Secure kNN Query Processing in Untrusted Cloud Environments

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 datasets 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 dataset 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 datasets. We present an extensive performance evaluation of our techniques to illustrate their viability in practice.

EXISTING SYSTEM:

Query processing that preserves both the data privacy of the owner and the query privacy of the client is a new research problem. It shows increasing importance as cloud computing drives more businesses to outsource their data and querying services. However, most existing studies, including those on data outsourcing, address the data privacy and query privacy separately and cannot be applied to this problem.

PROPOSED SYSTEM:

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), which guarantees indistinguishability under ordered chosen-plaintext attack (IND-OCPA). We also provide performance optimizations to decrease the computational cost inherent to processing on encrypted data, and we consider the case of incrementally updating datasets. Inspired by previous work in that brought together encryption and geometric data structures that enable efficient NN query processing, we investigate the use of Voronoi diagrams and Delaunay triangulations to solve the problem of secure outsourced kNN queries. We emphasize that previous work assumed that the contents of the Voronoi diagrams is available to the cloud provider in plaintext, whereas in our case the processing is performed entirely on cipher texts, which is a far more challenging problem.

PROBLEM STATEMENT:
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 geotagged datasets 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 $k$ NN 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 that return many false positives in addition to the actual results, are not desirable.

**SCOPE:**

This is a very challenging task, as conventional encryption does not support processing on top of cipher texts, 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 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 dataset partitioning techniques, but return a large number of false positives, which is not desirable due to the financial considerations outlined earlier.
Feature Enhancements:

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.

MODULE DESCRIPTION:

**Number of Modules**

After careful analysis the system has been identified to have the following modules:
1. **Spatial Database Module**

Spatial database is a database that is optimized to store and query data that represents objects defined in a geometric space. Most spatial databases allow representing simple geometric objects such as points, lines and polygons. Some spatial databases handle more complex structures such as 3D objects, topological coverages, linear networks, and TINs. While typical databases are designed to manage various numeric and character types of data, additional functionality needs to be added for databases to process spatial data types efficiently.

2. **Location Privacy Module**

As mentioned previously, the dataset 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 $k$NN queries, but will also try to infer the 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 dis-close to the server the encryption keys.

3. Database Outsourcing Module

The server receives the dataset 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. The server receives $k$NN 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. 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.

4. Voronoi Diagram-Based K Nearest Neighbor (KNN) Module

Voronoi Diagram

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 Figure 2. Denote the Euclidean distance between two points $p$ and $q$ by $d(p,q)$, 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 $p$ if and only if $p$ is the 1NN of $q$, i.e., for any other point $p'$ it holds that $\text{dist}(q, p) < \text{dist}(q, p')$ [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.

Fig 2. Voronoi diagram

Data Owners sends to Server the encoded Voronoi cell vertices coordinates, MBR boundaries for each cell, encoded right-handside $S$, and encrypted $S$, for each cell edge. Clients sends its encoded query point to the Server. 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 $S$, of the two edges to the Client. Client computes the left-handside $L$, encodes it and sends it to the Server. Server finds the Voronoi cell enclosing the query point and returns result to Client.

K Nearest Neighbor (KNN)
To support secure $k$NN queries, where $k$ is fixed for all querying users, we could extend the VD-1NN method from by generating order-$k$ Voronoi diagrams. However, this method, which we call VD-$k$NN, has several serious drawbacks:

1. The complexity of generating order-$k$ Voronoi diagrams is either $O(k \cdot n \log n)$ or $(k(n-k)\log n + n \log n)$, depending on the approach used. This is significantly higher than $O(n \log n)$ 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 $k << n$. 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-$k$NN, we first introduce a secure distance comparison method (SDCM). Next, in we devise Basic $k$NN (B$k$NN), a protocol that uses SDCM as building block, and answers $k$NN queries using repetitive comparisons among pairs of data points. B$k$NN is just an auxiliary scheme, very expensive in itself, but it represents the starting point for Triangulation $k$NN (T$k$NN), presented. T$k$NN builds on the B$k$NN concept and returns exact results for $k = 1$. For $k > 1$, it is an approximate method that provides high-precision $k$NN results with significantly lower costs.

**SOFTWARE REQUIREMENTS:**

- Operating System: Windows
- Technology: Java and J2EE
- Web Technologies: HTML, JavaScript, CSS
- IDE: My Eclipse
- Web Server: Tomcat
- Tool kit: Android Phone
- Database: My SQL
- Java Version: J2SDK1.5
HARDWARE REQUIREMENTS:

Hardware : Pentium
Speed : 1.1 GHz
RAM : 1GB
Hard Disk : 20 GB
Floppy Drive : 1.44 MB
Key Board : Standard Windows Keyboard
Mouse : Two or Three Button Mouse
Monitor : SVGA