side $R_{i,j}$ of Eq. (11) is independent of the query point, whereas the left-hand side $L_{i,j}$ depends on the query point. The data owner can thus encode the right-hand side and send it to the server, together with the slope $S_{i,j}$ of the bisector. Recall that, the slope may be encrypted using conventional encryption, e.g., AES.

At query time, in order to determine which data point is closer, the server sends the encrypted slope $S_{i,j}$ to the client. The client computes the left-hand side, encodes it and sends it back to the server, which in turn determines the outcome of inequality in Eq. (11).

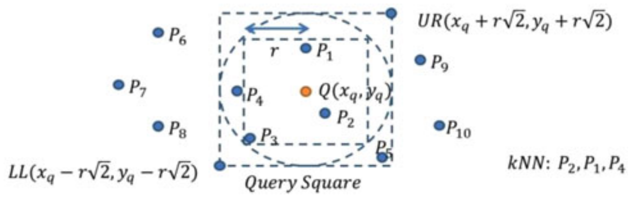


Fig. 8. BkNN using query square.

5.2 Basic k Nearest Neighbor (B-kNN)

Based on SDCM, we introduce the basic secure kNN scheme (BkNN), which in itself is not efficient, but it illustrates the general concept based on which we introduce a more efficient approach in Section 5.3.

For each pair of encrypted data points, the server must determine according to SDCM which data point is closer to the encrypted query point. When there are n data points, the perpendicular bisector must be determined for every pair, for a total of $n(n-1)/2$ bisectors. The encoded right-hand side and slope must be sent for each bisector from the data owner to the server, and the server needs to perform $n(n-1)/2$ comparisons on encoded data to find the first nearest neighbor. Clearly, such cost is prohibitive.

To reduce this overhead, we propose a basic k nearest neighbor scheme which uses the concept of query squares. We illustrate this concept in Fig. 8: the small query square with side $2r$ corresponds to a range query selected by the user, whereas the large query square is computed as the smallest square that encloses the circle in which the small query square is inscribed.

Suppose a user wishes to retrieve from the server the answer to a 3NN query ($k = 3$). Assume the small query square contains three data points and the large query square contains five data points. Note that, it is possible for a data point that is outside the query square (in our example P_4) to be closer to the query point than some point inside the square (say P_3). This means that if the small query square contains at least k data points, the large query square will certainly contain k nearest neighbors. If the small query square does not have at least k data points, then the client will generate a larger query square and re-issue the query, in a process similar to incremental range queries.

The size S_{sq} of the small query square can be determined by the client according to the estimated number of data points in the data domain. For instance, when the number of data points is n and the size of the data space side is l, then assuming the data points are uniformly distributed, we have

$$k : n = (2r)^2 : l^2 \Leftrightarrow S_{sq} = 2r = \sqrt{kl^2/n} = l\sqrt{k/n}.$$

The size of the query square is proportional to k and inversely proportional to n. If k is large, we need a large query square, whereas if n is large (i.e., the data set has a higher density), then a smaller query square suffices.

The number of data points in the large query square is expected to be $O(k)$, so the number of bisectors used in the query processing step is $O(k(k-1)/2)$, which is much cheaper than $O(n(n-1)/2)$.

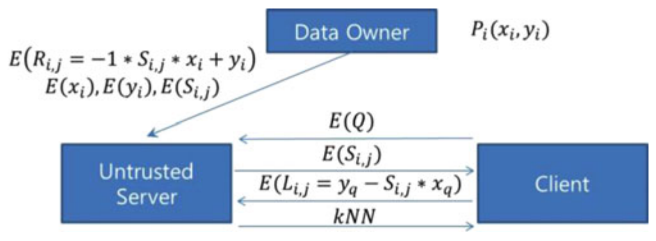

Data Owner
$P_i(x_i, y_i)$
$E(R_{i,j} = -1 * S_{i,j} * x_i + y_i)$
$E(x_i), E(y_i), E(S_{i,j})$
$E(Q)$
$E(S_{i,j})$
$E(L_{i,j} = y_q - S_{i,j} * x_q)$
kNN
Untrusted Server
Client

Fig. 9. BkNN protocol.

The BkNN protocol is summarized in the following pseudocode, and a system view with the communication pattern between parties is provided in Fig. 9.

The concept of using query squares has been used previously in [4], but that scheme uses an encryption method which is not secure against chosen plaintext attacks, and it also returns redundant results to the client.

B*k*NN Protocol

```
1. Data Owner sends to Server: all encoded data points,
   and for each pair of points the encoded right-hand
   side $R_{i,j}$ of Eq. (11), and encrypted slopes $S_{i,j}$.
2. Client sends the encoded query to Server.
3. Server finds the data points in the large query square
   and sends their AES-encrypted slopes $S_{i,j}$ to Client.
4. Client computes the encoded left-hand sides $L_{i,j}$ of
   Eq. (11) and sends them to Server.
5. Server returns $k$ result points to Client.
```

Performance analysis. Even if the query processing time is significantly reduced to $O(k(k-1)/2)$by using the query square concept, BkNN still incurs significant data encryption time $O(n(n-1)/2)$, because all perpendicular bisector slopes need to be sent to the server. Next, we focus on reducing data encryption time.

5.3 Triangulation-Based kNN (TkNN)

Triangulation-based kNN (TkNN) reduces the overhead at the data owner. TkNN is an approximate method for $k > 1$, i.e., it may not always return the true kNN. However, as we show later in Section 8, it achieves high precision in practice. The Delaunay triangulation is the dual of the order-1 Voronoi diagram [1], and is illustrated in Fig. 10. The thick lines show the edges of the triangulation, whereas the dotted lines show the edges of the Voronoi cells. Let b denote the number of points that lie on the boundary of the convex hull of the triangulation. Then the triangulation has $2n$-2-b triangles and $3n$-3-b edges.

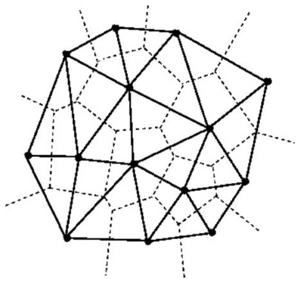


Fig. 10. Triangulation example.

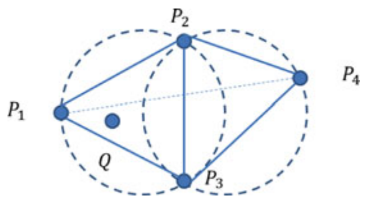


Fig. 11. TkNN evaluation.

In TkNN, the data owner computes bisectors for each edge of the triangulation, for a total of $3n$ bisectors. This is significantly less than BkNN, which requires $n(n-1)/2$ bisectors. In addition, since the Delaunay triangulation is the dual of the order-1 Voronoi diagram, the data generation time (i.e., the time to compute the data structure in plaintext) is $O(nlogn)$, no larger than the VD-1NN case.

In TkNN, a bisector is determined for each edge of the triangulation. In the example of Fig. 11, there are five edges and a bisector is determined for each edge. Note that, it is not necessary to determine a bisector for the pair of data points P_1 and P_4 since they do not take part as vertices in any triangle together. Hence, we can reduce the data encryption time and the query processing time to $O(n)$, and the query encryption time to $O(k)$.

Using SDCM for the left-hand triangle in Fig. 11, the server determines which data point is closer among P_1, P_2, P_3. In addition, from the right-hand triangle, the server determines which data point is closer among P_2, P_3, P_4. For instance, from the left-hand triangle, we know that P_1 is the nearest to the query point, P_3 is second-nearest, and P_2 is third-nearest. From the right triangle, P_3 is nearest, P_2 is second-nearest, and P_4 is third-nearest. Finally, combining the information from these two triangles, P_1 is the 1NN, P_3 the 2NN, P_2 the 3NN and P_4 is the 4NN. The server is able to determine the query answer completely from processing the triangulation.

However, the performance advantage of TkNN comes with a tradeoff in query accuracy. Specifically, when two data points do not exist in the same triangle, a bisector between the two data points is not determined. In this case, the server may not be able to determine which one between the two data points is closer to the query point.

For example, in Fig. 12, when the query point is closer to P_2, we can determine from the left-hand triangle that P_2 is nearest to the query point, P_1 is second-nearest, and P_3 is third-nearest. From the right-hand triangle, it results that P_2 is the nearest, P_4 is second-nearest, and P_3 is third-nearest.

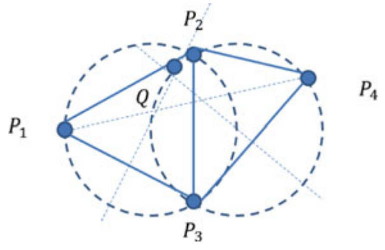


Fig. 12. Limitation of TkNN.

TABLE 1
Performance of VD-kNN, BkNN, and TkNN

	VD-kNN	BkNN	TkNN
Data Generation Time at Data Owner (on plaintext)	$k^2 nlogn$ or $k(n-k)+nlog^3 n$	n/a	$nlogn$
Data Encoding Time at Data Owner	$7kn$	$n(n\text{-}1)/2$	$5n$
Query Encoding Time at Client	$O(1)$	$O(k(k\text{-}1)/2)$	$O(k)$
Query Processing Time at Server	kn	$n(n\text{-}1)/2$	n

From these two triangles, we can establish a partial order for the four data points as follows:

$$P_2 < \{P_1, P_4\} < P_3.$$

The first nearest neighbor is always correct. However, in cases where $k > 1$, the rest of the returned k results may be approximate. For example, when a 2NN query is issued, the server may return P_2 and P_4 to the client as the result. P_2 is indeed the first nearest neighbor, but the 2NN is actually P_1.

Performance analysis. The data generation time of TkNN is $O(nlogn)$ and the data encoding time is $O(5n)$ (accounting for n two-dimensional points coordinates and $3n$ bisector right-hand side equation values). This is superior to VD-1NN which requires $O(7n)$ data encoding time ($2n$ two-dimensional Voronoi points and $3n$ right-hand side equation values).

In addition, since VD-kNN has kn Voronoi cells, it has $O(kn)$ query processing time. Triangulation has n data points, hence only $O(n)$ query processing time. TkNN is k times faster than VD-kNN in terms of query processing.

Finally, BkNN has $O(n(n–1)/2)$ data encoding time and $O(n(n–1)/2)$ query processing time. A performance comparison of the three schemes is provided in Table 1.

6 Optimizations

Our proposed methods for secure nearest-neighbor evaluation perform query processing on top of encrypted data, and for this reason they are inherently expensive. It is a well-known fact that achieving security by processing on encrypted data comes at the expense of significant computational overhead. Next, we propose two optimizations that aim at reducing this cost.

6.1 Hybrid Query Processing Using Kd-Trees

As shown in Table 1, the query processing time of VD-1NN and TkNN is $O(n)$. If there are a lot of data points, which is likely to be the case in cloud deployments, the query processing time will be several seconds or higher. Since the server needs to return the result to the client within a very short time for good usability (typically less than 1 second), we propose a hybrid query processing method using kd-trees [1], [13].

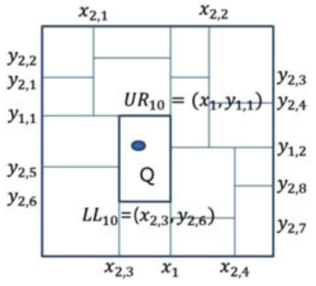


Fig. 13. Partitioning data points using a kd-tree.

The data owner performs a pre-processing phase in which the set of data points is partitioned according to a kd-tree space decomposition. Fig. 13 illustrates how the splitting is done. First, the data owner chooses a vertical line (e.g., $x = x_1$)and splits the set of points into two subsets of equal cardinality. Each of the resulting subsets is further split along a horizontal line (e.g., $y = y_{1,1}$ and $y = y_{1,2}$). In general, the data owner splits with a vertical line nodes whose depth is even, and with a horizontal line nodes whose depth is odd. The splitting ends when the cardinality of a node drops below a certain threshold.

Each of the resulting 16 partitions in Fig. 13 has roughly n/16 data points, and is enclosed by its own MBR. MBRs of different nodes do not overlap. For example, the 10th subspace's lower bound is $(x_{2,3}, y_{2,6})$ and the upper bound is $(x_1, y_{1,1})$. Each MBR is encrypted by the data owner and sent to the server.

When the client sends an encrypted query to the server, the server finds a subspace which contains the encrypted query point. For instance, for the example in Fig. 13, secure range query processing first performs the following test:

$$\begin{aligned} x_{2,3} &< x_q < x_1 \\ y_{2,6} &< y_q < y_{1,1}. \end{aligned} \tag{12}$$

If these two conditions are satisfied for some partition, then that partition contains the query point. The subspace has roughly $n/16$ data points. Next, the server applies VD-1NN or TkNN only to that partition. Consequently, query processing time is reduced to about 1/16 of the query processing time when there are n Voronoi cells or n data points. Furthermore, as the number of data points increases, the data owner can choose a larger number of partitions. The disadvantage of this method is that the server learns the count of data points in each partition, but since the partition MBRs are encrypted, that does not disclose significant information (all partitions have roughly the same cardinality).

6.2 Parallel Processing

In order to reduce the query processing time, the server can use parallel processing. Note that, the operations performed by the server for each Voronoi cell or triangle are independent from each other. Hence, each object (or partition of objects) can be dispatched for processing to a different processor. The algorithms for querying are embarrassingly parallel, which can lead to very good speedup values.

Nowadays, a lot of machines have multi-core processors, so the parallel processing optimization can be quite effective in practice. In addition, in the case of clusters of computers, the query processing time can be further reduced by using a parallel programming environment such as the message passing interface (MPI). We will show the effectiveness of parallel processing on reducing query processing time in the experimental evaluation.

7 Incremental Updates

So far, we have considered only the case of static data sets of points. However, in practice, data set of locations of interest change quite frequently. Re-generating a new encrypted data set at the data owner each time some points change incurs a prohibitively expensive overhead. In this context, it is important to address the issue of incremental updates.

When data points move, it is not necessary to recalculate the entire Voronoi diagram or Delaunay triangulation. These data structures can be updated in an incremental manner. In addition, the topological structures of the Voronoi diagram and the Delaunay triangulation are locally stable under sufficiently small continuous motions of the data points [14]. For incremental updates, we consider only TkNN and VD-1NN. BkNN is not considered since it is not based on triangulations or Voronoi diagrams, and handling updates in the case of BkNN is straightforward, albeit inefficient. Specifically, in the case of BkNN, when a data point moves, n slopes are changed and must be reencrypted.

In the case of TkNN, if a data point moves, the position of the data point is changed, and the slopes of d edges connected to the data point are also changed. The complexity of the update is $O(d)$, where d is the degree of the data point. Then, the position of the data point and the right-hand sides of Eq. (9) must be re-encoded with mOPE, whereas the slopes of the edges are re-encrypted with AES encryption. Recall that, the encoded coordinates of the MBR are among those of the data points, so no separate re-encoding for these is required.

In the case of VD-1NN, if a data point in the triangulation moves, the slopes of three edges corresponding to the cell vertices of that point also change. Then, the neighbors of that cell in the tessellation may also change. In total, when a data point moves, d Voronoi points and $2d$ Voronoi edges are changed and must be re-encoded. Note that, each cell vertex has three edges. However, an edge is shared with adjacent cell vertices.

Next, we discuss how topological changes are performed. The work in [14] shows how changes can be characterized as swaps of adjacent triples in the triangulation. Recall that a Voronoi diagram is the dual of a Delaunay triangulation. When a data point P_l leaves the circle determined by three points $\mathrm{C}(P_i, P_j, P_k)$, an inactive triple $\{P_i, P_j, P_k\}$ becomes activated. On the other hand, when a data point P_l enters the circle, an active triple $\{P_i, P_j, P_k\}$ becomes deactivated. Fig. 14 illustrates this concept.

The structure update proceeds in two steps: a preprocessing step and an iteration step [14]. In the preprocessing

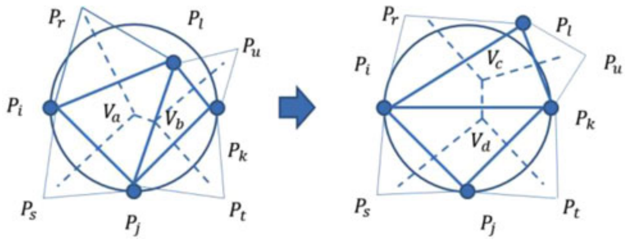


Fig. 14. Change of the topological structure.

step, the data owner computes the triangulation and calculates the potential topological events. A potential topological event is a pair of two adjacent triple which is called a *quadrilateral*, e.g., $\{P_i, P_j, P_k, P_l\}$. The data owner builds up a balanced SWAP-tree. In the iteration step, when there is a topological event, the data owner processes the event and updates the SWAP-tree.

The number of pairs of two adjacent triples is equal to the number of edges which is $3n$. The preprocessing step requires $O(nlogn)$ time. Next, when there is a swap, it determines the removal of only four quadrilaterals (e.g., $\{P_r, P_i, P_j, P_l\}$) while other four quadrilaterals are generated (e.g., $\{P_r, P_i, P_k, P_l\}$). The update time is $O(logn)$ [14].

There are two separate cases

1. *The data point P_l moves within the circle*. In this case, the topology is not changed. However, the position of the data point and the slopes of the edges connected to the point are changed. Then, for TkNN, only the point P_l and the edges including the point P_l and MBR boundaries of the triangles including the point P_l should be updated. The update time is $O(d)$ where d is the degree of the data point.

In the case of VD-1NN, since the data point P_l has d edges in the triangulation, d Voronoi vertices are changed, as well as 2d edges. The update time is $O(4d)$ where d is the degree of the Voronoi points.

1. *The data point P_l moves outside the circle*. In this case, the topology is changed. The time to update the triangulation is $O(logn)$ as explained earlier. In addition, for TkNN, the moving data point, d edges connected to it and the MBR boundaries of the triangles containing the data point should be updated. For VD-kNN, two Voronoi points V_a and V_b are deleted and two new Voronoi points V_c and V_d are inserted. Then, $O(d)$ Voronoi points, $O(2d)$ edges including the Voronoi points, and the MBR boundaries of the Voronoi cells corresponding to the Voronoi points should be updated. The total update time is $O(logn + 4d)$.

In summary, the incremental update of TkNN is more efficient than that of VD-1NN, since TkNN has $\mathrm{O}(logn + \mathrm{d})$ time complexity versus VD-1NN which has $\mathrm{O}(logn + 4d)$ time complexity.

8 Experimental Evaluation

8.1 Experimental Setup

We developed a Java prototype which implements the data owner, the server and the client protocols. We used the

Qhull library [15] to generate order-1 Voronoi diagrams and Delaunay triangulations. We implemented mOPE [15] using 32-bit encoding. The parallel computing section of our code was implemented using Java threads. Our experimental testbed consists of an Intel i7 CPU machine with four cores.

We used data sets of two-dimensional point coordinates ranging in cardinality from 200,000 to 1 million. We consider a uniform distribution of points in the unit space. We emphasize that, in the case of processing on encrypted data, the actual data distribution has little or no effect on performance, since all values are treated in a similar way in encrypted form. Therefore, we omit results obtained for other distributions. For encryption of slopes, we used 128 bit AES. The communication bandwidth for the wireless connection between the server and the client is set to 1Mbps.

The main performance metrics used to evaluate the proposed techniques are query response time and communication cost. The response time measures the duration from the time the query is issued until the results are received at the client. It includes the computation time at the server and the client, as well as the time required for transfer of final and intermediate results between client and server. Communication cost (measured in kilobytes) is important given that many wireless providers charge customers in proportion to the amount of data transferred.

We briefly review the functionality of the proposed methods. In the setup phase, the data owner builds the Voronoi diagram or Delaunay triangulation for the data set, encrypts these structures and sends them to the server. At runtime, there are two steps for each method:

VD-kNN. 1) The client sends its encoded query point to the server which finds the Voronoi cells whose MBRs enclose the query point. For each of these cells, the server sends the encrypted slopes $S_{i,j}$ of two cell edges intersecting the vertical line passing through the query point. 2) The client computes the left-hand sides $L_{i,j}$ (Eq. (9)) and sends their ciphertexts to the server, which finds the Voronoi cell enclosing the query point.

TkNN. 1) The client sends the encoded query square to the server, and the server finds the data points enclosed by the square. The server sends to the client the encrypted slopes $S_{i,j}$ of the perpendicular bisectors corresponding to each such data points. 2) The client computes the encoded left sides $L_{i,j}$ (Eq. (11)) and sends them to the server which finalizes processing and returns the results to the client.

We use as benchmark the method from [3] which relies on ASM-PH encryption and builds an encrypted R-tree index (shadow index) on top of the data. The complete tree is sent to the client, who engages in a multiple-round index traversal protocol with the server.

In Sections 8.2 and 8.3 we evaluate our techniques for 1NN and kNN queries, respectively. Next, in Section 8.4 we measure the overhead incurred at the data owner, which includes the time required to generate the Voronoi diagrams or Delaunay triangulations on plaintexts, as well as encoding/encryption time of these structures. Section 8.5 evaluates the precision of TkNN, whereas Section 8.6 measures the performance of handling updates.

8.1.1 1NN

Fig. 15a shows the query response time for all considered methods. For the benchmark method from [3] (label ASM-PH), the cost of transferring the shadow index is very large, as the index can grow to more than 100 megabytes for the considered data set. The authors in [3] argue that the cost of index transfer may be amortized over multiple queries. Even in this case, ASM-PH is at least an order of magnitude slower than our techniques. Therefore, we omit it from subsequent results.

Fig. 15b shows the communication cost for VD-1NN and TkNN. The methods exhibit comparable costs, with VD-1NN slightly more expensive, due to the fact that more slopes need to be sent for a Voronoi cell. The absolute values do not exceed 4 kilobytes, even for the largest data set considered.

Fig. 15c provides a breakdown of the response time into client CPU time, server CPU time and communication time. Note that, for both proposed methods the client time is a negligible fraction of the total time. This is a desirable feature, as clients are lightweight devices without powerful computation capabilities. In the case of VD-1NN, the server CPU time is the predominant source of overhead, whereas for TkNN there is a balanced split between server CPU and communication time. The higher server CPU time for VD-1NN is due to the fact that it needs to inspect four values for the MBR of each Voronoi cell, whereas TkNN needs only two values for each data point. Furthermore, in the mOPE tree, the height of VD-1NN is higher than that of TkNN since VD-1NN needs to represent 2n Voronoi points, compared to n data points for TkNN.

Overall, VD-1NN is considerably costlier than TkNN. However, the absolute response time is less than 200 msec in the worst case, which proves the practical applicability of both proposed methods. The response time of TkNN is always below 70 msec.

8.1.2 kNN

As discussed in Section 5, the cost of VD-kNN grows as O ($k^2 nlogn$) when k increases, due to the need to create an order-k Voronoi diagram. Thus, VD-kNN is not suitable for larger values of k. In this section, we consider only TkNN, and we compare its performance against ASM-PH [3]. As TkNN is highly parallelizable, we consider both the serial algorithm as well as a version with four CPU cores, which also partitions the dataspace into four regions, as discussed in Section 6.1.

Fig. 16a shows that the cost of ASM-PH with index transfer is prohibitively expensive for larger values of k as well. When k increases, the gap between ASM-PH without index transfer and TkNN get smaller, but TkNN still outperforms in each case. Furthermore, parallelism increases considerably the performance of TkNN. Note that, ASM-PH cannot be parallelized, due to its sequential nature in traversing the encrypted index. We do not consider ASM-PH further in this section.

Fig. 16b presents the communication cost for TkNN as k increases. Each line in the graph corresponds to a different setting for data set size. The amount of communication

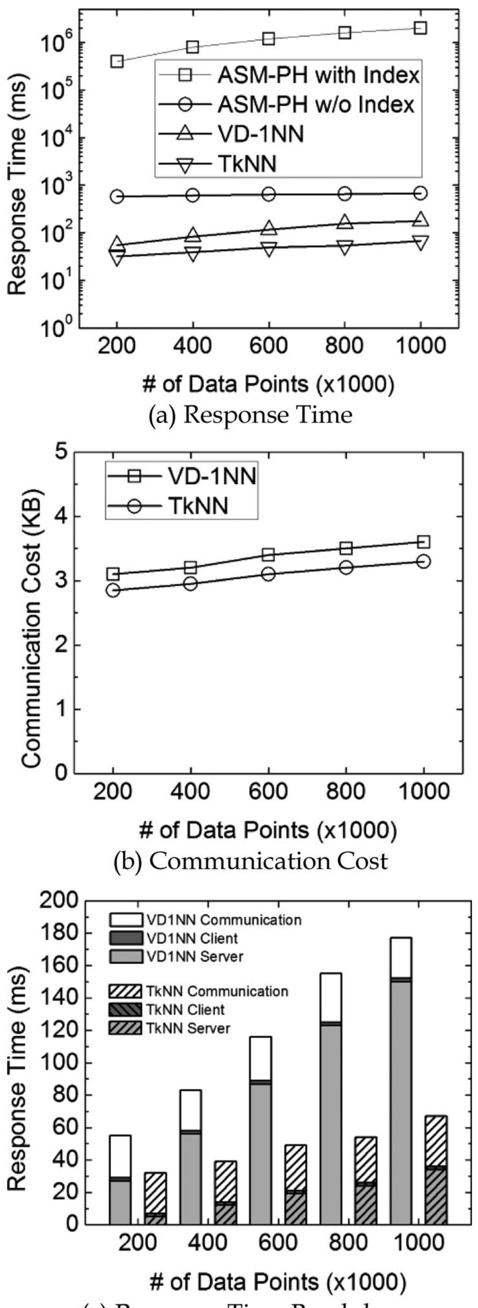


Fig. 15. 1NN Results.

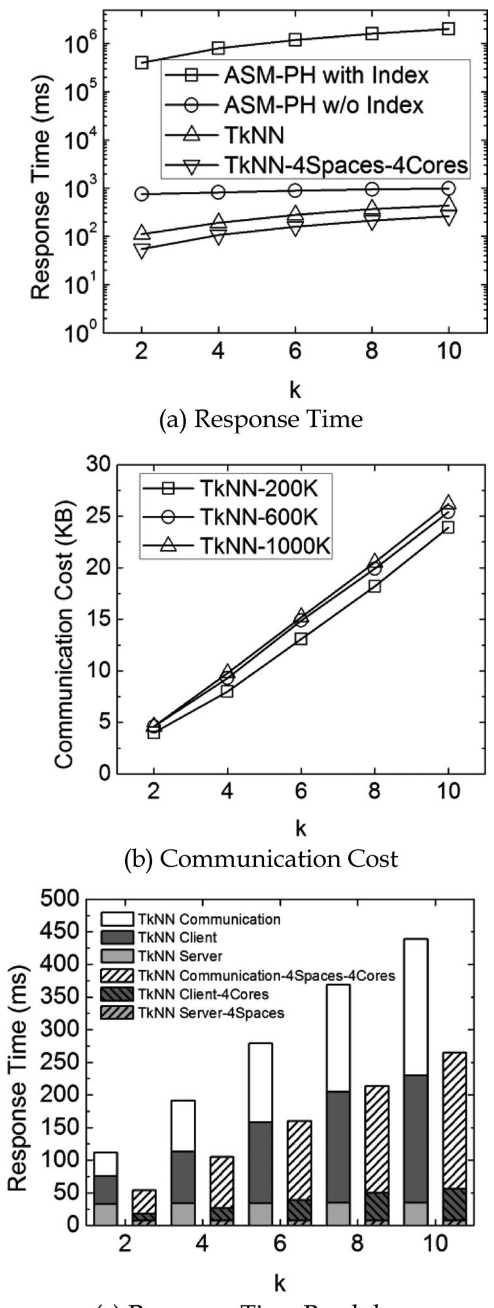


Fig. 16. kNN Results.

grows linearly with k, which is intuitive, as a proportionally larger number of results need to be returned to the client. Interestingly, increases in data set size do not determine a significant increase in the amount of communication required.

Fig. 16c provides a breakdown of the response time into client CPU time, server CPU time and communication time. The CPU time is significantly reduced by using parallelism. The split of the data set into four sub-spaces using a kd-tree further improves performance. The overall response time never exceeds half a second for the considered range of k values, whereas the parallel version halves the time to 250 msec.

8.2 Data Encryption Time at the Data Owner

Fig. 17 shows the data encryption time at the data owner for VD-1NN and TkNN. VD-1NN generates $2^{*}n$ Voronoi points, whereas TkNN has n data points. In addition, the data owner must encrypt the right side $R_{i,j}$ for each edge of every Voronoi diagram cell and triangulation object. The total numbers of such edges is $3n$ for both VD-1NN and TkNN. The overall data encryption overhead of VD-1NN is proportional to $7n$, whereas that of TkNN is proportional to $5n$. Fig. 17 captures this advantage of approximatively 30 percent that TkNN has over VD-1NN.

If the case of VD-kNN (not shown in the graph), which has $k^{*}n$ Voronoi cells, $2kn$ Voronoi points and $3kn$ edges are present, leading to an encoding overhead that is

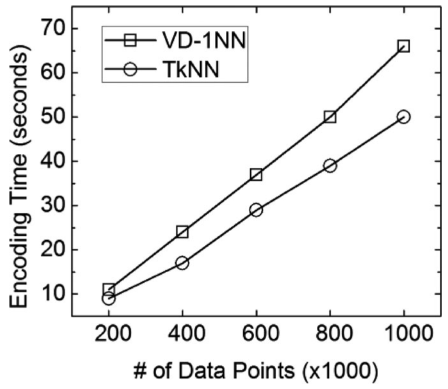


Fig. 17. Data encryption time.

proportional to $7kn$. This is another reason why VD-kNN is not suitable for larger k values. So in addition to the query response time evaluated in Section 8.2, TkNN also has an advantage with respect to data encoding/encryption time for larger k values compared to VD-kNN.

8.3 Precision of TkNN

Recall from Section 5.3 that TkNN yields approximate results for $k > 1$, since perpendicular bisectors are determined only for edges in the triangulation. When two data points do not exist in the same triangle, TkNN may not be able to determine which data point is closer to the query point. However, in many cases we can determine a total order from the partial orders given by individual triangles.

Next, we measure the precision of TkNN, defined as the ratio of the number of correct k nearest neighbors to the returned k results. In addition, we also use a weighted precision metric which assigns a higher score to the higher-order nearest neighbors. It is calculated as follows.

$$Weighted\ Precision = \left(\sum_{j \in C} 1/O_j\right) \Big/ \left(\sum_{i=1}^{k} 1/i\right),$$

where C is the set of correct k nearest neighbors among the returned k results and O_j is the order of the neighbor.

Fig. 18 shows that the precision of TkNN reaches 88 percent and the weighted precision 96 percent. Therefore, even though TkNN provides only approximate kNN results, it does so with high accuracy, and in the vast majority of cases the exact NN points are returned.

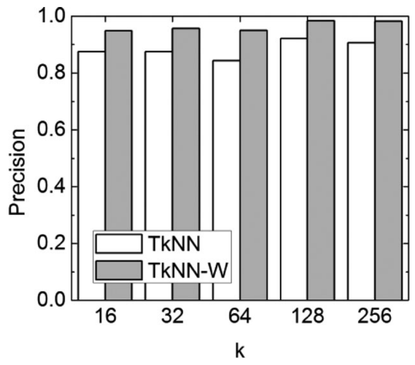


Fig. 18. Precision of TkNN.

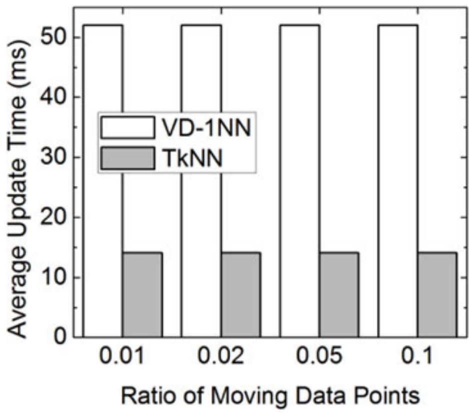


Fig. 19. Average incremental update time per moving data point (1 million points data set).

8.4 Incremental Update Time

For TkNN, when a data point moves, the point and d edges connected to it are changed and re-encoded/encrypted. For VD-1NN, when a data point moves, d Voronoi points and $2d$ Voronoi edges connected to the Voronoi points are changed and re-encoded/encrypted.

Incremental update time has two components: reconstruction time of Delaunay triangulations or Voronoi diagrams, and re-encoding/encryption time of changed points and edges. The reconstruction time is short compared to re-encoding/encryption time.

In Fig. 19, the average per-point incremental update time of TkNN is about three times faster than VD-1NN. The average incremental update time of TkNN is about 15 ms which is quite affordable in practice.

9 Conclusion

In this paper, we proposed two schemes to support secure k nearest neighbor query processing: VD-kNN which is based on Voronoi diagrams, and TkNN which relies on Delaunay triangulations. They both use mutable order-preserving encoding (mOPE) as building block. VD-kNN provides exact results, but its performance overhead may be high. TkNN only offers approximate NN results, but with better performance. In addition, the accuracy of TkNN is very close to that of the exact method. In future work, we plan to investigate more complex secure evaluation functions on ciphertexts, such as skyline queries. We will also research formal security protection guarantees against the client, to prevent it from learning anything other than the received k query results.

Acknowledgments

The work reported in this paper has been partially funded by US NSF awards CNS-1111512 and CNS-0964294. Hyo-Sang Lim has been supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2012R1A1A1042875).

REFERENCES

[1] M. de Berg et al., *Computational Geometry*. Springer.

[2] W.K. Wong, D.W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN Computation on Encrypted Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, 2009.

[3] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing Private Queries over Untrusted Data Cloud through Privacy Homomorphism," *Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE)*, 2011.

[4] H. Xu, S. Guo, and K. Chen, "Building Confidential and Efficient Query Services in the Cloud with RASP Data Perturbation," *IEEE Trans. Knowledge and Data Eng.*, vol. 26, no. 2, pp. 322-335, Feb. 2014.

[5] B. Yao, F. Li, and X. Xiao, "Secure Nearest Neighbor Revisited," *Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE)*, 2014.

[6] R. Ada Popa, F.H. Li, and N. Zeldovich, "An Ideal-Security Protocol for Order-Preserving Encoding," *Proc. IEEE Symp. Security and Privacy*, 2013.

[7] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private Queries in Location Based Services: Anonymizers Are Not Necessary," *Proc. ACM SIGMOD Int'l Conf. Management*, 2008.

[8] G. Ghinita, P. Kalnis, M. Kantarcioglu, and E. Bertino, "A Hybrid Technique for Private Location-Based Queries with Database Protection," *Proc. Advances in Spatial and Temporal Databases*, 2009.

[9] G. Ghinita, P. Kalnis, M. Kantarcioglu, and E. Bertino, "Approximate and Exact Hybrid Algorithms for Private Nearest-Neighbor Queries with Database Protection," *J. Geoinformatica*, vol. 15, pp. 699-726, 2011.

[10] A. Khoshgozaran and C. Shahabi, "Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy," *Proc. 10th Int'l Conf. Advances in Spatial and Temporal Databases*, 2007.

[11] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order Preserving Symmetric Encryption," *Proc. 28th Ann. Int'l Conf. Advances in Cryptology: The Theory and Applications of Cryptographic Techniques (EuroCrypt'09)*, 2009.

[12] A. Boldyreva, N. Chenette, and A. O'Neill, "Order_pPreserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," *Proc. 31st Ann. Conf. Advances in Cryptology (CRYPTO)*, 2011.

[13] J. Louis and L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *ACM Comm.*, vol. 18, pp. 509-517, 1975.

[14] R. Thomas, "Voronoi Diagrams Oover Dynamic Scenes," *Discrete Applied Math.*, vol. 43, pp. 243-259, 1993.

[15] http://www.qhull.org/, 2014.

[16] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through through Spatial and Temporal Cloaking," *Proc. First Int'l Conf. Mobile Systems, Applications and Services*, 2003.

[17] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: a A Personalized Anonymization Model," *Proc. IEEE 25th IEEE Int'l Conf. Distributed Computing Systems*, 2005.

[18] M.F. Mokbel, C.Y. Chow, and W.G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," *Proc. 32nd Int'l Conf. Very Large Data Bases*, 2006.

[19] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preserving Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE Trans. Knowledge and Data Eng.*, vol. 19, no. 12, pp. 1719-1733, Dec. 2007.

[20] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order Preserving Encryption for Numeric Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, 2004.

[21] D.-T. Lee, "On k-Nearest Neighbor Voronoi Diagrams in the Plane," *IEEE Trans. Computers*, vol. C-31, no. 6, pp. 478-487, June 1982.

[22] P.K. Agarwal, M. De Berg, J. Matousek, and O. Schwarzkopf, "Constructing Levels in Arrangements and Higher Order Voronoi Diagrams," *SIAM J. Computing*, vol. 27, pp. 654-667, 1998.

Sunoh Choi received the master's and bachelor's degrees in computer science from Korea University. He is currently working toward the PhD degree in the Deparment of Computer Science at Purdue University. His research interests include privacy-preserving query processing and authenticated query processing.

Gabriel Ghinita is currently an assistant professor in the Department of Computer Science, University of Massachusetts, Boston. His research interests include privacy-preserving transformation of microdata, private queries in location based services and privacy-preserving sharing of sensitive data sets. He serves as reviewer for top journals and conferences such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *VLDBJ*, *VLDB*, *WWW*, *ICDE and ACM SIGSPATIAL GIS*.

Hyo-Sang Lim received the PhD degree from Korea Advanced Institue of Science and Technology. He is currently an assistant professor in the Department of Computer and Telecommunications Engineering, Yonsei University, Wonju, Korea. His research interests include database security and data stream processing technology. He was a research associate at Purdue University.

Elisa Bertino is currently a professor of computer science at Purdue University, and serves as the research director of the Center for Education and Research in Information Assurance and Security (CERIAS) and the director of Cyber Center (Discovery Park). Previously, she was a faculty member and the department head in the Department of Computer Science and Communication at the University of Milan. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She served as editor in chief of the *VLDB Journal*, editorial board member of *ACM TISSEC* and *IEEE Transactions on Dependable and Secure Computing*, and is the editor in chief of *IEEE Transactions on Dependable and Secure Computing* starting since January 2014. She co-authored the book *Identity Management—Concepts*, *Technologies*, *and Systems*. She received the 2002 IEEE Computer Society Technical Achievement Award for outstanding contributions to database systems and database security and advanced data management systems and the 2005 IEEE Computer Society Tsutomu Kanai Award for pioneering and innovative research contributions to secure distributed systems. She is a fellow of the IEEE and ACM.