Kamel *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2017) 6:5
DOI 10.1186/s13677-017-0077-0

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Dynamic spatial index for efficient query processing on the cloud

Ibrahim Kamel[2], Ayesha M. Talha[1*] and Zaher Al Aghbari[3]

Abstract

Data owners with large volumes of data can outsource spatial databases by taking advantage of the cost-effective cloud computing model with attractive on-demand features such as scalability and high computing power. Data confidentiality in outsourced databases is a key requirement and therefore, untrusted third-party service providers in the cloud should not be able to view or manipulate the data. This paper proposes DISC (Dynamic Index for Spatial data on the Cloud), a secure retrieval scheme to answer range queries over encrypted databases at the Cloud Service Provider. The dynamic spatial index is also able to support dynamic updates on the outsourced data at the cloud server. To be able to support secure query processing and updates on the Cloud, spatial transformation is applied to the data and the spatial index is encrypted using Order-Preserving Encryption. With transformation and cryptography techniques, DISC achieves a balance between efficient query execution and data confidentiality in a cloud environment. Additionally, a more secure scheme, DISC*, is proposed to balance the trade-off between query results returned and security provided. The security analysis section studies the various attacks handled by DISC. The experimental study demonstrates that the proposed scheme achieves a lower communication cost in comparison to existing cloud retrieval schemes.

Keywords: Data outsourcing, Spatial queries, Encryption, Dynamic updates

Introduction

With increase in spatial data, data owners require the services of untrusted remote servers that can store huge amount of data and allow fast access to outsourced data. Cloud computing allows a third-party service provider to manage the data and provide services directly to the end-user. Cloud computing provides attractive features such as scalability, cost-effectiveness and high-computing power. Popular examples of cloud-based services include Google Maps and Amazon EC2. In recent years, mobile devices and navigational systems have become common and this has created the need for location-based services (LBSs). Mobile users issue queries from devices with limited storage and computational resources. Spatial range queries performed at the cloud server must be completed in real-time and only relevant results should be returned to the user.

*Correspondence: atalha@sharjah.ac.ae
[1]Research Institute of Sciences & Engineering, University of Sharjah, Sharjah, United Arab Emirates
Full list of author information is available at the end of the article

The cloud model consists of three entities, namely the Data Owner (DO), Cloud Service Provider (CSP) and the Trusted User (TU). The DO outsources the spatial data and index for fast retrieval to the CSP, while the TU issues encrypted queries to the CSP. The query is processed directly on the encrypted data at the CSP without additional communication overhead between the TU and CSP. The relevant results are returned in a secure format to the TU, where decryption reveals the actual data points.

The fact that the data is controlled by an untrusted third-party [1–4], raises security concerns about data confidentiality. Data confidentiality requires that data is not disclosed to untrusted servers, as they could release sensitive information to competitors. Therefore, when outsourcing spatial databases in the cloud, the data should not be visible to the service provider or adversaries. The CSP provides services to multiple DOs and hence cannot be trusted. Another prime concern is efficient query execution, which can be resolved by using a spatial indexing structure for fast data access.

To achieve total confidentiality, the naïve solution is to encrypt the whole dataset and send only the encrypted

data to the cloud service provider. During the query phase, the TU retrieves the entire encrypted data from the server, decrypts it and searches for the required data points. This makes ideal security achievable, but it is clearly not practical in real-time applications as the resulting data communication cost would be high, especially if only a small portion of the data is queried. Furthermore, the high processing power of the cloud environment would not be utilized in this case.

This paper focuses on the development of an efficient retrieval technique that can be executed on encrypted data at the cloud service provider. Several specialized retrieval techniques have been proposed to answer queries on encrypted data. Researchers have adopted two different approaches to resolve this issue. The first approach is to use spatial transformation techniques to obfuscate the original data prior to sending it to the CSP [5–8]. The other approach is to use cryptographic techniques [9–12] to protect the confidentiality of the outsourced data. To provide a double layer of security, we apply both transformation and encryption techniques on the outsourced data.

In cryptographic approaches, some existing works use the Advanced Encryption Standard (AES) [13], which can be used only to answer exact-match queries. While others use the Order-Preserving Encryption (OPE) technique [10]. OPE is a class of cryptographic techniques that preserves the relative order of the encrypted objects, executing range queries on the encrypted data directly at the server without having to decrypt it. Although the security of OPE falls short of the targeted industry standards, there is a lot of interest by researchers in OPE as it allows comparisons directly on the encrypted data. Since this paper supports queries on encrypted data at the CSP, we employ the OPE technique for the index and the secure AES for the spatial data points.

Security and query processing efficiency are important when designing schemes applicable in a cloud environment. In this paper, the **DISC** (**D**ynamic **I**ndex for **S**patial data on the **C**loud) scheme is proposed for answering spatial range queries on encrypted databases at the CSP. In DISC, a combination of transformation and encryption is used to provide a fair balance between data confidentiality and query execution. Another key advantage of the proposed approach is that there is no need for the DO to install an additional trusted front-end i.e. a tamper-resistant device [14–16], between the user and cloud service provider during query processing.

Briefly, the DISC retrieval scheme works as follows. The DO transforms the spatial data points and indexes them. The index is encrypted using the OPE technique for fast data access at the CSP, while the spatial data points are encrypted using the more secure AES. Next, the indexed data is outsourced and encrypted queries are processed entirely on the encrypted data at the CSP. Encrypted query results are returned to the TU, where they are decrypted using the key provided by the DO and then false positives are filtered to obtain the actual query results. With DISC, it is also possible to perform updates dynamically on the encrypted index at the CSP, where the DO issues an encrypted update request to be partially carried out at the CSP.

Contributions:

- A retrieval scheme is proposed to answer queries over encrypted data at the CSP, ensuring confidentiality of data outsourced by the DO.
- DISC provides efficient communication between the TU and CSP (one round of communication).
- The proposed scheme supports dynamic updates such as insert, delete and modify on the encrypted data at the CSP.
- An enhanced and secure scheme, DISC*, is proposed as well to further obscure the data at the CSP.
- Furthermore, a comprehensive security analysis against known attacks used in the literature is provided.
- Simulation experiments were conducted on real data to evaluate the performance of DISC and the proposed scheme is compared to an existing cryptographic transformation scheme in terms of communication cost.

The remainder of the paper is organized as follows. The next section surveys some existing work in this area. "Problem statement" section briefly discusses the cloud system model. Then, "DISC: a retrieval scheme" section describes DISC used in our approach. "Indexing spatial data" section discusses the indexing scheme in detail, and "Answering spatial range queries at CSP" section presents the spatial query phase, i.e., processing encrypted queries at the service provider. "Dynamic updates at CSP" section focuses on dynamic updates on DISC at the CSP. "Secure scheme: DISC*" section proposes a secure and enhanced DISC* scheme, which discusses the trade-offs between efficiency and security. Next, "Security analysis" section provides a security analysis on the transformation and encryption technique incorporated in DISC. Lastly, experiments are conducted on two real spatial datasets, and the results with comparative and evaluative measures are offered in "Experimental evaluation" section, followed by conclusions in "Conclusions" section.

Related work

The issue of secure outsourcing of data has been addressed in several recent papers. Hacigümüs et al. [17] were the first to formally propose database services

outsourcing to a third-party service provider. The typical model comprises of the data owner, service provider and authorized users. The data owner outsources data and query services to the untrusted service provider. In Location-based services, trusted users issue queries to the service provider. Despite the fact that the cloud environment provides on-demand services along with scalable storage and extensive computational power, it poses data security and privacy challenges. The primary goal is to secure data by encrypting it and allowing queries on encrypted spatial data at the cloud service provider. Batten et al. [18] propose a cloud storage model, which comprises of cloud customers, cloud service provider and cloud service operator. The customer rents storage from the service provider, which owns the cloud resources and maximizes storage resource utilization between numerous customers, and the management of data storage is taken care of by the service operator.

One of the existing work by Yiu et al. [5] proposed several transformation as well as a cryptographic scheme for outsourcing spatial databases. In data transformation schemes, the data points are relocated in the space based on an equation. In these schemes, the attacker can gain knowledge about nearby points with limited background information. The encryption based scheme inherits the security of AES, where the DO stores the encrypted R^*-tree index in the cloud. To process a query, the data owner retrieve encrypted nodes of the R^*-tree level by level, decrypts them and select intersected nodes. They are able to hide the spatial data from the CSP, but cannot provide range query processing at the server. Thus, answering queries requires multiple rounds of data communication between the server and user.

Similarly, Kim et al. [19, 20] designed a transformation scheme based on the Hilbert curve. The space is transformed by clustering the data points and reducing the dimensionality to $1 - D$. The data is encrypted using the conventional AES and stored at the server. To process a range query, the entire encrypted file has to be sent to the user to search for relevant records. The user then requests for required data, hence requiring multiple rounds of communication.

Both [5] and [19] result in a high communication cost between the service provider and user owing to multiple rounds of data exchange. To overcome this, Talha et al. [21] present a dual transformation approach that allows query processing on encrypted data at the server. The original spatial data points coordinates are hidden using the Hilbert space-filling curve and grouped in packets. The encrypted data is stored at the server. The user sends encrypted spatial range queries and the results are decrypted by the user. This lowers the communication cost as it is limited to a single round.

However, the above-mentioned transformation-based and cryptographic approaches are designed for static data and therefore cannot handle dynamic updates. In the event of any insertions, deletions or modifications to the data, the dataset would have to be indexed and encrypted again before outsourcing it to the CSP. Whereas, the DISC scheme proposed in this work can handle dynamic updates from the DO.

To support range queries over large datasets efficiently, Damiani et al. [14] build a tree-index and store the nodes of a B+ tree as encrypted blocks. To process a range selection, the user repeatedly retrieves a node, starting with the root, and decrypts it to identify the child node to traverse to. Upon reaching the target leaf node, he then follows the sibling pointers in the leaf level.

On the other hand, Hore et al. [16] partition the data into a set of buckets. The data owner builds indices for buckets which are not hierarchically structured, so the index search must be linear in the number of buckets. Increasing the bucket size improves privacy but reduces efficiency, since the indices must be locally stored, and index searching is linear.

Data privacy and query integrity is assured by Ku et al. [22] for outsourced databases. The points are encrypted with a symmetric key and indexed based on the Hilbert curve. A probabilistic approach is applied to a portion of data encrypted with a different key to ensure reliability of query results.

Recently, Wang et al. [9] proposed a framework that provides both security and efficiency. They use an $\hat{R}$-tree index that is encrypted using Asymmetric Scalar-product Preserving Encryption (ASPE) scheme. However, it does not provide a privacy guarantee, nor does it provide confidential query processing because it leaks information on the ordering of the MBR of the leaf nodes and requires result post-processing as it introduces false positives.

Wong et al. [23] propose a scheme for secure kNN (k-Nearest Neighbors) queries on encrypted data. Distance comparisons between an encrypted query and data points are achieved using ASPE, with query points and data points being encrypted differently. Given two data points and a query point, the cloud can determine which data point is closer to the query point. Lu et al. [24] proposed an outsourced range query scheme using predicate encryption. This scheme provides provable security and can achieve logarithmic-time search since it orders the encrypted data points. However, it is not very practical as it only supports 1-D data.

In contrast to the approaches mentioned above, Hu et al. [3] propose that the DO outsources decryption keys to the server, and provides users with encrypted data. In a query process, a user first sends encrypted data and query to the cloud, then the cloud uses the decryption key to decrypt data and query, and return the result. The novelty

of this paper is that they use homomorphic encryption to ensure that the cloud cannot learn anything during the query process. However, fully homomorphic encryption [25] is highly impractical in practice.

To overcome the limitations of the existing schemes, our approach is modeled to achieve a balance between data confidentiality and efficient query processing i.e single round of communication. To allow range queries over encrypted numeric data, Agrawal et al. [10] propose the Order-Preserving Encryption (OPE), where a plaintext is converted to ciphertext through order-preserving mapping functions This scheme is secure against ciphertext-only attacks and fails when the data distribution or the plaintext are known, as it is then straightforward to associate an encrypted record with its plaintext counterpart.

Yiu et al. [26] utilize OPE in a metric-preserving transformation, where each data point is assigned to its closest pivot. The index reveals information about the space by not hiding the number of points per pivot. The query is evaluated with regards to the original space but the query point is mapped to a pivot point. Furthermore, adaptive chosen-plaintext attacks can reveal the secret key and help identify dense areas in the space.

Problem statement

System model

In this paper, DISC is proposed, which is a Dynamic encrypted Index for Spatial data on the Cloud. The DISC cloud system model is shown in Fig. 1, and it comprises of three distinct entities. Namely the Data Owner (**DO**), Cloud Service Provider (**CSP**) and the Trusted Users (**TU**). Briefly, the process works as follows, DO outsources spatial data to CSP which is queried by TUs. The database is transformed, indexed and encrypted before it is stored on the cloud. The keys are sent to the users by the DO. The CSP processes encrypted range queries and modifies the dynamic index with encrypted updates sent by the DO. TUs issue encrypted range queries to the CSP and obtain encrypted query results in a single round. Lastly, the TU decrypts the results returned.

Notations

The DISC retrieval technique can be generalized to several domains. Given a spatial dataset, D, at the DO with s two-dimensional spatial data points, $D = (d_1, d_2, \ldots, d_s)$, represents physical locations in the space. The domain of each dataset is normalized to the unit square $[0, 1]^2$ in $2-$D Euclidean space, $\mathbb{E}^2$. Table 1 lists the notations that will be used throughout the paper.

Preliminaries

Hilbert Transformation: Space-filling curves are used to map multidimensional data to one-dimensional data where they pass through every partition in a given space without any intersection with itself. The mapping has to be *distance-preserving* such that points closer in space are mapped onto nearby points on the curve. One of the widely used curves is the Hilbert curve due to its superior clustering properties [27]. Spatial points are traversed exactly once and indexed based on the order in which they are visited by the curve. We begin by representing the area of an N x N grid as a single cell. We iterate, and in the i^{th} iteration, $i = 0, \ldots, n-1$ (for $N = 2^n$), we partition the area of the N x N grid into $2^i.2^i$ blocks. Next, the points are assigned Hilbert cell values based on the curve. The grid is spanned according to the curve using the Hilbert Space Key (HSK) [22]. The HSK = $\{x_0, y_0, \theta, g\}$, where (x_0, y_0) is the curve's starting point, θ is the curve's orientation and g is the curve granularity. Based on the HSK, it is possible for two or more points to have the same cell value in the curve.

Order-preserving encryption: It is a type of homomorphic encryption scheme. Fully Homomorphic

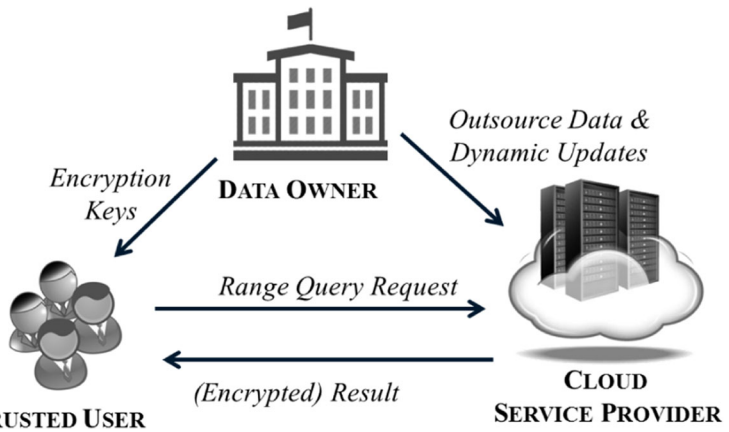


Fig. 1 Proposed system model for spatial data outsourcing

Table 1 List of Notations used

Notation	Description
DO	Data owner
CSP	Cloud service provider
TU	Trusted user
D	2D spatial data point set
F_{HSK}	Hilbert transformation function
H	Hilbert cell value
K_{OPE}	Order-preserving encryption key
K_{AES}	Advanced encryption standard key
LHV	Largest Hilbert value
nodeCapacity(n_L)	Node Capacity of a leaf node
QR	Query Hilbert value set
R	Query result set

Encryption [25] would be ideal as it allows query execution on encrypted data and is completely secure, however, its high computation cost makes it prohibitive in practice. Therefore, we employ the order-preserving encryption (OPE) scheme proposed by Boldyreva et al. [28], which allows range queries to be evaluated on the numeric data. The order of plaintext data is preserved in the ciphertext domain through a random mapping without revealing the data itself. For an encryption key K_{OPE}, if $x < y$, then $E_{K_{OPE}}(x) < E_{K_{OPE}}(y)$. OPE allows efficient access to the encrypted data by maintaining a set of indexes for simple comparisons, such as relational and logical operations between values in a spatial range query request. For OPE applied to integer values, the output set is bigger than the input set meaning that no two values will have the same encrypted value.

Hilbert R-tree: It is a hybrid structure based on the R-tree and B^+-tree that utilizes the Hilbert space-filling curve. The nodes in the tree are sorted on the Hilbert value of their rectangle centroid. A defining characteristic of the Hilbert R-tree is that there exists an order of the nodes at each tree level, respecting the Hilbert order of the Minimum Bounding Rectangles (MBRs). Searching procedure is similar to that of R-tree. Each internal node entry consists of the MBR that encloses all the objects in the corresponding subtree, the largest Hilbert value (LHV) of the subtree, and a pointer to the next level. Each leaf node of the Hilbert R-tree stores the Hilbert coordinate of the MBR of each data point stored and has a well-defined set of sibling nodes. Thus, the Hilbert R-tree can keep spatial data ordering according to Hilbert value when it is updated dynamically. Dynamic Hilbert R-trees achieve high degree of space utilization and good response time, while other R-tree variants have no control over space utilization. The Hilbert R-Tree allows around 28% saving in response time compared with existing structures. Moreover, the R-tree index is non-deterministic as the tree structure differs based on the sequence of insertions, whereas the Hilbert R-tree does not suffer from this.

System model data flows

With DISC, it is possible to process user queries in a secure and efficient manner. The data index is built and encrypted at the DO and stored at the CSP, while the TU issues encrypted spatial queries. There are four different data flows in the model:

1. **DO to CSP:** Outsources the encrypted spatial index to the CSP as well as the encrypted dynamic updates. The Hilbert curve is used to transform the location of spatial data points. Following the formal definition of the Hilbert space-filling curve in [27], H_{2D} with granularity, $g \geq 1$ is used for a two-dimensional space. This implies that any point in the $2-$D set, D, is mapped to a $1-$dimensional integer set $[0, \ldots, 2^{2g} - 1]$ using the Hilbert transformation function $F_{HSK} = f(d)$ based on the HSK (cf. "Preliminaries" section). Next, the DO forms the Hilbert R-tree. The encryption function $E_{K_{OPE}}$ is applied to the internal nodes in the index and $E_{K_{AES}}$ is used for the spatial points. Encrypted updates, $U = E_{K_{OPE}}(u)$, are sent individually to the CSP as required.
2. **DO to TU:** The transformation key, HSK, and encryption keys, K_{OPE} and K_{AES}, are sent by the DO. Communication channel between the DO and TUs is assumed to be secured under existing security protocols such as SSL.
3. **TU to CSP:** TU converts the range query request to a set of $1-$D Hilbert indices, $QR = (q_1, \ldots, q_m)$, using F_{HSK}. This integer set is encrypted using $E_{K_{OPE}}$ and sent to the CSP to be executed over the encrypted index.
4. **CSP to TU:** The encrypted query is processed entirely at the CSP and the resulting data point set, $R = (E_{K_{AES}}(d_1), \ldots, E_{K_{AES}}(d_r))$, encrypted using AES, is returned to the TU where it is decrypted using $E_{K_{AES}}$ and filtered to remove false positives.

DISC: a retrieval scheme

The data owner stores data on remote servers that provide querying services to trusted users. Data confidentiality in an outsourcing retrieval scheme is key as data is managed by an untrusted party i.e. CSP. A common mechanism to ensure secure data outsourcing is encryption, so that the CSP can learn as little information about the plain data as possible. In this work, DISC is proposed to process encrypted queries directly on encrypted data,

in order to keep the data and query results confidential from adversaries. It is required to have a balance between data confidentiality and query execution. A spatial indexing structure is built to provide fast data access and in turn, efficient query processing.

Indexing spatial data

Spatial index is a data structure used to improve the efficiency of spatial data operations on data objects. Common spatial index methods include the R-tree and its variants. **DISC**, a **D**ynamic encrypted **I**ndex for **S**patial data on the **C**loud, uses the dynamic Hilbert R-tree [29] with regards to the Hilbert transformation used to discretize the data points. The index structure is then encrypted using the Order-Preserving Encryption scheme (OPE [28], while the actual data is encrypted separately with a secure symmetric encryption method, AES.

The indexing scheme used by the DO is illustrated in Figs. 2 and 3. The index construction process at the DO is initiated with static 2−D spatial data points in the space. Next, each spatial point in the space is assigned a Hilbert cell value in the grid. In the example in Fig. 2, the space is partitioned into 64 cells by applying a Hilbert curve of granularity 3 with starting point $(0, 0)$.

Next, the Hilbert R-tree index is constructed bottom-up based on the ascending Hilbert cell values. The DO then encrypts the Hilbert R-tree internal nodes and Hilbert values in leaf nodes using OPE, while the data points in the leaf nodes are encrypted using AES before being outsourced to the CSP, since it is not a trusted entity. This protects the sensitive data from being leaked by the CSP (i.e. to third-party vendors). The CSP does not have the ability to decrypt the encrypted data without the secret keys. The DO sends the transformation and encryption keys only to TUs.

The encrypted index and data points at the CSP are shown in Fig. 3. The leaf and non-leaf nodes are encrypted using OPE [enclosed in a red dashed box] to support range queries and dynamic updates, while the data points in the leaf nodes [enclosed in the blue solid box] are encrypted individually using the secure and traditional encryption scheme, AES [13]. The parent-children relationships (i.e. pointers) in the index are not encrypted in order to allow efficient query search. Additionally, the DO sends the OPE key and the AES key to the TUs so that they can send encrypted queries to the CSP and decrypt the returned query results.

Algorithm 1 lists the pseudo-code for the dynamic index construction process. In the first loop (Lines 1–4), each spatial data point d_i in D is normalized to $[0, 1]^2$ and then its Hilbert cell value is computed using the Hilbert transformation function, F_{HSK}, based on the Hilbert Space Key. In Line 5, the resulting Hilbert cell values for all data points are stored in C and sorted before building the Hilbert R-tree Index (Line 6). Lines 7–12: all the nodes in the tree are encrypted. If it is an internal node, MBR and LHVs are encrypted using the OPE scheme to allow for comparisons on the encrypted data. While, data points in the leaf nodes are encrypted using AES. Lastly

Algorithm 1 Encrypted Index Construction (by **DO**)

Input:
Spatial Data Points, $D = (d_1, \ldots, d_s)$
Encryption Keys, K_{OPE} and K_{AES}

1: **for all** d in D **do**
2: Normalize d_i
3: Compute Hilbert value, c of d_i using F_{HSK} and add to C
4: **end for**
5: `I = BuildHilbertR-Tree()`
6: **for all** n in I **do**
7: Encrypt MBR and LHV using K_{OPE}
8: **if** n_i is a leaf node **then**
9: Encrypt data points in n_i using K_{AES}
10: **end if**
11: **end for**
12: Send Encrypted Index, I_E, to **CSP**

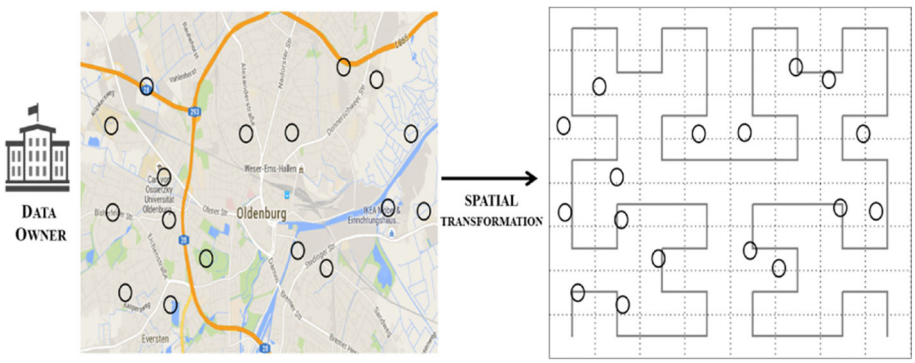


Fig. 2 Spatial data points at the DO mapped to Hilbert cells

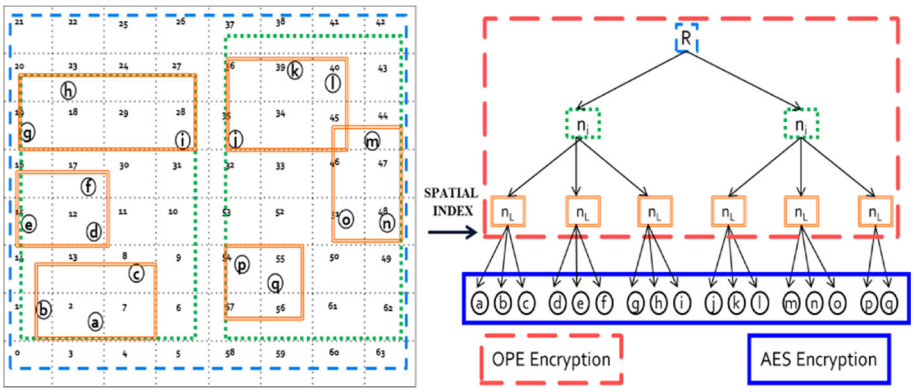


Fig. 3 Index construction process based on transformation

(Line 13), the index is outsourced to the Cloud Service Provider.

Answering spatial range queries at CSP

The key requirement for query processing at the cloud server is fast response time. The encryption scheme in DISC encrypts nodes without modifying the index structure, hence faster than linear search is possible. The DISC retrieval scheme deals with 2−D spatial range queries due to their popularity. In spatial range query algorithms, the index search propagates downwards starting from the root node, considering whether node entries overlap with cells in the query region.

When a query request is initiated by the TU, the rectangular region of the range query, (QW ([(cx_0, cy_0), (cx_1, cy_1)])), is converted to a set of 1-D Hilbert cell values [30] by the TU, which includes cells that may partially or completely overlap with the query region. Since some of these cells only partially overlap with the query region, the set of cells might retrieve irrelevant data points (i.e. false positives) in the query response. Having false positives in the results is a reasonable trade-off for security, given that the ordering information and data points are securely encrypted in DISC. The Hilbert cells in the query set are then encrypted by the TU using the OPE key. The CSP is responsible for processing the encrypted query request over the spatial index.

Figure 4 shows the spatial range query procedure. Queries are formulated in terms of their OPE encrypted Hilbert values and AES encrypted data points corresponding to encrypted Hilbert values are returned. The CSP searches the index created by DISC, performing comparison tests level-by-level, and proceeding to a node's children if and only if the node's LHV is greater than the query value. The CSP thereby obtains all leaves contained in the query region, and then returns data points in leaf nodes to the TU with Hilbert values corresponding to the queried values. We assume that the granularity for the Hilbert curve transformation is high enough, such that each data point is associated with a unique Hilbert value. The CSP cannot query the data points themselves as comparisons cannot be made on data encrypted using AES.

The Hilbert R-tree index improves the search performance as it uses Hilbert cell values to impose a total order on the entries. Algorithm 2 shows the complete spatial range query procedure. Lines 1 − 6 and Line 18 list the process at the TU, while Lines 7 − 17 highlight the query processing procedure at the CSP. First, the rectangular query region, QW ([(cx_0, cy_0), (cx_1, cy_1)]), is converted

Algorithm 2 Spatial Query Processing on Encrypted Data (CSP)

Input:
Query Window QW, [(cx_0, cy_0), (cx_1, cy_1)]
Encryption Keys, K_{OPE} and K_{AES}

1: Q = Hilbert cells contained in QW ▷ At **TU**
2: **for all** $q \in Q$ **do**
3: $QR = QR \cup K_{OPE}(q)$
4: **end for**
5: Send encrypted QR to CSP
6: **for all** n nodes in Index **do** ▷ At **CSP**
7: **if** (n_i is a leaf node **then**
8: Check for intersection between QR and all Hilbert Values in n_i
9: Add corresponding data points to R
10: **else if** LHV(n_i) is greater than QR_i **then**
11: Search recursively in the subtree under n_i
12: **end if**
13: **end for**
14: Return encrypted spatial data points, R, to TU
15: **TU** decrypts R using K_{AES} to obtain actual data objects

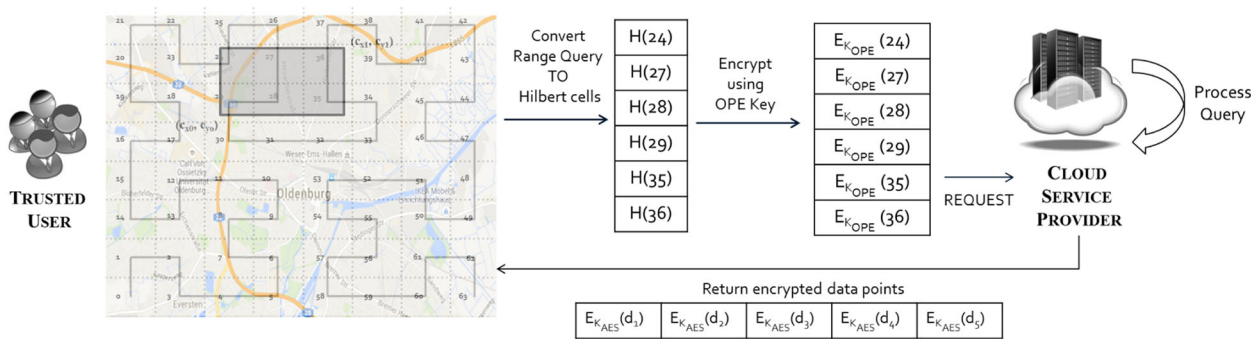


Fig. 4 Query processing at the cloud service provider on encrypted data sent by trusted users

into a set of ascending Hilbert cell values (Line 2) and stored as *QR* at the user end. Next, in Lines 3 – 5, the TU encrypts *QR* using the encryption function K_{OPE}. The encrypted *QR* is sent to the CSP as *QR* along with the encrypted query window *QW*. In Lines 9 – 16, starting from the root, the search descends the tree structure and examines all nodes, *n*, that have Hilbert values less than the queried values, *QR*. If n_i is at the leaf level, each Hilbert value in the leaf node is checked against query Hilbert values in *QR*. The encrypted data points of matched values are returned as the query response, *R*, to the user (Line 17). In Line 18, the TU then decrypts the retrieved data points using the K_{AES} key and generates the actual query response.

Dynamic updates at CSP

Besides processing spatial range queries efficiently at the cloud service provider, DISC allows dynamic updates on encrypted data. DISC takes advantage of the total ordering based on Hilbert transformed values in order to support updates. The proposed scheme is capable of updating an OPE encrypted index at the CSP without revealing the underlying index structure. Dynamic data includes three update operations on the encrypted index: insert, delete and modify. In order to update a spatial data point in DISC, the data owner first needs to issue an update request. Based on the request, the CSP has to locate (i.e. query) which leaf nodes of the index are directly affected. Lastly, all updates applicable to the parent nodes are propagated upwards till the root.

Insert: The new spatial data point to be inserted is sent in an encrypted format by the DO to the CSP. The data point is encryped using AES and the corresponding Hilbert value (H) of the MBR centroid is computed and encrypted using OPE, so that comparison operations can be made. For insertion, the Hilbert R-tree index performs binary search on the total ordering of Hilbert values and these are used as the key value to find the insertion location in the encrypted index, based on simple value comparisons. Starting from the root node, at each level the node with the minimum LHV greater than H of all its sibling nodes is chosen. When a leaf node, n_L, is reached, insertion can be done. If the node capacity of n_L has not exceeded the node capacity, $nodeCapacity_{max}$, the AES encrypted data point is inserted in n_L along with the H value. If the leaf node, n_L, is full, overflow has to be handled by by splitting the leaf node into two and moving half of the ordered entries to a new node. Lastly, the index has to be adjusted such that the LHV values of the parent nodes reflect the newly inserted value. The algorithm at the functional level is listed in Algorithm 3.

Algorithm 3 Insert Encrypted Data in DISC

Input:
Encrypted Data Point, K_{AES}(d) ▷ Sent by **DO**
Encrypted Hilbert Value, K_{OPE}(H)
1: n_L = `FindLeafNode` (K_{OPE} (H)) ▷ At the **CSP**
2: **if** $n_L < nodeCapacity_{max}$ **then**
3: `Insert` (K_{OPE} (H), K_{AES} (d)) in n_L
4: **else**
5: `HandleOverflow` (n_L)
6: **end if**
7: Invoke `AdjustTree()`

Delete: To delete a data point, the Hilbert value of the data point to be deleted is sent in an encrypted format by the DO to the CSP. The Hilbert value (H) is encrypted using OPE, so that comparison operations can be conducted on the index. In the Hilbert R-tree deletion process, the entry with the Hilbert key value (i.e. leaf node, n_L, with H) is removed without visiting multiple paths in the index. A delete update on the leaf node may cause n_L to go under the minimum node capacity, $nodeCapacity_{min}$. In the event of an underflow, sibling nodes can be merged together. A functional-level algorithm is listed in Algorithm 4.

Algorithm 4 Delete Encrypted Data from DISC

Input:
Encrypted Hilbert Value, K_{OPE}(H) ▷ Sent by **DO**
1: n_L = `FindLeafNode` (K_{OPE} (H)) ▷ At the **CSP**
2: `Delete` (K_{OPE} (H)) in n_L
3: **if** $nodeCapacity(n_L) < nodeCapacity_{min}$ **then**
4: `HandleUnderflow` (n_L)
5: **end if**
6: Invoke `AdjustTree` ()

Modify: In order to modify a data point over encrypted DISC, the CSP conducts an insert and a delete operation on the index at the server. Thus, the DO has to send the data point to be deleted and the new data point to be inserted. For the modify operation, the CSP needs to first locate where the point to be deleted lies in the index, then delete it. It is not feasible to insert the data point in the same location as the deleted point as their positions in the indexed tree may differ. Lastly, updates are propagated upwards till the root node of the tree.

Secure scheme: DISC*

The data owner creates encrypted range queries, and sends them to the CSP, where they are processed over the encrypted index. The CSP performs the search starting from the root and proceeding to a node's children if the query value is less than the largest Hilbert value. In DISC, the lowest level of the index comprises of leaf nodes which stores the data points in a particular Hilbert cell. Thus, the CSP returns the encrypted data points matching the query Hilbert values (in OPE) to the TU. The CSP is unable to query the spatial points themselves as they are securely encrypted using AES. DISC is a retrieval scheme which is capable of returning the exact set of encrypted points. Given that the LHVs in the leaf nodes are encrypted using OPE, the ordering information of points within the Hilbert transformation can leak over time. Given a series of spatial queries over a period of time, the attacker can rebuild the order of data points in the transformed space based on the intersection of query results. The DISC index can be fine-tuned to accomodate the trade-off between security and false positives in the query result. Therefore, we propose DISC*, a more secure retrieval scheme that protects the ordering of spatial points in the leaf nodes of the index. As displayed in the index in Fig. 5, the leaf node level along with the data points is encrypted in AES [enclosed in blue solid box]. This may induce a number of false positives in the query results returned and thus filtering will have to be performed at the user-end. This is a reasonable trade-off, given the additional layer of security at the CSP.

Secure spatial range queries at CSP

The query processing is still done entirely at the CSP and follows the same procedure as listed in Algorithm 2 (cf. "Answering spatial range queries at CSP" section). The query processing time is reduced in DISC*, as there is no need to compare Hilbert values in the leaf (Line 8). The search space is restricted to the non-leaf nodes in the index and this helps locate the leaf node whose Hilbert value is closest to the query value. In Line 9, instead of returning relevant data points, the whole leaf node is returned. This will induce some additional points that are not part of the query result, and can be filtered by the TU after decryption in a post-processing step.

Secure dynamic updates

In DISC, it is possible to update the index at the CSP given the update operation and encrypted data from the DO. But in DISC*, given that the entire leaf node is encrypted using AES and not just the data points, it is not possible to update directly at the CSP. Thus, secure updates require a single round of communication between the DO and CSP. The complete procedure is illustrated in Fig. 6. First, The update request is initiated by the DO. The DO sends the OPE encrypted Hilbert value corresponding to the update operation. Second, the CSP searches the

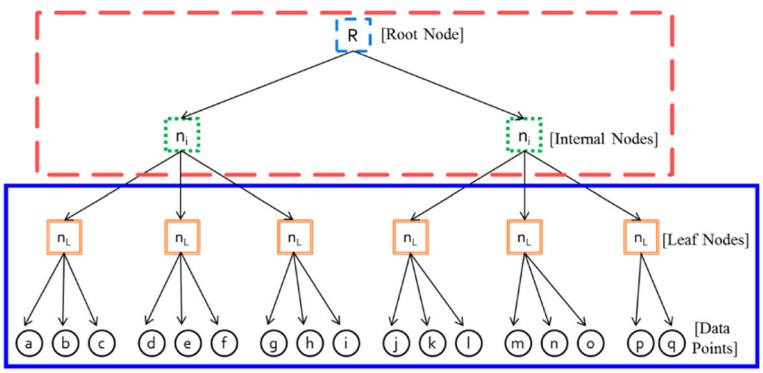


Fig. 5 Spatial data index construction by the data owner

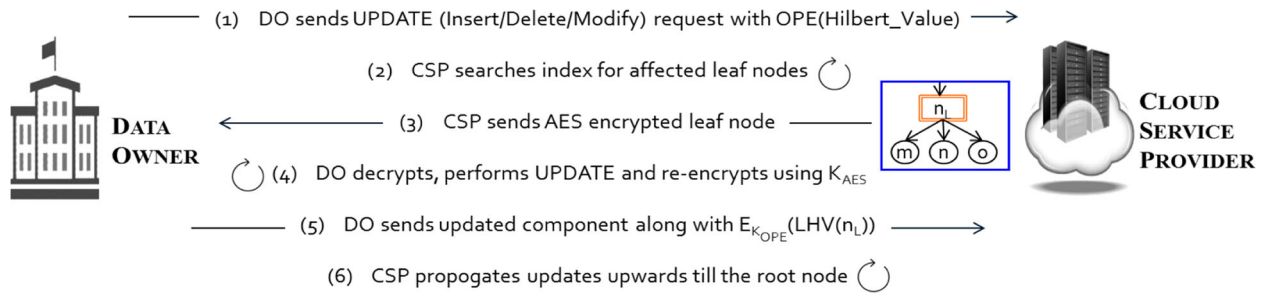


Fig. 6 DISC* scheme procedure for dynamic updates between data owner and cloud service provider

encrypted index and locates the leaf node corresponding to the Hilbert value. Third, the leaf node is sent from the CSP to the DO. Fourth, in order to perform the update, the DO has to decrypt the AES encrypted node and perform the respective update. Next, The DO encrypts the node using AES and sends it to the CSP along with the LHV of the node encrypted using OPE. Lastly, the CSP replaces the updated node in the index and propogates updates upwards till the root based on the updated Hilbert value. Even though partial update is performed at the DO, it is clear that hiding ordering information in DISC* enhances data confidentiality.

Security analysis

The key requirements of a secure data outsourcing scheme demand that: 1) data confidentiality at the server and 2) queries are efficiently processed by the CSP and results are returned to the user without any alteration. As mentioned previously, the TUs are trusted by the DO and hence have been provided with both the HSK, as well as the encryption keys. We focus on the *curious intruder* model [31] to analyze the attacks posed to DISC, which requires Hilbert transformation of the data points before encryption. Moreover, the Hilbert cell values of data points are encrypted using OPE, while the actual data points are encrypted using AES. AES provides the standard security guarantee and hence, is not susceptible to common attacks triggered by obtaining background knowledge of a subset of the data.

Hilbert transformation

It is intuitive that if an attacker is aware of the space transformation technique used (i.e. Hilbert curve in DISC), as well as a subset of the original spatial data points along with their transformed Hilbert values, the attacker can determine the key of the transformation technique i.e. Hilbert Space Key (HSK). But, the study by [7] suggests that it is infeasible for a malicious adversary to infer the exact HSK being used as there exist an exponential number of possible HSKs, as shown in the analysis of the attack below.

Brute-force attack: In the event of a brute-force attack, the attacker will have $2^b * 2^b * 2^b$ elements for x_0, y_0 and θ in the entire search space. The number of possibilities for the curve granularity g as G. To accurately find the curve's starting point, it should lie on the intersection of two edges. Using b bits for each x_0 and y_0, the attacker can generate 2^b values on each axis and this will require an exhaustive search over the grid. Likewise, for the curve orientation, the entire continuous 360° space should be discretized to generate 2^b values. Since $G \ll 2^b$, the complexity of the brute-force attack to find the transformation key is $O(2^{3b})$, where b is the number of bits used to represent each parameter in the HSK. Choosing a large enough value for b, will make the Hilbert mapping irreversible. Given that b is chosen to be 32 bits, the complexity of finding the HSK parameters would be $O(2^{3*32})$ for different possibilities of the curve granularity.

Order-preserving encryption

In OPE, the ciphertext is in the form of numeric data. Since OPE schemes can support comparison operations on the ciphertext, it is infeasible to achieve ideal security for OPE. Boldyreva et al. [28] were the first to provide a complete security analysis on OPE. The higher the security provided by the encryption scheme, the lower its efficiency and support of operations. Therefore, to achieve a low communication cost and a single round of data exchange between the CSP and TU, we settle on a weaker OPE scheme that leaks as little information of plaintext as possible. Also, with traditional encryption methods, it is not possible for untrusted CSP to process user queries on the encrypted data.

Ciphertext-only attack: This is the most common attack for encryption techniques. The *one-wayness* property of encryption was proven to hold, where the adversary is unable to invert the encryption without the knowledge of the key. But the adversary may be able to gain information about the order of encrypted values revealed by the OPE scheme and predict the plaintext values (i.e. if (p_1, c_1) and (p_2, c_2) are known for $p_1 < p_2$ and no other known

plaintext-ciphertext pairs occur between these two). If the adversary knows a certain number of plaintext-ciphertext pairs, the scheme splits the plaintext and ciphertext spaces into subspaces. On each subspace, the analysis under each *one-wayness* definition reduces to that of the random order-preserving function domain and range of the subspace. The ciphertext space must be at least twice the size of the plaintext space. Thus, in the OPE scheme adapted by the DISC/DISC* approach, the OPE parameters are chosen in such a way that subspaces are unlikely to violate this condition.

Discussion: The AES encrypted spatial points are stored along with their OPE Hilbert values in the index at the CSP. The security of the scheme can be exposed if the attacker can gather limited background knowledge about the data distribution without the encryption keys. However, Hilbert R-trees do not expose the complete ordering of spatial data points. In some cases, even if it is possible to gather some information regarding the ordering of the leaf nodes from queries over a period of time, the attacker cannot infer the actual location of the spatial point (cf. Lemma 1).

Lemma 1 *Considering the worst case, assuming that an attacker can decrypt the ordering of a subset of Hilbert cells in the grid, the attacker can estimate the Hilbert value of the actual spatial data point as one of the* $O(2^{32})$ *different possibilities of the Hilbert curve granularity, g.*

Proof Given the total number of spatial points in D, and the granularity of the Hilbert curve, g, the average number of points assigned to each Hilbert cell is computed as follows:

$$cell_{avgPoints} = \frac{size(D)}{2^{2*g}} \quad (1)$$

Therefore, selecting a small value for the Hilbert curve granularity, g (i.e. $g << size(D)$), results in multiple spatial points being assigned to the same Hilbert cell. This increases the security provided by DISC and DISC*, but increases the number of false positives returned in the query result. □

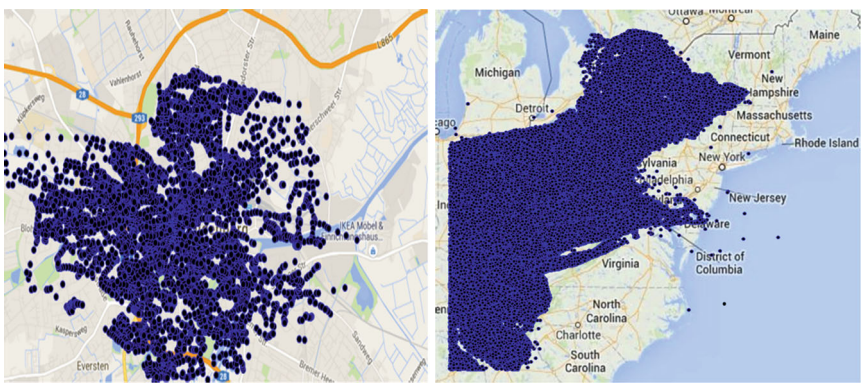
Experimental evaluation

To evaluate the performance of the proposed approaches, DISC and secure DISC*, several experiments were conducted. We compare and analyze the difference in communication cost based on the query size and node capacity of the index. DISC and DISC* are empirically compared with the Cryptographic Transformation (CRT) method proposed by Yiu et al. [5]. Experiments were performed on an Intel Core i7-3770 CPU @ 3.40 GHz with 16 GB of RAM running the 64-bit Ubuntu operating system and implemented in C++.

Experiment on all datasets are conducted with varying query sizes ranging from 5 to 20%, where each spatial range query is a randomly distributed region in the normalized domain space. Each MBR in the non-leaf and leaf nodes is represented by 4 coordinates. Each LHV in the nodes is the Hilbert value and is represented by 4 bytes (an integer), and each spatial data point consists of x and y coordinates in double precision (16 bytes). The results shown in the experiments below are averaged over 100 runs for 4 varying query sizes.

Spatial datasets

Our experiments are performed on four real-world spatial datasets obtained from [32]: (1) City of Oldenburg (**OL**) Road Network comprising of 6104 points, (2) City of San Joaquin County Road Network (**TG**) having 18,263 points, (3) North East USA (**NE**) consisting of 123,593 points, which represent the real postal addresses and (4) Road Network of North America (**NA**) with 175,813 points is approximately 2800 KB. The domain of each dataset is normalized to the unit square $[0, 1]^2$ and two of these datasets are shown in Fig. 7.



(1) Oldenburg (**OL**) (2) North East USA (**NE**)

Fig. 7 Two real-world spatial datasets: road networks. (*1*) Oldenburg (OL), (*2*) North East USA (NE)

Table 2 Index construction time (s) for various node capacities

	Index construction time (s)			
Node capacity	OL	TG	NE	NA
10	0.26	1.51	4.31	8.14
20	0.29	2.36	5.50	9.72
30	0.35	3.91	7.36	11.19

Index construction time

In DISC, the Data Owner builds an index over the spatial dataset which is based on the Hilbert R-tree. The dataset is transformed using the Hilbert curve. An empty Hilbert value means that there is no data point associated with it and these empty values are discarded during the initialization process. The index construction time is averaged over 100 runs. The DISC construction time is proportional to the number of spatial points in the dataset and Table 2 shows time taken to build the index for all 4 datasets. DISC is constructed initially at the DO and outsourced to the CSP where it is used to answer queries and handle dynamic updates from DO.

Effect of node capacity

The value of node capacity is the maximum number of data points per node, which dictates how the index is constructed. An optimal value of the capacity reduces the amount of pages searched at the server as well as the amount of data sent from the CSP to the TU. Figure 8 shows the effect of the node size on the query processing time on the DISC index at the cloud server for a small and big dataset, with node capacity ranging from 10 to 30. It can be seen that the query time and node capacity have an inversely proportional relationship, the query search time decreases when the node capacity is increased. This is due to the fact that Hilbert R-tree index will have a greater height (i.e. more levels) when a smaller node capacity is used. Other datasets and query sizes follow a similar trend, as does DISC*.

Query processing time on the cloud

The search performance of the DISC index is faster-than-linear with regard to the number of spatial data points. Figure 9 shows the time taken to process range queries on an encrypted index at the Cloud Service Provider. The experiments have been conducted on all datasets and results of OL and NE are displayed due to space limitations. The other two datasets display a similar trend. The range query size in this experiement is 20% and the average query time is measured in *ms* over 100 runs for varying node capacities of the index in DISC. The x-axis shows the node capacity, while the y-axis shows the average query processing overhead in milliseconds. When the node capacity is small, the average query time is more, but as the node capacity increases, the query time drops and this is appropriate for a CSP as they have adequate computing power. With a smaller node capacity, more levels in the tree have to be navigated through. Comparing DISC with the more secure DISC* reveals that DISC* has a faster processing time as the leaf nodes are encrypted in AES and its values cannot be compared against Hilbert values. The most time-consuming task in DISC is to sequentially check overlapped leaf node for queried Hilbert values. As the size of the query increases, the time taken to process the query also increases.

False positive rate

Both DISC and DISC* induce a number of false positives in the query result. Given that false positives increase security, a significant false positive rate is not a major concern in a secure outsourcing scheme such as DISC. Hence it is expected that the more secure scheme DISC* has a higher false positive rate than DISC. As we have noted previously, data owners can tune the spatial index for different tradeoffs between security, query efficiency and false positives. Figure 10 shows the average false positive rate for 100 queries over the larger data set NE, while other datasets follow a similar trend. We compare both schemes and show the impact of different query sizes

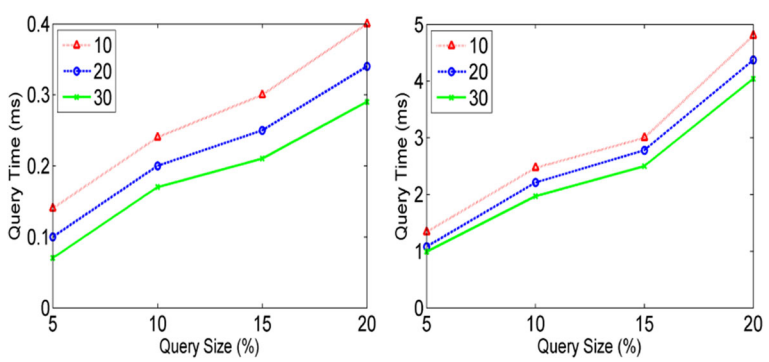


Fig. 8 Query processing time (ms) based on various node capacities (Datasets: OL & NE)

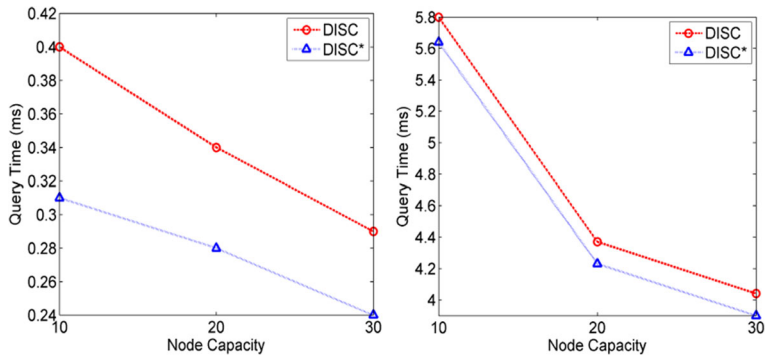


Fig. 9 Query processing time (ms) at the CSP (Datasets: OL & NE)

and various node capacities on the false positive rate. For larger query sizes, the false positive rate increases as more data points are returned. For larger node capacities, the false positive rate is usually less than 50% and the increase in the average false positive rate becomes steady.

End-to-end time for a query between the DO and CSP

Figure 11 illustrates the average end-to-end user time of the approach evaluating spatial range queries (10%) on the **OL** and **NE** datasets for both schemes. The TU makes use of an Android mobile phone (HTC) in this experimental evaluation. The transfer bandwidth of the network considered here is 1 Gbps [33]. The round-trip network delay time is the time taken for one round-trip between the TU and the CSP, which is 125 *ms* [34]. It is not shown in the graph as our approach requires only one round between the CSP and TU and is hence fixed for all query sizes. Other network delays and faulty packet transmission issues due to network errors is not taken into account. The *end-to-end user time* comprises of four components:

- Query Encryption Time (**TU**): Is the amount of time taken to encrypt the query Hilbert cell set using Order-Preserving Encryption.
- Query Request Time (**TU**): Is the amount of time taken to transform the range query window to the Hilbert cell set and send it to the CSP over the network.
- Query Response Time (**CSP**): Is the amount of time taken to process the spatial query over DISC and generate the result set to send back to the user over the network.
- Result Decryption Time (**TU**): Is the amount of time taken filter the false positives and decrypt the query result set using the AES key. The decryption time taken for 1 KB of data is 0.015 ms using the AES scheme at the user-end [35].

Figure 12 shows the network transer time in milliseconds over a 1 Gbps connection. The query request from the TU to the CSP as well as the query response from the CSP to the TU over the network is shown for different query sizes for the NE dataset. The end-to-end time relies on the size of the query as well as the indexed dataset. It is visible that the bulk of the time is taken to return the query result from the CSP to the TU. The CSP has scalable computational power and this does not affect the performance of the location-based services provided. The main idea is to have minimal processing done at the

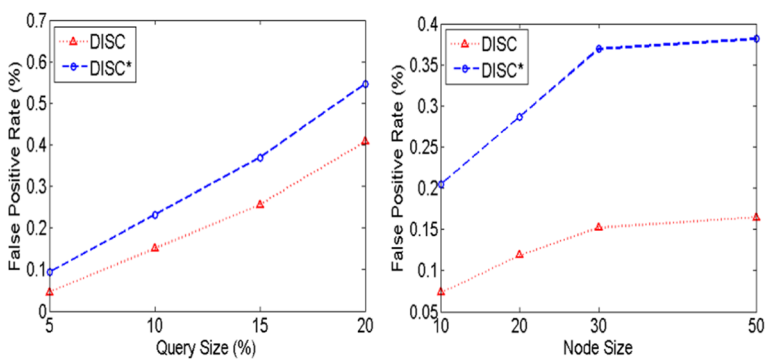


Fig. 10 Average false positive rate (%) for query sizes and node capacity

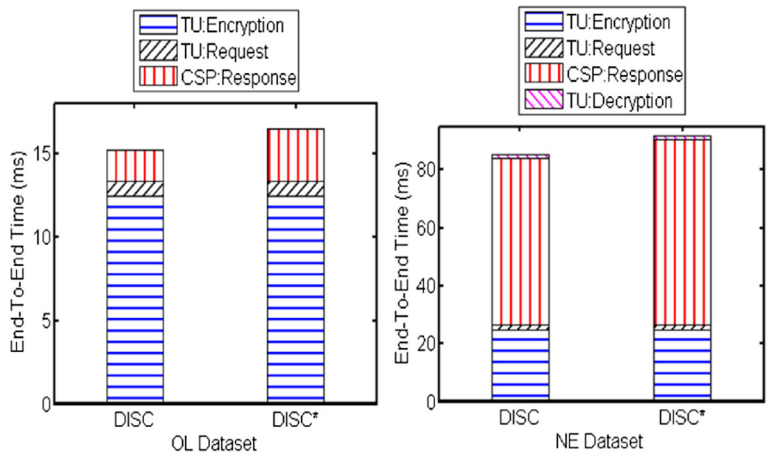


Fig. 11 User to cloud service provider: end-to-end spatial query time (ms)

user-end as they have limited resources. Forming the query takes a minute amount of time and the efficient encryption scheme (AES) allows fast decryption of query results. Since the OL dataset is small in size, the decryption time is almost negligible and hence not shown in the graph. The secure DISC* scheme has a higher time at the CSP as a larger result set is returned.

Query communication cost between user and CSP

In the experiment shown in Fig. 13, 100 random spatial range queries of varying sizes were generated and the amount of data (i.e. average communication cost in bytes) exchanged between the CSP and TU is measured. The communication cost includes: 1) Query Request: TU transforms the query region into a set of Hilbert cell values and transmits the encrypted set of values to the server, 2) Query Response: the CSP searches DISC for corresponding leaf nodes and returns the encrypted data points as the result. The query sizes ranging from 5 to 20% are used in the experiments.

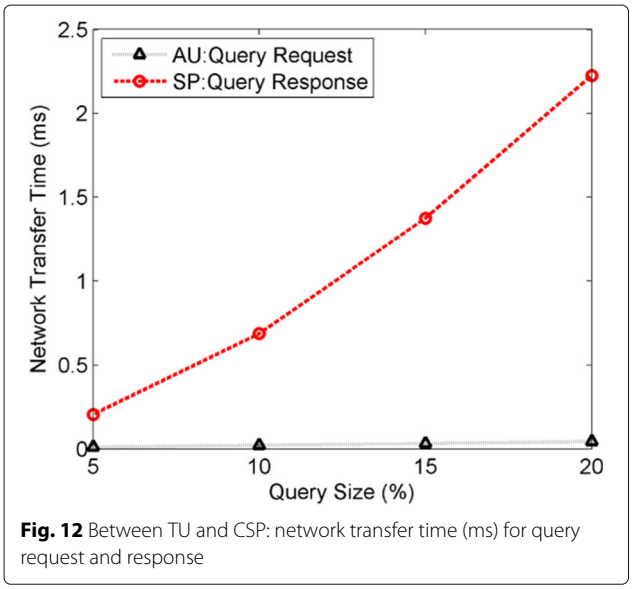


Fig. 12 Between TU and CSP: network transfer time (ms) for query request and response

In DISC, the leaf nodes are searched sequentially for query Hilbert values so that the exact points can be returned. Therefore, varying the node capacity of DISC would result in identical communication cost. Whereas, DISC* has a higher communication cost in comparison to DISC as the entire leaf nodes are returned as part of the query result. Hence, increasing the node size in DISC* would increase the communication cost as well. It is clear that the average communication cost increases linearly as the query size increases due to the increase in number of points returned. The result is decrypted at the user-end using the AES key. The OL and NE dataset results are displayed here, while the rest of the datasets follow a similar trend.

The proposed retrieval scheme is compared with the CRT technique (the R^*-tree index structure is built and encrypted using AES), and shown in Fig. 13. This is an appropriate comparison, since the R^*-tree achieves the same query time complexity as these schemes. It is demonstrated that our method is at least twice as fast as we require only one round-trip between the CSP and TU, which minimizes the communication overhead. It can be seen in experiments that the communication cost increases as the size of the spatial dataset increases. Moreover, for query sizes greater than 15%, there is a sharp increase in the communication cost of **CRT** due to the amount of messages exchanged between the user and server, which is dominated by the depth of the R^*-tree utilized. This is due to the fact that in CRT, the query is processed at both the user and server side, resulting in multiple rounds of communication. Moreover, CRT returns entire leaf nodes to the user, while using DISC, the

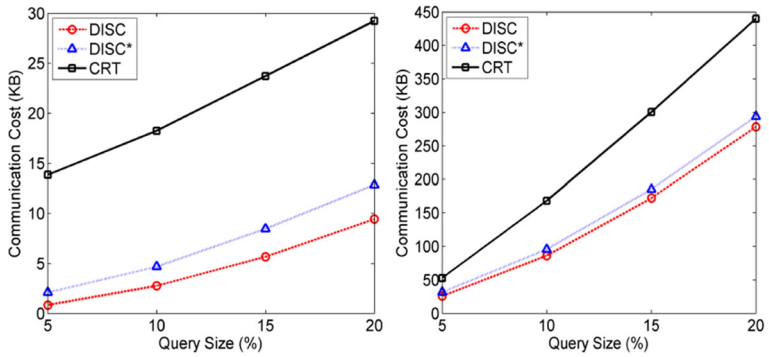


Fig. 13 DISC and DISC* vs. CRT: Effect on communication cost (KB) with different query sizes (Datasets: OL & NE)

server is able to retrieve the relevant data points in the leaf nodes based on the query Hilbert values. Even DISC* is able to achieve highly efficient queries, while hiding the ordering of data points within each leaf node.

Dynamic updates at the CSP

Dynamic updates are initiated by the DO and transmitted to the CSP. The CSP executes the updates on the encrypted index based on the Hilbert value of the given update. The DISC scheme allows dynamic updates on encrypted data by preserving the order of data (using OPE) in the encrypted leaf and non-leaf nodes, but preserving the parent-child relationships by not encrypting the pointers. To the best of our knowledge, none of the prior works provide the update characteristic in their retrieval schemes. The main advantages of the Hilbert R-tree is its ability to handle dynamic updates. The CRT [5] technique does not allow updates and only handles static data, as the tree split/merge procedure cannot be applied on AES encrypted data in the R^*-tree.

Experiments were performed on DISC for all dynamic update operations and the only communication cost entailed is the request sent from the DO to the CSP, where the update is handled. For instance, the insertion of a new object and overflow handling is peformed using the Hilbert value of the new data point. Whereas in DISC*, updates cannot be performed entirely at the CSP as the leaf nodes are encrypted using AES which does not allow comparison operations. In Fig. 14, we show the communication cost for three types of insert operations. First, the DO sends the update request which is just an encrypted Hilbert value. For the sake of simplicity, the simple insert update is explained in detail in "Secure dynamic updates" section. The CSP responds by returning the affected leaf node and the DO returns the updated encrypted nodes. In the case of overflow and sibling insert, multiple rounds of communication are required as neighboring leaf nodes are also sent to the DO. Further details on the more complex insert operations can be found in [29]. The other update operations, delete and modify, follow a similar exchange of data as insert and excluded for brevity. After analysis, the most costly operation was modify, followed by delete and lastly insert.

Conclusions

Data outsourcing has attracted much attention recently due to the emergence of cloud computing. Cloud computing virtualizes storage and computing resources at the server and provides data to trusted users. However, securing the outsourced data in the untrusted cloud server has security concerns. In this work, we are trying to strike a balance between data confidentiality and efficient query processing at the cloud service provider. We propose the DISC retrieval scheme, which has a dynamic encrypted index for spatial data at the CSP. The index is encrypted using the OPE technique, as it allows comparison operations on encrypted data at the server. Moreover, DISC allows dynamic updates at the CSP. An enhanced and

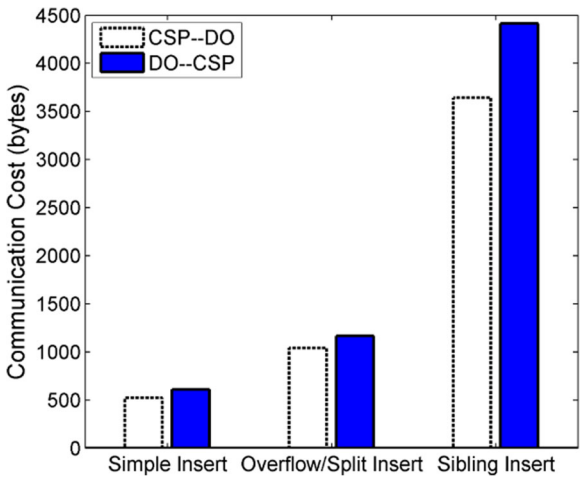


Fig. 14 Communication cost (bytes) between DO and CSP for an INSERT update

secure version, DISC*, is proposed as well. Several attack models are defined and it is shown that our scheme provides proven security against well-known attacks. Lastly, experiments were conducted and it is demonstrated that the DISC retrieval scheme improves the range query performance and is superior to existing cryptographic approaches. In conclusion, the retrieval scheme proposed in this paper enables users to retrieve spatial range query responses efficiently and allows dynamic updates on the encrypted index.

Authors' contributions
All the listed authors made intellectual contributions to the research. AT implemented the solution, conducted the experiments and, was responsible for preparing and editing the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Research Institute of Sciences & Engineering, University of Sharjah, Sharjah, United Arab Emirates. [2]Department of Electrical & Computer Engineering, University of Sharjah, Sharjah, United Arab Emirates. [3]Department of Computer Science, University of Sharjah, Sharjah, United Arab Emirates.

Received: 30 November 2016 Accepted: 7 February 2017
Published online: 28 February 2017

References

1. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In: IEEE Infocom, 2010 Proceedings. IEEE, San Diego, pp 1–9
2. Xu H, Guo S, Chen K (2014) Building confidential and efficient query services in the cloud with rasp data perturbation. IEEE Trans Knowl Data Eng 26(2):322–35
3. Hu H, Xu J, Ren C, Choi B (2011) Processing private queries over untrusted data cloud through privacy homomorphism. In: IEEE 27th International Conference on Data Engineering. IEEE, Hannover, pp 601–612
4. Zhao G, Rong C, Li J, Zhang F, Tang Y (2010) Trusted data sharing over untrusted cloud storage providers. In: IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, Indianapolis, pp 97–103
5. Yiu ML, Ghinita G, Jensen CS, Kalnis P (2010) Enabling search services on outsourced private spatial data. VLDB J 19(3):363–84. Springer
6. Lawder JK, King PJH (2001) Querying multi-dimensional data indexed using the hilbert space-filling curve. ACM Sigmod Record 30(1):19–24
7. Khoshgozaran A, Shahabi C (2007) Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In: Advances in Spatial and Temporal Databases. Springer, Boston, pp 239–257
8. Ku WS, Hu L, Shahabi C, Wang H (2013) A query integrity assurance scheme for accessing outsourced spatial databases. Geoinformatica 17(1):97124. Springer
9. Wang P, Ravishankar CV (2013) Secure and efficient range queries on outsourced databases using r-trees. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, Brisbane, pp 314–325
10. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. ACM, Paris, pp 563–574
11. Wang B, Li M, Wang H (2016) Geometric range search on encrypted spatial data. IEEE Trans Inform Forensics Secur 11(4):704–719
12. Liu Z, Chen X, Yang J, Jia C, You I (2016) New order preserving encryption model for outsourced databases in cloud environments. J Netw Comput Appl 59:198–207
13. Pub NF (2001) 197: Advanced encryption standard. Federal Inform Process Stand Publ 197:441–0311
14. Damiani E, Vimercati S, Jajodia S, Paraboschi S, Samarati P (2003) Balancing confidentiality and efficiency in untrusted relational dbmss. In: Proceedings of the 10th ACM Conference on Computer and Communications Security. ACM, Washington, pp 93–102
15. Hore B, Mehrotra S, Tsudik G (2004) A privacy-preserving index for range queries. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases. VLDB. pp 720–731
16. Hore B, Mehrotra S, Canim M, Kantarcioglu M (2012) Secure multidimensional range queries over outsourced data. VLDB J Int J Very Large Data Bases 21(3):333–358. Springer-Verlag New York, Inc.
17. Hacigümüs H, Iyer B, Mehrotra S (2002) Providing database as a service. In: 18th International Conference on Data Engineering, 2002. Proceedings. IEEE, San Jose, pp 29–38
18. Batten LM, Abawajy J, Doss R (2011) Prevention of information harvesting in a cloud services environment. In: CLOSER 2011: Proceedings of the 1st International Conference on Cloud Computing and Services Science. INSTICC, Noordwijkerhout. pp 66–72
19. Kim HI, Hong ST, Chang JW (2014) Hilbert-curve based cryptographic transformation scheme for protecting data privacy on outsourced private spatial data. In: 2014 International Conference on Big Data and Smart Computing (BIGCOMP). IEEE, Bangkok. pp 77–82
20. Kim HI, Hong S, Chang JW (2015) Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data. Data Knowl Eng 104(2016):32–44. Elsevier
21. Talha AM, Kamel I, Aghbari ZA (2015) Enhancing confidentiality and privacy of outsourced spatial data. In: 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud). IEEE. pp 13–18
22. Ku WS, Hu L, Shahabi C, Wang H (2009) Query integrity assurance of location-based services accessing outsourced spatial databases. In: Advances in Spatial and Temporal Databases. Springer, Aalborg. pp 80–97
23. Wong WK, Cheung DW-L, Kao B, Mamoulis N (2009) Secure knn computation on encrypted databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. ACM, Providence, pp 139–152
24. Lu Y (2012) Privacy-preserving logarithmic-time search on encrypted data in cloud. In: NDSS, San Diego
25. Gentry C, et al (2009) Fully homomorphic encryption using ideal lattices. In: STOC, Bethesda Vol. 9. pp 169–178
26. Yiu ML, Assent I, Jensen CS, Kalnis P (2012) Outsourced similarity search on metric data assets. IEEE Trans Knowl Data Eng 24(2):338–52
27. Moon B, Jagadish HV, Faloutsos C, Saltz JH (2001) Analysis of the clustering properties of the hilbert space-filling curve. IEEE Trans Knowl Data Eng 13(1):124–141
28. Boldyreva A, Chenette N, ONeill A (2011) Order-preserving encryption revisited: Improved security analysis and alternative solutions. In: Advances in Cryptology–CRYPTO 2011. Springer, Santa Barbara, pp 578–595
29. Kamel I, Faloutsos C (1993) Hilbert r-tree: An improved r-tree using fractals. Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, September 1994, pp 500–509
30. Chung KL, Tsai YH, Hu FC (2000) Space-filling approach for fast window query on compressed images. IEEE Trans Image Process 9(12):2109–16
31. Goldreich O (2004) Foundations of Cryptography: Volume 2, Basic Applications, Cambridge university press
32. Real Spatial Datasets. http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm
33. Georgiou S, Tsakalozos K, Delis A (2013) Exploiting network-topology awareness for vm placement in iaas clouds. In: Third International Conference on Cloud and Green Computing (CGC), 2013. IEEE, Karlsruhe. pp 151–158
34. Chen YC, Nahum EM, Gibbens RJ, Towsley D, sup Lim Y (2012) Characterizing 4g and 3g networks: Supporting mobility with multi-path tcp. University of Massachusetts Amherst, Tech. Rep
35. Popa RA, Redfield C, Zeldovich N, Balakrishnan H (2011) Cryptdb: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, Cascais, pp 85–100