Secure Reverse k-Nearest Neighbours Search over Encrypted Multi-dimensional Databases

Theodoros Tzouramanis
University of the Aegean
83200 Samos, Greece
ttzouram@aegean.gr

Yannis Manolopoulos
Open University of Cyprus
2220 Latsia, Cyprus
yannis.manolopoulos@ouc.ac.cy

ABSTRACT

The reverse k-nearest neighbours search is a fundamental primitive in multi-dimensional (i.e. multi-attribute) databases with applications in location-based services, online recommendations, statistical classification, pat-tern recognition, graph algorithms, computer games development, and so on. Despite the relevance and popularity of the query, no solution has yet been put forward that supports it in encrypted databases while protecting at the same time the privacy of both the data and the queries. With the outsourcing of massive datasets in the cloud, it has become urgent to find ways of ensuring the fast and secure processing of this query in untrustworthy cloud computing. This paper presents searchable encryption schemes which can efficiently and securely enable the processing of the reverse k-nearest neighbours query over encrypted multi-dimensional data, including index-based search schemes which can carry out fast query response that preserves data confidentiality and query privacy. The proposed schemes resist practical attacks operating on the basis of powerful background knowledge and their efficiency is confirmed by a theoretical analysis and extensive simulation experiments.

CCS CONCEPTS

• Information systems~Data encryption • Security and privacy~Privacy-preserving protocols • Security and privacy~Management and querying of encrypted data • Theory of computation~Database query processing and optimization (theory) • *Theory of computation~Cryptographic protocols*

KEYWORDS

Data outsourcing, cloud database services, secure cloud computing, confidential multi-dimensional data retrieval, encryption.

IDEAS 2018, June 18–20, 2018, Villa San Giovanni, Italy

ACM ISBN 978-1-4503-6527-7/18/06…$15.00
https://doi.org/10.1145/3216122.3216170

ACM Reference format:

T. Tzouramanis and Y. Manolopoulos. 2018. S Secure Reverse k-Nearest Neighbours Search over Encrypted Multi-dimensional Databases. In *Proceedings of the 22th ACM International Database Engineering and Applications Symposium, Villa San Giovanni, Italy, June 2018 (IDEAS 2018)*, 11 pages. https://doi.org/10.1145/3216122.3216170

1 INTRODUCTION

In recent years, cloud computing has become an increasingly prevalent platform for the deployment, management, and provisioning of large-scale services through an Internet-based infrastructure. At the same time, much of the focus has been on ways of coping with the security and privacy problems which are particular to cloud computing [1]. An interesting application of this platform is the Database as a Service (DaaS) [2], by means of which individuals and organizations can outsource both their database and its management functionality to the cloud service provider to reduce the database management cost. When these data are stored and queried on the cloud, encryption represents a straightforward solution to ensure that the cloud owner or any third party gaining access to the outsourced database do not have access to information related to the sensitive data or to the queries of their clients.

When the data owner attempts to take advantage of the computation capabilities of the cloud service to analyse or query the data stored in the cloud for the purpose of extracting information and patterns, encryption will impede querying and analysing functionalities and performance since the classical encryption methods do not allow even simple operations over the ciphertext. To address this issue searchable encryption was proposed to protect the privacy of the users' data on the cloud without a loss of searching functionalities on the server-side. More specifically, a data owner can encrypt data with searchable encryption prior to outsourcing these data to the server, thus the server can carry out a confidential search of the encrypted data without the need to decrypt them. The first searchable encryption scheme was designed back in 2000 [3] and various techniques related to query processing over encrypted data were subsequently proposed to improve efficiency and to enrich the searching functionalities of the DaaS cloud model, including the range [4, 5], *k*-nearest-neighbours [6, 7, 8], top-k [9] and aggregate [10] queries.

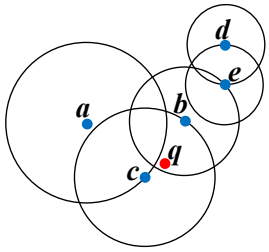


Figure 1: A RkNN query example with k = 1.

This work addresses the problem of the secure and efficient processing of the reverse *k*-nearest neighbours (R*k*NN) query over multi-dimensional (*i.e.* multi-attribute) datasets on the cloud. Given a query point on the multi-dimensional space, the R*k*NN query finds all the data points of a dataset which have the query point as one of their *k* nearest neighbours. This database analysis operation can be used either as a standalone query (e.g. in similarity search applications) or as a core module of common data analysis tasks such as machine learning and data mining (e.g. classification and clustering). Figure 1 illustrates a simple example in two dimensions: let's imagine that it represent a real-life application involving a small dataset of five research articles *a*, *b*, *c*, *d* and *e*, which an enterprise wishes to store and process on the cloud. A new article *q* arrives, which the service wishes to promote and advertise. The service will identify the appropriate research community on the basis of the distance between the area of interest of their research and the feature vector representing this new article *q*. The circles around the articles indicate an area corresponding to the work most closely related them. The answer of the R*k*NN query for $k = 1$ with regard to this new article *q* will return the articles (and the email addresses of their authors) that contain *q* in their circles, *i.e.* $R1NN(q) = \{b, c\}$.

The R*k*NN problem is complementary to that of finding the *k* nearest neighbours of a query point. For example, if we posit that $p \in k\text{NN}(q)$ holds, this does not necessarily imply that $p \in \text{R}k\text{NN}(q)$ truly holds as well, and via-versa. Figure 1 illustrates such an example, in which, while *b* is a reverse nearest neighbour of *q*, *b* is not at the same time the nearest neighbour of *q*.

To the author's knowledge, to-date, solutions have not yet been put forward to evaluate securely R*k*NN query predicates on sensitive data on the cloud.

The solutions proposed in this paper for R*k*NN query processing offer strong security guarantees and efficient performance. The paper considers both the case of encrypted data that can be supported, and the case of encrypted data that cannot be supported by specialized indexing methods to execute the R*k*NN query. In a real-life application, a cloud service might provide a storage and processing environment in which the data indexing mechanism may be offered as an additional and chargeable option, allowing the user to disable it to reduce financial costs. This paper suggests four schemes and identifies trade-offs between security and efficiency. The schemes proposed allow a cloud server to correctly verify whether a data object belongs to the R*k*NN set of a query point in the encrypted data domain, without the privacy of the data or of the user's query and preferences being compromised.

Section 2 discusses the necessary preliminaries and notations; Section 3 briefly surveys some of the work done in fields closely related to the R*k*NN search over encrypted data; Section 4 presents four schemes for R*k*NN query processing in encrypted multi-dimensional databases; Section 5 discusses the performance of experimental results with both real and synthetic datasets; and Section 6 draws conclusions and makes suggestions for future research.

2 RELATED WORK

R*k*NN query processing has received considerable attention in the past two decades [11, 12, 13, 14, 15, 16, 17, 18, 19, 20] because this query represents a fundamental and crucially important data processing operation [21, 22]. Intuitively, the R*k*NN of an object *o* are those objects over which *o*, being one of their *k* nearest neighbours, has significant "influence". Such "influence sets" may lead to useful observations on the correlation among data, as shown by Korn and Muthukrishnan in their pioneering paper [21]. Indeed, the R*k*NN search is inherent to any application where the similarity between two objects can be quantified into a single value, using an appropriate evaluating process.

The approach of [21] suggests the pre-computation of the *k* nearest neighbours of every data point on the workspace. Then, given the query point *q*, the proposed algorithm can compare it to the existing *k*-th nearest neighbour of every point. For every data point *p*, the algorithm computes and stores a spherical region centred at *p* and with a radius equal to the distance from *p* to its *k*-th nearest neighbour, as illustrated by Figure 1 for $k = 1$. The authors of [21] prove that if the query point *q* falls within the spherical region of *p*, then *p* is an R*k*NN of *q*. All the circular regions can be organized into a multi-dimensional index structure, such as the R-tree [23], for effective storage and query performance. Yang and Lin [19] make adjustments to this approach and propose the RDNN-tree (R-tree containing Distance of Nearest Neighbours) for applications in which both the *k* nearest neighbours and the R*k*NN queries need to be supported. Stanoi et al. [20] divide the two-dimensional space centred at the query *q* into six equal partitions. Their paper demonstrates that only one candidate R1NN data point can exist in every partition. However, the number of regions to be searched to find possible candidate R1NN data points increases exponentially with the dimensionality, rendering this approach inefficient even in a three dimensional setting. Later, Tao et al. [17] introduced the TPL index-based method for the R*k*NN search. Given a query point *q*, TPL recursively prunes the space using the bisector between the query point *q* and its *k*-th nearest neighbour. Some of the most popular R*k*NN algorithms are surveyed and empirically compared in [15].

Specialised fields have also applied the R*k*NN query among others: large networks and graphs [16], moving objects [12], data streams [18], uncertain data [14], spatial keywords search [13] and continuous queries monitoring [11]. All these approaches assume a fully-trustworthy environment (such as that of a local database server) and none of them deals with the matter of the protection of the data's confidentiality in cases when these data

are outsourced for storage and processing in a potentially untrustworthy cloud server.

From another viewpoint, there have been significant developments in the related fields of secure queries processing in multi-dimensional databases. Various solutions to the opposite problem of supporting securely the *k*-nearest neighbours query have been put forward (the recent work of [6] and [7] contain references to less recent research efforts). However, since the nearest neighbour search differs from the reverse nearest neighbour search, including in the plaintext domain, these models are not applicable to the problem under consideration in this paper. Secure range query processing in cloud databases, for both indexed and non-indexed cloud data, has also received some attention in recent years [4, 5] but again, these methods cannot be applied to the R*k*NN problem. Recently solutions were proposed for the problems of the top-*k* query [9], the skyline query [24], as well as several types of aggregate queries [10].

To the authors' knowledge, the issue of making secure R*k*NN query processing in multi-dimensional databases in the cloud has not yet been addressed. In the field of untrustworthy location-based systems, Lin et al. [25] propose an algorithm that computes the probability of a moving user being the R1NN result of a query point, on the premise that the mobile users' locations and identities are hidden using anonymization techniques. Other research propose solutions that protect the query input of a R*k*NN query that is sent to a location-based service: they employ either a clocking region that hides the query input [26] or a private information retrieval technique which accesses the database anonymously [27]. None of these solutions deals with the protection of the plaintext database from any unauthorized access when these data are hosted in an untrustworthy data centre in current applications.

3 PRELIMINARIES AND NOTATIONS

The model proposed in this paper is based on the DaaS cloud services model and it is constituted of two separate entities: the client, who is the legitimate owner of the data, and the cloud server. The client outsources the data to the server in an encrypted form with the expectation that s/he will be able to carry out a remote search without incurring a breach of privacy in the process. Therefore, one of the goals of the design of the model should be to prevent the cloud provider or any outsider (hereafter "the observer") from obtaining any amount of information about the protected plaintext database besides what can be derived from the legitimate client's encrypted search results. The client, on the other hand, should be able to obtain access to the plaintext of the relevant encrypted content. The system should offer an efficient encrypted R*k*NN search functionality at a low computation cost and ensuring secure communication.

The observer is assumed to be honest-but-curious, and to intend to gain full access to the plaintext of the encrypted stored data but with no intention of altering any of it. The threats model also assumes that, besides already knowing all the procedures involved (such as the encryption and decryption algorithms), the observer might be able to carry out the *known input-output attack*, in which it is assumed that s/he has access to a sample of pairs of plaintext and ciphertext of data tuples or of the client's queries, by knowing which ciphertext corresponds to which plaintext. The known input-output attack is an attack targeting the encryption key and all the plaintext tuples of an encrypted dataset to which the observer has access.

This study considers that every data tuple is a multi-dimensional point p $(p_1, p_2, ..., p_d)$. It models its d attributes as dimensions and their values as their coordinates. The data points construct a dataset P which is encrypted and stored on the cloud. The R*k*NN query is defined as follows.

Definition: given a points dataset P and a query point q, a R*k*NN query aims to find the subset $RkNN(q)$ of data points in P: $RkNN(q) = \{p \in P: dist(p, q) \leq dist(p, p_k)$, where p_k is the k-th nearest neighbour of p and $dist$ is a distance metric (Euclidean distance is assumed in this paper)$\}$.

4 THE PROPOSED ENCRYPTION SCHEMES

4.1 Solutions that do not rely on data indexing mechanisms

4.1.1 The Basic Non-Index-based Encryption Scheme

Every data and query point will be considered as a column vector $p\ [p_1, p_2, \ldots, p_d]^T$ and $q\ [q_1, q_2, \ldots, q_d]^T$, respectively, where p^T and q^T are the transposes of p and q. Given a dataset P and a query point q, a simple, yet not secure, solution for computing the R*k*NN (q) set will be to initially represent every data point p using two vectors $\hat{p} = [p_1, p_2, \ldots, p_d, -0.5p^2]^T$ and $\widehat{ps} = [s \cdot p_1, s \cdot p_2, \ldots, s \cdot p_d, s]^T$, where p^2 is the scalar product between the transpose p^T of p and p in the form $p^2 = p^T * p = p_1 \cdot p_1 + p_2 \cdot p_2 + \ldots + p_d \cdot p_d$, and s is a random positive number. The encrypted versions p' and ps' of the vectors $\hat{p}$ and $\widehat{ps}$ will be computed as $p' = E(p) = M^T * \hat{p}$ and $ps' = E_s(p) = M^{-1} * \widehat{ps}$, respectively, *i.e.* as a scalar product between the vectors $\hat{p}$ and $\widehat{ps}$ with a $(d + 1) \times (d + 1)$ square matrix M which will represent the encryption key. These two encrypted values p' and ps' will then be sent to the server and the first will be stored in table T while the second will be stored in table Ts. Additionally, the query point q will be represented as the vector $\widehat{qs} = [s \cdot q_1, s \cdot q_2, \ldots, s \cdot q_d, s]^T$, and its encrypted version $qs' = E_q(q) = M^{-1} * \widehat{qs}$ will also be stored in table Ts. Using this setting and the secure k-nearest neighbours query processing model presented in [8], the server can then pick up every encrypted point $p' \in T$ and compute its $k + 1$ nearest neighbours from the points that are stored in table Ts[1]. If qs' is one of the $k + 1$ nearest neighbours of any point p', then $p \in$ R*k*NN (q), thus p' needs to be returned to the client.

[1] It should be pointed out that $\forall\ p' \in T$ the described process will search in table Ts for the $k + 1$ nearest neighbours of p' (instead of its k nearest neighbours) because the corresponding encrypted version $ps' \in Ts$ of p' will also be found to be one of the k nearest points of p' (since they actually coincide), therefore the client will need to ignore it from the set of the results.

This solution however is not secure: because $\forall\ p' \in T$ it preserves the order of the distances of all the data points in table Ts to p'. For example, this solution would reveal that the nearest point to a in Figure 1 is the data point c and that the second nearest is the query point q, the third nearest is the data point e, *etc*. In addition, the server will learn that the nearest data point to c is the query point q, the second nearest is the data point e, *etc*. This process will eventually reveal to the server quite a detailed map of possible positions of all the data points in the workspace.

Figure 2 illustrates a novel solution which will be called the *Basic Non-Index-based Encryption Scheme* since it does not require that any specialised data indexing method be used to compute its results. It will be proved that the new scheme does not reveal to the server anything other than the simple fact that if the query point q is among the k nearest neighbours of every data point, which is the only information that the server needs to obtain in order to provide and send to the client the correct encrypted results.

Private Keys: two $(d + 1) \times (d + 1)$ invertible matrices M_1 and M_2.

Data encryption functions: $\forall$ d-dimensional data point p $(p_1, p_2, \ldots, p_d)$, three encrypted versions p', ps' and pss' are computed, using the functions $p' = E(p) = M_1^{-1} * \hat{p}$, $ps' = E_s(p) = M_1^{T} * \widehat{ps}$ and $pss' = E_{ss}(p) = M_2^{T} * \widehat{pss}$, where $\hat{p}$, $\widehat{ps}$ and $\widehat{pss}$ are the $(d + 1)$-dimensional vectors $\hat{p} = [p_1, p_2, \ldots, p_d, -0.5p^2]^T$ and $\widehat{ps} = \widehat{pss} = [s{\cdot}p_1, s{\cdot}p_2, \ldots, s{\cdot}p_d, s]^T$, and s is a random positive number. The encrypted values $<p', ps', pss'>$ are then sent to the server.

Query encryption function: $\forall$ query point q $(q_1, q_2, \ldots, q_d)$, its encrypted version q' is computed using the function $q' = E_q(q) = M_2^{-1} * \hat{q}$, where $\hat{q}$ is the $(d + 1)$-dimensional vector $\hat{q} = [q_1, q_2, \ldots, q_d, -0.5q^2]^T$. The encrypted value q' is then sent to the server.

Distance comparison operation: assuming the encrypted versions of two data points p and r and that of a query point q, in order to determine whether q is nearer to p than r is, the server needs to check whether the inequality $pss'^{T} * q' \geq ps'^{T} * r'$ holds.

Data decryption function: assuming an encrypted data point pss', the vector $\widehat{pss}$ is decrypted using the function $\widehat{pss} = E_{ss}^{-1}(pss') = (M_2^{T})^{-1} * pss'$. The coordinates of p are finally extracted after dividing $\widehat{pss}$ with the positive number s, the value of which can be found in the $(d + 1)$-th position of the vector $\widehat{pss}$.

Figure 2: The Basic Non-Index-based Encryption Scheme.

***Theorem* 1.** The distance comparison operation of the Basic Non-Index-based Encryption Scheme illustrated in Figure 2 can correctly determine whether a data point $p \in RkNN\ (q)$.

Proof: assuming two data points p and r and a query point q, by starting with $pss'^{T} * q' \geq ps'^{T} * r'$ we get the inequality $E_{ss}(p)^{T} * E_q(q) \geq E_s(p)^{T} * E(r) \Leftrightarrow (M_2^{T} * \widehat{pss})^{T} * (M_2^{-1} * \hat{q}) \geq (M_1^{T} * \widehat{ps})^{T} * (M_1^{-1} * \hat{r}) \Leftrightarrow (\widehat{ps}^{T} * M_2) * (M_2^{-1} * \hat{q}) \geq (\widehat{ps}^{T} * M_1) * (M_1^{-1} * \hat{r}) \Leftrightarrow \widehat{pss}^{T} * \hat{q} \geq \widehat{ps}^{T} * \hat{r} \Leftrightarrow [s{\cdot}p_1, \ldots, s{\cdot}p_d, s] * [q_1, \ldots, q_d, -0.5q^2]^T \geq [s{\cdot}p_1, \ldots, s{\cdot}p_d, s] * [r_1, \ldots, r_d, -0.5r^2]^T \Leftrightarrow s{\cdot}p_1{\cdot}q_1 + \ldots + s{\cdot}p_d{\cdot}q_d - s0.5q^2 \geq s{\cdot}p_1{\cdot}r_1 + \ldots + s{\cdot}p_d{\cdot}p_d - s0.5p^2 \Leftrightarrow 2s{\cdot}p_1{\cdot}q_1 + \ldots + 2s{\cdot}p_d{\cdot}q_d - s(q_1^2 + \ldots + q_d^2) - s(p_1^2 + \ldots + p_d^2) \geq 2s{\cdot}p_1{\cdot}r_1 + \ldots + 2s{\cdot}p_d{\cdot}r_d - s(r_1^2 + \ldots + r_d^2) - s(p_1^2 + \ldots + p_d^2) \Leftrightarrow -s(p_1^2 - 2p_1{\cdot}q_1 + q_1^2) - \ldots - s(p_d^2 - 2p_d{\cdot}q_d + q_d^2) \geq -s(p_1^2 - 2p_1{\cdot}r_1 + r_1^2) - \ldots - s(p_d^2 - 2p_d{\cdot}r_d + r_d^2) \Leftrightarrow (p_1 - q_1)^2 + \ldots + (p_d - q_d)^2 \leq (p_1 - r_1)^2 + \ldots + (p_d - r_d)^2 \Leftrightarrow dist\ (p, q) \leq dist\ (p, r)$.

Therefore the distance comparison operation of the Basic Non-Index-based Encryption Scheme is equivalent to the inequality $dist\ (p, q) \leq dist\ (p, r)$. Thus, if the server compares the distance $dist\ (p, q)$ between a data point p and a query point q, with the $dist\ (p, r)$ distance between p and every other data point r in the dataset, it can be easily determined whether q is among the k nearest neighbours of p, therefore if $p \in RkNN\ (q)$. □

Wong et al. [8] have proved that the preservation through the encryption scheme of the scalar product between two data points reveals the distance between the data points. However, the encryption scheme proposed here does not preserve this undesirable property because, for every data points p and r, the scalar product $ps'^{T} * r' = E_s(p)^{T} * E(r) = (M_1^{T} * \widehat{ps})^{T} * (M_1^{-1} * \hat{r}) = \widehat{sp}^{T} * \hat{r} = -0.5s(r^2 - 2p_1{\cdot}r_1 - \ldots - 2p_d{\cdot}r_d)$ is different from the scalar product $p^T * r = p_1{\cdot}r_1 + \ldots + p_d{\cdot}r_d$. Also, for every data point p, the scheme does not preserve the scalar product $p^T * p$ between the data point and itself, since $ps'^{T} * p' = 0.5s{\cdot}p^2$, which is different from the scalar product $p^T * p = p^2$, bearing in mind that s is an unknown parameter for the server. The preservation of this product would also partially compromise the security of the model because it would reveal to the observer that p lies on a hyper-sphere that is centred at the origin of the space with a $\sqrt{p_1^2 + \ldots + p_d^2}$ radius. And last, it is not possible for someone using the proposed scheme to determine the values of the data point p and of the query point q by knowing only the values of p', ps', pss' and q' if s/he does not also know the values of M_1 and/or M_2.

In addition, by using a different function E_q to encrypt the query points than the functions E, Es and Ess to encrypt the data points, it is ensured that the encrypted version q' of a query point q will not coincide with the corresponding encrypted versions p', ps' and pss' of any data point p, when q coincides with p, in which case the server would be able to compute the order of the distances between all the other data points and p. Also, by encrypting the query points and the data points using different encryption functions, in the distance comparison operation the server cannot replace the encrypted query point with an encrypted data point, in order to eventually discover the order of distances between all the data points.

With regard to attacks, the following theorem holds:

Theorem 2: The Basic Non-Index-based Scheme is not secure against the known input-output attack.

Proof: Assuming that the observer knows both the plaintext p and the ciphertext values $<p', ps', pss'>$ of a data point (the case of the knowledge of the plaintext of a query point q and of its corresponding encrypted version q' is similar), the observer can use the equalities $p' = M_1^{-1} * \hat{p}$ and $ps' = M_1^{T} * \widehat{ps}$ and $pss' = M_2^{T} * \widehat{pss} = M_2^{T} * \widehat{ps}$ in order to construct a set of $3(d + 1)$ equations, in which s/he will have $2(d + 1) \cdot (d + 1)$ unknown parameters in the encryption keys M_1 and M_2,[2] and one unknown parameter in the vector $\widehat{ps}$, *i.e.* the random positive value of s.

[2] we do not consider unknown parameters for the matrix M_1^{-1} since its formulation depends on the formulation of M_1.

By knowing, in total, n pairs of plaintext data points and their corresponding ciphertexts, the observer can construct a set of $n3(d + 1)$ equations with $2(d + 1)\cdot(d + 1) + n$ unknown parameters. It is thus clear that if $n3(d + 1) \geq 2(d + 1)\cdot(d + 1) + n \Rightarrow n \geq 2(d + 1)\cdot(d + 1)/(3d + 2)$, then the number of equations will be larger than the number of unknown parameters. Therefore the observer will be able to solve the system of linear equations to eventually find the secret keys M_1, M_2 and the plaintext of every ciphertext. □

4.1.2 The Enhanced Non-Index-based Encryption Scheme

To make the Basic Non-Index-based Encryption Scheme secure against the known input-output attack, this section presents the *Enhanced Non-Index-based Encryption Scheme*, which is a solution inspired from analogous schemes introduced in [4, 8] for the range and the k-nearest neighbours queries, respectively. The new solution suggests the random splitting of all the values in every $(d + 1)$-dimensional vector $\widehat{pss}$ and $\widehat{ps}$ (we remind here that $\widehat{pss} = \widehat{ps}$) in order to generate two random shares $\widehat{pssa}$ and $\widehat{pssb}$ (resp. $\widehat{psa}$ and $\widehat{psb}$), such that $\forall\ i \in \{1, ..., d + 1\}$ we will have $\widehat{pss}_i = \widehat{pssa}_i + \widehat{pssb}_i$ (resp. $\widehat{pss}_i = \widehat{psa}_i + \widehat{psb}_i$). Therefore the product $\widehat{pss}^T * \hat{q}$ (resp. $\widehat{ps}^T * \hat{r}$) will be equal to $\widehat{pssa}^T * \hat{q} + \widehat{pssb}^T * \hat{q}$ (resp. $\widehat{psa}^T * \hat{r} + \widehat{psb}^T * \hat{r}$). The vector $\widehat{pssa}$ (resp. $\widehat{psa}$) will be then encrypted with the secret key Ma_1^T (resp. Ma_2^T) and the vector $\widehat{pssb}$ (resp. $\widehat{psb}$) with the secret key Mb_1^T (resp. Mb_2^T) in order to produce the vectors $pssa'$ and $pssb'$ (resp. psa' and psb'). In addition, every vector $\hat{p}$ will be encrypted twice using the matrices Ma_1^{-1} and Mb_1^{-1}, in order to produce the vectors pa' and pb', respectively. Every query point vector $\hat{q}$ will also be encrypted twice, using the matrices Ma_2^{-1} and Mb_2^{-1}, in order to produce the vectors qa' and qb', respectively. The final distance comparison operation will be performed using the equation $pssa'^T * qa' + pssb'^T * qb' \geq psa'^T * ra' + psb'^T * rb' \Leftrightarrow (Ma_2^T * \widehat{pssa})^T * Ma_2^{-1} * \hat{q} + (Mb_2^T * \widehat{pssb})^T * Mb_2^{-1} * \hat{q} \geq (Ma_1^T * \widehat{psa})^T * Ma_1^{-1} * \hat{r} + (Mb_1^T * \widehat{psb})^T * Mb_1^{-1} * \hat{r} \Leftrightarrow \widehat{pssa}^T * \hat{q} + \widehat{pssb}^T * \hat{q} \geq \widehat{psa}^T * \hat{r} + \widehat{psb}^T * \hat{r} \Leftrightarrow \widehat{pss}^T * \hat{q} \geq \widehat{ps}^T * \hat{r}$, for which the proof of Theorem 1 shows that it is equivalent to the inequality $dist(p, q) \leq dist(p, r)$.

However to achieve a higher level of security, instead of splitting the values in the vectors $\widehat{pss}$ and $\widehat{ps}$, the method can split the values of $\widehat{pss}$ and $\widehat{ps}$ in only some of the rows, together with the values in some other rows of $\hat{q}$ and $\hat{p}$, respectively (however, we cannot split the values in the same rows in all $\widehat{pss}$, $\widehat{ps}$, $\hat{q}$ and $\hat{p}$ at the same time). Therefore, during the encryption, the client will need to store two configuration bit strings Sss and Ss, which are $(d + 1)$-bits vectors, with every entry in Sss (resp. Ss) indicating whether pss-splitting or q-splitting (resp. ps-splitting or p-splitting) is used for the corresponding row in $\widehat{pss}$ and $\hat{q}$ (resp. $\widehat{ps}$ and $\hat{p}$). Since there are $2^{(d+1)}$ possible configurations per $(d + 1)$-bits vector, the Enhanced Non-Index-based Scheme is more secure against attacks in comparison to the Basic Non-Index-based Scheme.

To boost security further, the solution suggests the increase of the d number of dimensions by adding artificial attributes to all $\hat{p}$, $\widehat{ps}$, $\widehat{pss}$ and $\hat{q}$. This can be achieved by extending every $d + 1$ column vector to a $d' + 1$ column vector by padding artificial values such that the scalar product over the added attribute values will be 0.

Figure 3 summarizes the procedures implementing the proposed Enhanced Non-Index-based Scheme.

Private Keys: four $(d' + 1) \times (d' + 1)$ invertible matrices Ma_1, Mb_1, Ma_2 and Mb_2, two configuration $(d' + 1)$-bits strings Ss and Sss and two series of $d' - d$ pre-generated random numbers ws_{d+2}, ..., $ws_{d'+1}$ and wss_{d+2}, ..., $wss_{d'+1}$.

Data encryption functions: $\forall$ d-dimensional data point p $(p_1, p_2, \ldots, p_d)$, six encrypted quantities pa', pb', psa', psb', $pssa'$ and $pssb'$ are computed, as follows:

- $pa' = Ma_1^{-1} * \widehat{pa}$ and $pb' = Mb_1^{-1} * \widehat{pb}$, where $\widehat{pa} + \widehat{pb} = \hat{p}$,
- $psa' = Ma_1^T * \widehat{psa}$ and $psb' = Mb_1^T * \widehat{psb}$, where $\widehat{psa} + \widehat{psb} = \widehat{ps}$ and
- $pssa' = Ma_2^T * \widehat{pssa}$ and $pssb' = Mb_2^T * \widehat{pssb}$, where $\widehat{pssa} + \widehat{pssb} = \widehat{pss}$.

The parameters $\hat{p}$, $\widehat{ps}$, and $\widehat{pss}$ are $(d' + 1)$-dimensional vectors. The first $d + 1$ dimensions of these are the same as in the corresponding vectors $\hat{p}$, $\widehat{ps}$ and $\widehat{pss}$ in the Basic Non-Index-based Scheme in Figure 2 and for the remainder $i = d + 2$ to $i = d' + 1$ dimensions, if $Ss_i = 1$ then $\hat{p}_i = ws_i$ and $\widehat{ps}_i$ is set to be equal to a random number (resp. if $Sss_i = 1$ then $\widehat{pss}_i$ is set to be equal to a random number), otherwise if $Ss_i = 0$ then $\hat{p}_i$ is set to be equal to a random number and $\hat{p}s_i = ws_i$ (resp. if $Sss_i = 0$ then $\widehat{pss}_i = wss_i$). For the last dimension with which $Ss_i = 0$, $\hat{p}_i$ is given a value so that the scalar product over its artificial attributes $d + 2$ to $d' + 1$ is 0 (see [8] for more details). Also, for the last dimension with which $Ss_i = 1$ (resp. $Sss_i = 1$), $\widehat{ps}_i$ (resp. $\widehat{pss}i$) is given a value so that the scalar product over its artificial attributes $d + 2$ to $d' + 1$ is 0. Additionally, for $\forall$ i, if $Ss_i = 1$, then the value of $\hat{p}_i$ is randomly split into $\widehat{pa}_i$ and $\widehat{pb}_i$. In this case, $\widehat{psa}_i$ and $\widehat{psb}_i$ are both set to be equal to $\widehat{ps}_i$ (resp. if $Sss_i = 1$ then $\widehat{pssa}_i$ and $\widehat{pssb}_i$ are both set to be equal to $\widehat{pss}_i$). Otherwise, if $Ss_i = 0$ then $\widehat{pa}_i$ and $\widehat{pb}_i$ are both set to be equal to $\hat{p}_i$ and the value of $\widehat{ps}_i$ is randomly split into $\widehat{psa}_i$ and $\widehat{psb}_i$ (resp. if $Sss_i = 0$ then the value of $\widehat{pss}_i$ is randomly split into $\widehat{pssa}_i$ and $\widehat{pssb}_i$). The encrypted values $<(pa', pb'), (psa', psb'), (pssa', pssb')>$ are sent to the server.

Query encryption functions: $\forall$ query point q $(q_1, q_2, \ldots, q_d)$, two encrypted parts qa' and qb' are computed using the functions $qa' = Ma_2^{-1} * \widehat{qa}$ and $qb' = Mb_2^{-1} * \widehat{qb}$, where $\widehat{qa} + \widehat{qb} = \hat{q}$ and $\hat{q}$ is a $(d' + 1)$-dimensional vector, in which the first $d + 1$ dimensions have the same value as the corresponding vector $\hat{q}$ in the Basic Non-Index-based Encryption Scheme in Figure 2 and for the remainder $i = d + 2$ to $i = d' + 1$ dimensions, if $Sss_i = 1$ then $\hat{q}_i = wss_i$, otherwise $\hat{q}_i$ is set to be equal to a random number. For the last dimension with which $Sss_i = 0$, $\hat{q}_i$ is given a value so that the scalar product over the artificial values $i = d + 2$ to $i = d' + 1$ is 0. Additionally, $\forall$ i, if $Sss_i = 1$ then the value of $\hat{q}_i$ is randomly split into $\widehat{qa}_i$ and $\widehat{qb}_i$. Otherwise, if $Sss_i = 0$ then $\widehat{qa}_i$ and $\widehat{qb}_i$ are both set to be equal to $\hat{q}_i$. The encrypted values $<qa', qb'>$ are sent to the server.

Distance comparison operation: assuming the encrypted tuples of two data points p and r and of a query point q, in order to determine whether q is nearer to p than r is, the server needs to check whether the inequality $pssa'^T * qa' + pssb'^T * qb' \geq psa'^T * ra' + psb'^T * rb'$ holds.

Data decryption function: assuming an encrypted data point in the form $<pssa', pssb'>$, the vector $\widehat{pss}$ is decrypted using the function $\widehat{pss} = \widehat{pssa} + \widehat{pssb} = (Ma_2^T)^{-1} * pssa' + (Mb_2^T)^{-1} * pssb'$. The coordinates of p are finally extracted after dividing $\widehat{pss}$ with the positive number s, the value of which can be found in the $(d + 1)$-th position of the vector $\widehat{pss}$.

Figure 3: The Enhanced Non-Index-based Encryption Scheme.

Theorem 3: The Enhanced Scheme is secure against the known input-output attack if the sample that is known to the observer is made up of pairs of plaintext and ciphertext tuples of data points. It is not secure if the known sample is made up of pairs of plaintext and ciphertext tuples of the client's queries.

Proof*:* For the sake of simplicity, we shall assume that $d' = d$, *i.e.* that no artificial dimensions have been added (the addition of artificial attributes can only make the scheme more secure). Depending on the external knowledge available, the observer might issue a known input-output attack targeting the secret keys Ma_1, Mb_1, Ma_2 and Mb_2 or targeting directly the plaintext dataset.

In the *first case* we shall assume that the observer knows a sample of n pairs of plaintext and ciphertext tuples of data points. For every such pair of a plaintext p and its corresponding tuple of encrypted values $<pa', pb', psa', psb', pssa', pssb'>$, the observer can use the following six equalities:

- $pa' = Ma_1^{-1} * \widehat{pa}$ and $pb' = Mb_1^{-1} * \widehat{pb}$
- $psa' = Ma_1^{T} * \widehat{psa}$ and $psb' = Mb_1^{T} * \widehat{psb}$, and
- $pssa' = Ma_2^{T} * \widehat{pssa}$ and $pssb' = Mb_2^{T} * \widehat{pssb}$,

in order to construct a set of $6(d + 1)$ equations, in which s/he will have $4(d + 1)\cdot(d + 1)$ unknown parameters in the encryption keys Ma_1, Mb_1, Ma_2 and Mb_2, and $6(d + 1)$ unknown parameters in the vectors $\widehat{pa}$, $\widehat{pb}$, $\widehat{psa}$, $\widehat{psb}$, $\widehat{pssa}$ and $\widehat{pssb}$ since s/he does not know how these vectors have been created, *i.e.* s/he does not know the configuration bit strings Ss and Sss.[3]

By knowing in total n pairs of plaintext data points and their corresponding ciphertexts, the observer can construct a set of $n6(d + 1)$ equations with $4(d + 1)\cdot(d + 1) + n6(d + 1)$ unknown parameters. It becomes thus clear that, independently of the value of n, there will always be $4(d + 1)\cdot(d + 1)$ more unknown parameters than there are equations, preventing the observer from working out the system of linear equations and from finding the secret keys Ma_1, Mb_1, Ma_2 and Mb_2. Thus, the scheme can guarantee protection against the input-output attack if the known sample is made up of pairs of plaintext and ciphertext tuples of data points.

In the *second case* it will be assumed that the observer knows a sample of n pairs of plaintext and ciphertext of the client's queries. Two sub-cases need to be examined here. If the observer targets the queries encryption keys Ma_2^{-1} and Mb_2^{-1}, then a study similar to that of the known sample of pairs of plaintext and ciphertext data tuples conducted above will prove that the proposed model is secure. However in the case of a known sample of pairs of plaintext and ciphertext queries, the observer can also target directly the plaintext coordinates of the data points without needing to work out the encryption keys beforehand. Assuming an unknown data point p and a known query point q, the observer can use the equation $pssa'^{T} * qa' + pssb'^{T} * qb' = \widehat{pss}^{T} * \hat{q}$ in order to construct one equation, in which s/he will have $d + 1$ unknown parameters in the vector $\widehat{pss}$ (*i.e.* the coordinates of the data point p and the random number s).

By knowing in total n pairs of query points and their corresponding encrypted versions, the observer can construct a system of n equations with $d + 1$ unknown parameters in order to derive the coordinates of p. It is thus clear that if $n \geq d + 1$ then the number of equations will be larger than the number of unknown parameters. Therefore the observer will be able to solve the system of linear equations and find the coordinates of p. Thus, the scheme is not secure against the query input-output attack. □

4.2 Solutions that rely on data indexing mechanisms

4.2.1 The Basic Index-based Encryption Scheme

Since the encryption schemes that do not rely on any sophisticated indexing mechanism need to compare the distance $dist(p, q)$ between every data point p to the query point q against the distance $dist(p, r)$ of p to every other data point r in the dataset, the performance cost of these schemes in respect of processing the RkNN query is analogous to $O(n^2)$, where n is the dataset cardinality. In order to reduce this cost, and especially with regard to large-scale databases, solutions will be proposed that rely on data indexing methods so that the RkNN query results can be produced faster. The schemes will put forward a secure version of the pioneer RkNN method proposed in [21]. A secure version of the SS-tree [28] structure, called the *S^3-tree* (*Secure SS-tree*), will serve as the backbone spatial indexing method for processing the query on the server-side. The description of the S^3-tree is as follows:

- Every node contains between b and B entries, unless it is the root node of the tree. The root has at least two entries, unless it is a leaf.
- Every leaf entry U is a minimum bounding hyper-sphere (MBS) which corresponds to a vicinity sphere centred at a specific d-dimensional data point p $(p_1, p_2, \ldots, p_d)$ with a radius l that is equal to the Euclidean distance between p and its k-th nearest neighbour data point. The entry U is stored in the server in its encrypted form U', using the function $U' = E(p, l) = M^{T} * \hat{U}$, where M is a $3d \times 3d$ invertible matrix that serves as the encryption key, and $\hat{U}$ is the $3d$-dimensional vector $\hat{U} = [s_p, s_p \cdot p_1, s_p(p_1^2 - l^2/d), \ldots, s_p, s_p \cdot p_d, s_p(p_d^2 - l^2/d)]^T$, in which s_p is a random positive number.
- For every non-leaf entry $<ptr, U>$, ptr is a pointer to a child node and U is the MBS that completely encloses the hyper-spheres in that descendant node. The MBS U is represented using its centroid point c $(c_1, \ldots, c_d)$ and its radius l. The entry U is stored in the server in its encrypted form U', using the same function as in the corresponding case of the leaf entries.
- All the leaves of the tree appear at the same level.

The difference between the traditional SS-tree and the S^3-tree lies, therefore, in the fact that the latter stores encrypted MBSs instead of plaintext MBSs in its leaves and in its non-leaf nodes. While

[3] it must be pointed out that, in the case of a brute-force approach the observer would need to examine all possible bit vectors Ss and Sss, which would lead to $(2^{(d+1)})\cdot(2^{(d+1)})$ candidate systems of linear equations, that cannot be expected to be solved in reasonable time, especially if d is a large number. However, even if the observer manages to work out these systems, they will produce $(2^{(d+1)})\cdot(2^{(d+1)})$ candidate quadruplets of encryption keys $<Ma_1, Mb_1, Ma_2, Mb_2>$, with a $1/((2^{(d+1)})\cdot(2^{(d+1)}))$ probability that any single one of these might be the correct one.

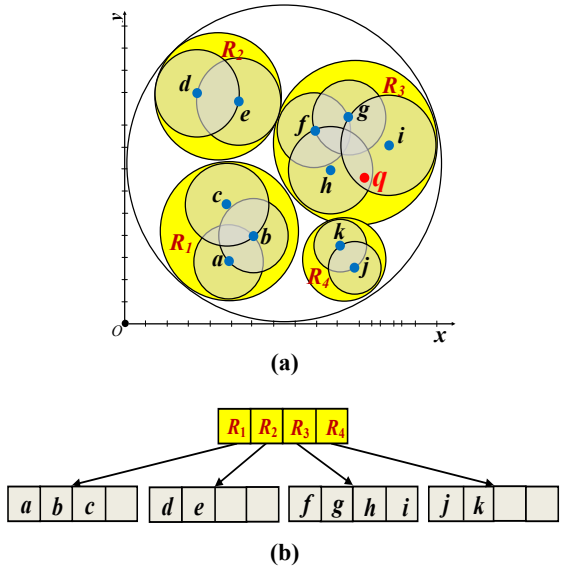


Figure 4. (a) a set of eleven two-dimensional data points and (b) the S^3-tree for k = 1 built on top of these objects.

the S^3-tree was selected as the backbone indexing model, several other data partitioning indexing methods can also be used instead for medium- or high- dimensional workspaces, with the proper adjustment.

Using the S^3-tree, the RkNN of a query point q can be efficiently retrieved using a point location query which will target the hyper-spheres that contain q. Figure 4 illustrates several data points in the two-dimensional space together with their MBSs, and the corresponding S^3-tree for k = 1 built on top of these MBSs, assuming a minimum and maximum node capacity of b = 2 and B = 4 records, respectively (for the sake of simplicity, the data points are shown in the leaves of the S^3-tree, instead of their encrypted MBSs, which are omitted). The R1NN query for the given point q in Figure 4(a) will return the data points $R1NN(q) = \{h, i\}$, because q falls inside the circles of h and i. The correction study of the above strategy is provided in [21]. Figure 5 summarizes the procedures implementing the proposed *Basic Index-based Encryption Scheme*.

***Theorem* 4.** The range enclosure operation of the Basic Index-based Encryption Scheme illustrated in Figure 5 can correctly determine whether a data point $p \in RkNN(q)$.

Proof: Assuming that $q' = E_q(q)$ is the encrypted version of the d-dimensional query point q, for every level of the S^3-tree, the server will use the range enclosure operation in Figure 5 in order find out if q lies inside any of the MBSs U in the S^3-tree, where U is centered at the point c (c_1, …, c_d) and having a radius l. By starting with $U'^T * q' \le 0$ we get $E(c, l)^T * E_q(q) \le 0 \Leftrightarrow (M^T * \hat{U})^T * (M^{-1} * \hat{q}) \le 0 \Leftrightarrow \hat{U}^T * M * M^{-1} * \hat{q} \le 0 \Leftrightarrow \hat{U}^T * \hat{q} \le 0 \Leftrightarrow [s_p, s_p \cdot c_1, s_p(c_1^2 - l^2/d), \ldots, s_p, s_p \cdot c_d, s_p(c_d^2 - l^2/d)] * [s_q \cdot q_1^2, -2s_q \cdot q_1, s_q, \ldots, s_q \cdot q_d^2, -2s_q \cdot q_d, s_q]^T \le 0 \Leftrightarrow s_p s_q((q_1^2 - 2c_1 \cdot q_1 + (c_1^2 - l^2/d)) + \ldots + (q_d^2 - 2c_d \cdot q_d + (c_d^2 - l^2/d))) \le 0 \Leftrightarrow (q_1 - c_1)^2 + \ldots + (q_d - c_d)^2 \le l^2 \Leftrightarrow dist^2(q, c) \le l^2 \Leftrightarrow dist(q, c) \le l$.

Private Key: a $3d \times 3d$ invertible matrix M.

Data encryption functions:

- In the leaf nodes of the S^3-tree, every d-dimensional MBS U centered at a specific data point p (p_1, p_2, …, p_d) with a radius l that is equal to the Euclidean distance between p and its k-th nearest neighbour data point, is encrypted using the function $U' = E(p, l) = M^T * \hat{U}$, where $\hat{U}$ is the $3d$-dimensional vector of the form $\hat{U} = [s_p, s_p \cdot p_1, s_p(p_1^2 - l^2/d), \ldots, s_p, s_p \cdot p_d, s_p(p_d^2 - l^2/d)]^T$, in which s_p is a random positive number.
- In the non-leaf nodes of the S^3-tree, every d-dimensional MBS U centered at the point c (c_1, …, c_d) and having a radius l is encrypted using the same function $U' = E(c, l) = M^T * \hat{U}$ as in the leaf nodes.

Query encryption function: $\forall$ query point q (q_1, q_2, …, q_d), its encrypted version q' are computed using the function $q' = E_q(q) = M^{-1} * \hat{q}$, where $\hat{q}$ is a $3d$-dimensional vector $\hat{q} = [s_q \cdot q_1^2, -2s_q \cdot q_1, s_q, \ldots, s_q \cdot q_d^2, -2s_q \cdot q_d, s_q]^T$, in which s_q is a random positive number.

Range enclosure operation: On every level of the S^3-tree, assuming $U' = E(c, l)$ and $q' = E_q(q)$, in order to determine whether the query point q lies within a hyper-sphere U centered at a point c with a radius l, the server needs to check whether $U'^T * q' \le 0$ holds.

Data decryption function: assuming an encrypted MBS U', the vector $\hat{U}$ is decrypted using the function $\hat{U} = E^{-1}(U') = (M^T)^{-1} * U'$. The coordinates of the centroid data point p of the MBS U are finally extracted after dividing $\hat{U}$ by the positive number s_p, the value of which can be found in the first cell of $\hat{U}$.

Figure 5. The Basic Index-based Encryption Scheme.

Therefore, the evaluation $U'^T * q' \le 0$ is equivalent to checking whether q lies within the hyper-sphere U. It must be kept in mind that at the leaf level of the tree, the centre point of U is a data point p and the radius l is equal to the Euclidean distance between p and its k-th nearest neighbour data point. Therefore the Basic Index-based Encryption Scheme can correctly determine whether q is among the k nearest neighbours of p, therefore if $p \in RkNN(q)$. □

With regard to the scheme's resistance to attacks, the following theorem holds.

Theorem 5: The Basic Index-based Encryption Scheme is not secure against the input-output attack.

Proof: Assuming that the observer has knowledge of the plaintext p and of the radius l of its vicinity hyper-sphere U, as well as of the corresponding ciphertext U' (the case of the knowledge of a query point q and of its corresponding encrypted version q' is similar), the observer can use the equality $U' = M^T * \hat{U}$ in order to construct a system of $3d$ equations, in which s/he will have $(3d) \cdot (3d)$ unknown parameters in the encryption key M and one unknown parameter in $\hat{U}$, *i.e.* the random number s_p.

By knowing, in total, n pairs of plaintext data points (together with their vicinity hyper-spheres) and their corresponding ciphertexts, the observer can construct a set of $n3d$ equations with $3d3d + n$ unknown parameters. It is thus clear that, if $n3d \ge (3d) \cdot (3d) + n \Rightarrow n \ge 9d^2/(3d - 1)$, the number of equations will be larger than the number of unknown parameters. Therefore the observer will be able to solve the system of linear equations and will eventually work out the secret key M and the plaintext of every ciphertext. □

4.2.2 The Enhanced Index-based Encryption Scheme

To make the Basic Index-based Encryption Scheme secure against known input-output attacks, this section suggests the random splitting of all the values in every $3d$-dimensional column vector $\widehat{U}$ in order to generate two random shares for each, *i.e.* $\widehat{Ua}$ and $\widehat{Ub}$, such that $\forall\ i \in \{1, ..., 3d\}$ we will have $\widehat{U}_i = \widehat{Ua}_i + \widehat{Ub}_i$. Therefore the product $\widehat{U}^T * \hat{q}$ will be equal to $\widehat{Ua}^T * \hat{q} + \widehat{Ub}^T * \hat{q}$, respectively. The vector $\widehat{Ua}$ will be then encrypted with the secret key Ma^T in order to produce the vector Ua', and the vector $\widehat{Ub}$ will be encrypted with the secret key Mb^T in order to produce the vector Ub'. In addition, every query point vector $\hat{q}$ will be encrypted twice, using the matrices Ma^{-1} and Mb^{-1}, in order to produce the vectors qa' and qb', respectively. The final range enclosure evaluation will be performed using the equation:

$Ua'^T * qa' + Ub'^T * qb' \leq 0 \Leftrightarrow (Ma^T * \widehat{Ua})^T * Ma^{-1} * \hat{q} + (Mb^T * \widehat{Ub})^T * Mb^{-1} * \hat{q} \leq 0 \Leftrightarrow \widehat{Ua}^T * \hat{q} + \widehat{Ub}^T * \hat{q} \leq 0 \Leftrightarrow \widehat{U}^T * \hat{q} \leq 0.$

For a higher level of security, instead of splitting the values in the vector $\widehat{U}$, the method can split the values in some of the rows of $\widehat{U}$ and the values in some other rows of $\hat{q}$. Therefore, during the encryption phase, the client will need to store a configuration bit string S, which is a $3d$-bits vector, with every entry in it indicating whether data-splitting or query-splitting is used for the corresponding row in $\widehat{U}$ and $\hat{q}$. Since there are 2^{3d} possible configurations of the bit string S, the *Enhanced Index-based Encryption Scheme* is more secure against attacks as compared to the Basic Index-based Encryption Scheme.

To boost security further, we can once again increase the number of dimensions, from d to d', by adding artificial attributes to $\widehat{U}$ and $\hat{q}$, so that the scalar product between $\widehat{U}$ and $\hat{q}$ over the added attribute values will be 0.

Figure 6 summarizes the procedures implementing the proposed Enhanced Index-based Encryption Scheme.

Theorem 6: The Enhanced Index-based Encryption Scheme is secure against the input-output attack.

Proof: It will be assumed that $d' = d$, *i.e.* that no artificial dimensions have been added.

In the *first case* we will assume that the observer has knowledge of a sample of n pairs of plaintext and ciphertext data tuples. For every such tuple of a plaintext p together with the radius l of its vicinity hyper-sphere U and the corresponding ciphertext U', the observer can use the equalities $Ua = Ma^T * \widehat{Ua}$ and $Ub = Mb^T * \widehat{Ub}$ in order to construct a set of $2(3d)$ equations, in which s/he will have $2(3d)\cdot(3d)$ unknown parameters in the encryption keys Ma and Mb, $3d$ unknown parameters in the vector $\widehat{Ua}$ and $3d$ unknown parameters in the vector $\widehat{Ub}$ (since s/he does not know how these two vectors have been created, *i.e.* s/he does not know the configuration bit string S).

By knowing in total n pairs of plaintext data points (together with the radius of their vicinity hyper-spheres) and their corresponding ciphertexts, the observer can construct a set of $n2(3d)$ equations with $2(3d)\cdot(3d) + n3d + n3d$ unknown parameters. It is thus evident that, independently of the value of n, there will always be $2(3d)\cdot(3d)$ more unknown parameters than there are equations. Therefore, the observer will not be able to solve the system of linear equations in order to find the secret keys Ma and Mb. Thus the scheme can guarantee protection against the known input-output attack if the sample that is known to the observer is made up of pairs of plaintext and ciphertext tuples of data points.

In the *second case* we will assume that the observer has knowledge of a sample of n pairs of plaintext and ciphertext of the client's queries. Two sub-cases need to be examined here. In the first sub-case, the observer may target the encryption keys Ma^{-1} and Mb^{-1}, in which sub-case, with a study similar to the one conducted above for the case of the known sample of pairs of plaintext and ciphertext data tuples, we can prove that the proposed model is secure. In the second sub-case, the observer may target directly the plaintext coordinates of the data points, without needing to solve the encryption keys beforehand. Assuming the vicinity hyper-sphere U of an unknown data point p

Private Keys: two $3d' \times 3d'$ invertible matrices Ma and Mb, a configuration $3d'$-bits string S, and $3d' - 3d$ pre-generated random numbers $w_{3d+1}, w_{3d+2}, ..., w_{3d'}$.

Data encryption functions: In the leaf (resp. in the non-leaf) nodes of the S^3-tree, $\forall$ d-dimensional MBS U centred at a data point p ($p_1, p_2, \ldots, p_d$) with a radius l that is equal to the Euclidean distance between p and its k-th nearest neighbour data point (resp. centered at the point c ($c_1, \ldots, c_d$) with a radius l), two encrypted parts Ua' and Ub' are computed using the functions $Ua' = Ma^T * \widehat{Ua}$ and $Ub' = Mb^T * \widehat{Ub}$, where $\widehat{U} = \widehat{Ua} + \widehat{Ub}$ is a $3d'$-dimensional vector, in which the first $3d$ dimensions are the same as in the corresponding vector $\widehat{U}$ in the Basic Index-based Encryption Scheme in Figure 5 and for the remainder $i = 3d + 1$ to $i = 3d'$ dimensions, if $S_i = 0$ then $\widehat{U}_i = w_i$, otherwise $\widehat{U}_i$ is set to be equal to a random number. For the last dimension for which $S_i = 1$, $\widehat{U}_i$ is given a value so that the scalar product over its artificial dimensions $3d + 1$ to $3d'$ is 0. Additionally, $\forall$ i, if $S_i = 1$ then the value of $\widehat{U}_i$ is randomly split into $\widehat{Ua}_i$ and $\widehat{Ub}_i$. Otherwise, if $S_i = 0$ then $\widehat{Ua}_i$ and $\widehat{Ub}_i$ are both set to be equal to $\widehat{U}_i$.

Query encryption function: $\forall$ query point q ($q_1, q_2, \ldots, q_d$), two encrypted parts qa' and qb' are computed using the functions $qa' = Ma^{-1} * \widehat{qa}$ and $qb' = Mb^{-1} * \widehat{qb}$, where $\widehat{qa} + \widehat{qb} = \hat{q}$ and $\hat{q}$ is a $3d'$-dimensional vector, in which the first $3d$ dimensions have the same value as the corresponding vector $\hat{q}$ in the Basic Index-based Encryption Scheme in Figure 5 and for the remainder $i = 3d + 1$ to $i = 3d'$ dimensions, if $S_i = 1$ then $\hat{q}_i = w_i$, otherwise $\hat{q}_i$ is set to be equal to a random number. For the last dimension with which $S_i = 0$, $\hat{q}_i$ is given a value so that the scalar product over the artificial attributes $i = 3d + 1$ to $i = 3d'$ is 0. Additionally, $\forall$ i, if $S_i = 0$ then the value of $\hat{q}_i$ is randomly split into $\widehat{qa}_i$ and $\widehat{qb}_i$. Otherwise, if $S_i = 1$ then $\widehat{qa}_i$ and $\widehat{qb}_i$ are both set to be equal to $\hat{q}_i$.

Range enclosure operation: In every level of the S^3-tree, assuming $U' = E(p, l) = Ua' + Ub'$ and $q' = E_q(q) = qa' + qb'$, in order to determine whether q lies within a hyper-sphere U, the server needs to check whether $Ua'^T * qa' + Ub'^T * qb' \leq 0$ holds.

Data decryption function: assuming an encrypted MBS U' in the form $<Ua', Ub'>$, the vector $\widehat{U}$ is decrypted using the function $\widehat{U} = \widehat{Ua} + \widehat{Ub} = (Ma^T)^{-1} * Ua' + (Mb^T)^{-1} * Ub'$. The coordinates of the centroid data point p of the MBS U are finally extracted after dividing $\widehat{U}$ with the positive number s_p, the value of which can be found in the first cell of $\widehat{U}$.

Figure 6. The Enhanced Index-based Encryption Scheme.

and a known query point q, the observer can use the equation $Ua'^{T} * qa' + Ub'^{T} * qb' = \hat{U}^{T} * \hat{q}$ in order to construct one equation, in which s/he will have $d + 2$ unknown parameters in the vector $\hat{U}$ (*i.e.* the coordinates of a data point p, plus the radius l of its vicinity hyper-sphere and the random number s_p) and one unknown parameter in the vector $\hat{q}$ (*i.e.* the random number s_q).

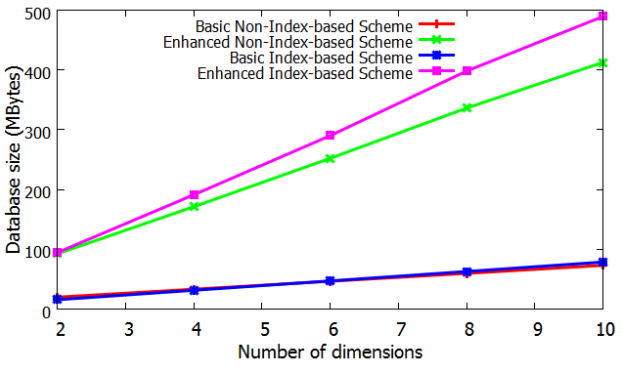
By knowing in total n pairs of query points and their corresponding encrypted versions, the observer can construct a system of n equations with $(d + 2) + n$ unknown parameters to derive the coordinates of p. Since independently of the value of n, there will always be $(d + 2)$ more unknown parameters than there are equations, the observer will not be able to solve the system of linear equations in order to find the coordinates of p. Thus the scheme can guarantee protection against the input-output attack. □

5 EXPERIMENTAL EVALUATION

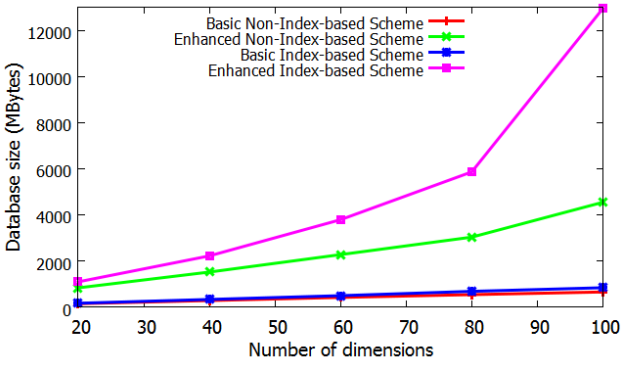
This section presents an experimental performance evaluation of a prototype implementation of the four proposed schemes in Java. The workstation that was used for simulating the two participants, *i.e.* the client and the server, was equipped with Intel Core i7 CPU running at 3.4GHz with 32GB RAM and Microsoft Windows 10 Professional 64-bit OS. The file system page size and the S^3-tree node size for the index-based encryption schemes is considered to be equal to 16 Kbytes. The experiments were conducted using two datasets, the Forest 'Covertype' dataset from the University of California Irvine machine learning repository [29], which is a real-life dataset for forest measurements counting 581,012 54-dimensional points (only the first 10 dimensions per data point are considered in the experiments because of the large number of zero values appearing in the remaining dimensions), as well as a synthetic dataset containing an equal number of 581,012 uniformly distributed 100-dimensional points.

Every one of the experiments was repeated ten times and the average value of every measured parameter was calculated at every run. In respect of the R*k*NN query processing on the server-side, a different randomly located query point was chosen with every renewed execution of the process. In the case of the two enhanced encryption schemes, the number of artificial dimensions $d' - d$ was set to be equal to twice the d number of dimensions of the actual data points, therefore $d' = 3d$. If the findings of the performance investigation of the four proposed schemes are comparable, irrespective of whether the real or the synthetic data are used, then only half of these (*i.e.* either with the real or with the synthetic data) are depicted.

The graphs of the first experiment in Figure 7 study the impact of the d number of data dimensions on the size of the database that is stored on the server-side for all four proposed schemes, using both the Covertype and the synthetic datasets. As a result of artificial dimensions having been added, the enhanced schemes appear to need more space to store the encrypted database than is needed by their counterpart basic schemes. According to Figure 2, the ciphertext of every data point in the Basic Non-Index-based Encryption Scheme is comprised of three encrypted $(d + 1)$-dimensional vectors, therefore its size is compa-



(a)

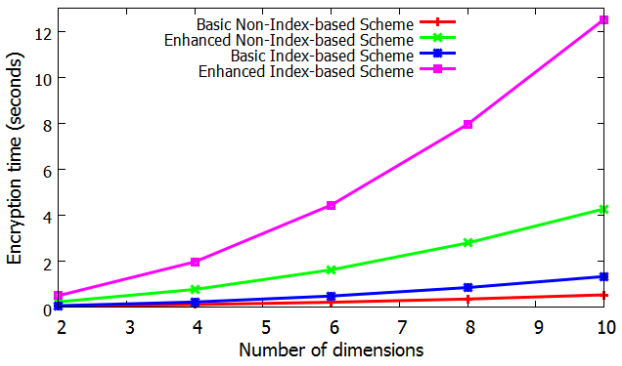


(b)

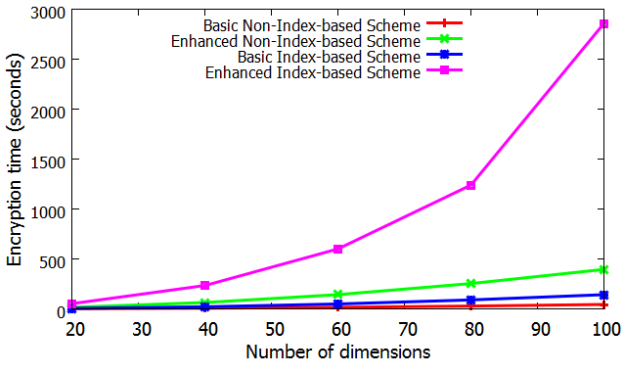
Figure 7: Database size: The impact of the number of dimensions on the database size using (a) real data and (b) synthetic data.

rable to the size of the ciphertext of every data point in the Basic Index-based Encryption Scheme, which, according to Figure 4, is comprised of one $3d$-dimensional vector. Similarly, the size of every encrypted data point in the Enhanced Non-Index-based and in the Enhanced Index-based Encryption Schemes is also comparable. On this basis, Figure 7 shows that, clearly, the Basic and Enhanced Index-based Encryption Schemes need more space to store the data than the corresponding Non-Index-based Encryption Schemes do, which difference in size is added for the storage of the S^3-tree index. It is also observed that the node utilization in the S^3-tree was about 85%, which is quite similar to that reported in [28] using plaintext data.

The next graphs in Figure 8 study the impact of the d number of data dimensions on the encryption time on the client-side for all four proposed schemes, using both the real-life and the synthetic datasets. The two graphs show an expected growth of the time cost as the number of data dimensions increases. The graphs also show that the Index-based Encryption Schemes need more time to encrypt the data in comparison to their counterpart Non-Index-based Encryption Schemes, which extra time is used to construct the encrypted index. It should be pointed out, however, that the data encryption time cost will be spent only once, which is in the beginning of the lifetime of every database.

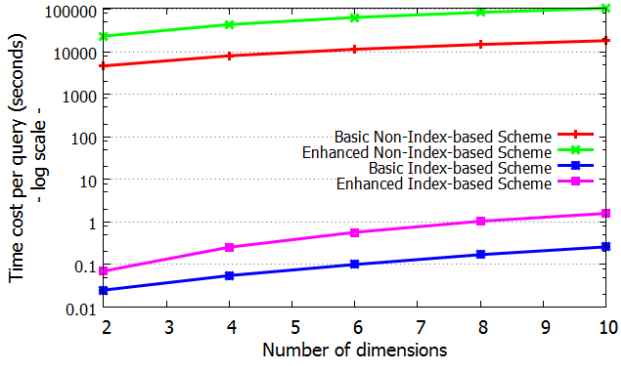


(a)

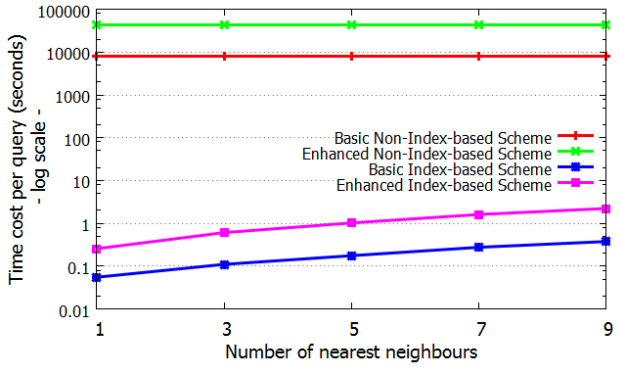


(b)

Figure 8: Data encryption: The impact of the number of dimensions on the data encryption time using (a) real data and (b) synthetic data.



(a)



(b)

Figure 9: RkNN query: (a) The impact of the number of dimensions, and (b) the impact of the number of nearest neighbours, in both cases, on the query execution time, using the real-life dataset.

The next two graphs in Figure 9 (please note the logarithmic scale on the *y*-axis in both graphs) indicate the time cost for processing the R*k*NN query on the server-side using the Covertype dataset. Figure 9(a) studies the impact of the d number of data dimensions on the query execution time for $k = 1$. The results in the graph indicate an expected growth of the time cost in relation to the increase of the number of data dimensions. The graph also points to the definite superiority of the Index-based Encryption Schemes in supporting the query, in comparison to the corresponding Non-Index-based Encryption Schemes, because the latter need $O(n^2)$ time to process the query, where n is the dataset size; while the Index-based Encryption Schemes can prune large parts of the dataset and they need to process only a small fraction of it.

Figure 9(b) studies the impact of the k number of nearest neighbours on the query execution time for all four proposed schemes. The results for the Non-Index-based Encryption Schemes indicate that the processing time cost is not affected by the value of k since, in every query evaluation, these schemes access all the points in the database. The results for the Index-based Encryption Schemes indicate a slight gradual increase of the processing time with the increase of the value of k. This happens because the increase of k increases the radius of the MBS of every data point in the S^3-tree (keeping in mind that the radius of the MBS is the Euclidean distance between the data point and its k-th nearest neighbour data point), therefore, the increase of k enlarges the overlap between the stored MBSs in the S^3-tree, which results in the R*k*NN query algorithm accessing more branches of the tree when processing the query. In this experiment the number of data dimensions is considered to be $d = 4$ (thus $d' = 12$).

6 CONCLUSIONS AND FUTURE RESEARCH

The paper proposes efficient solutions to evaluate securely R*k*NN query predicates directly over encrypted sensitive data in the cloud. The research in searchable encryption to-date has mainly focused on range, *k*-nearest neighbours and top-*k* queries. Although the literature has demonstrated that the R*k*NN query is a fundamental and crucially important tool for data processing in many application domains, and although the algorithms for processing other types of database queries such as the range and nearest neighbour queries are inefficient with regard to answering the R*k*NN query, this work, to the author's knowledge, is the first to put forward schemes for the secure processing of this query on untrustworthy cloud servers.

The paper introduces several encryption schemes with proved realistic security guaranties and efficient performance, which has been put to the test through an experimental evaluation using a real-life and a synthetic dataset. Two of the proposed schemes mobilize a new specialized indexing method, named S^3-tree, in order to provide in practice an efficient query processing cost in massive quantities of encrypted data.

Plans for the future include endeavouring to further optimize the proposed encryption schemes towards improving their speed, without undermining their security. The flexibility of the new schemes with handling updates in dynamic datasets will also be explored. And last, the construction of models for supporting other well-known queries for multi-dimensional data will also be examined, including the *bichromatic RkNN query* [30], closely related to our problem, in which two types of datasets of points are recognised, the "users" and the "facilities", and for a specific facility q the query looks for the set of users that have q between their k nearest neighbours facilities.

REFERENCES

[1] Takabi, H., Joshi, B., and Ahn, G.J.: Security and privacy challenges in cloud computing environments. *IEEE Security and Privacy*, 8(6), (2010) 24-31.

[2] Hacıgümüs, H., Iyer, B., Li, C., and Mehrotra, S.: Executing SQL over Encrypted Data in the Database-As-a-Service Model. In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD*). (2002) 216-227.

[3] Song, D.X., Wagner, D., and Perrig, A.: Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy*, (*S&P*), (2000) 44-55.

[4] Tzouramanis, T.: Secure Range Query Processing over Untrustworthy Cloud Services. In *Proceedings of the* 21*st ACM International Database Engineering and Applications Symposium* (*IDEAS*), (2017) 108-117.

[5] Shi, E., Bethencourt, J., Chan, H.T.-H., Song, D.X., and Perrig, A.: Multi-dimensional Range Query over Encrypted Data. In *Proceedings of the IEEE Symposium Security and Privacy*. (2007) 350-364.

[6] Kim, H.I., Kim, H.J., and Chang, J.W.: A secure kNN query processing algorithm using homomorphic encryption on outsourced database. *Data and Knowledge Engineering*, in press, (2017).

[7] Xue, W., Li, H., Peng, Y., Cui, J., and Shi, Y.: Secure k Nearest Neighbours Query for High-dimensional Vectors in Outsourced Environments. *IEEE Transactions on Big Data*, in press, (2017).

[8] Wong, W.K., Cheung, D.W.L., Kao, B., and Mamoulis, N.: Secure kNN Computation on Encrypted Databases. In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD*). (2009) 139-152.

[9] Zerr, S., Olmedilla, D., Nejdl, W., and Siberski, W.: Zerber+r: Top-k Retrieval from a Confidential Index. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, (2009) 439-449.

[10] Yilmaz, E., Ferhatosmanoglu, H., Ayday, E., and Aksoy, R.C.: Privacy-Preserving Aggregate Queries for Optimal Location Selection. *IEEE Transactions on Dependable and Secure Computing*, to appear, (2017).

[11] Hidayat, A., Yang, S., Cheema, M.A., and Taniar, D.: Reverse Approximate Nearest Neighbour Queries. *IEEE Transactions on Knowledge and Data Engineering*, 30(2), (2018) 339-352.

[12] Wang, S., Bao, Z., Culpepper, S., Sellis, T., and Cong, G.: Reverse k Nearest Neighbour Search over Trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 30(4), (2017) 757-771.

[13] Zhao, P., Fang, H., Sheng, V.S., Li, Z., Xu, J., Wu, J., and Cui, Z.: Monochromatic and bichromatic ranked reverse boolean spatial keyword nearest neighbours search. *World Wide Web*, 20(1), (2017) 39-59.

[14] Gao, Y., Miao, X., Chen, G., Zheng, B., Cai, D., and Cui, H.: On efficiently finding reverse *k*-nearest neighbours over uncertain graphs. *The VLDB Journal*, 26(4), (2017) 467-492.

[15] Yang, S., Cheema, M.A., Lin, X., and Wang, W.: Reverse k nearest neighbours query processing: experiments and analysis. *Proceedings of the VLDB Endowment*, 8(5), (2015) 605-616.

[16] Yiu, M., Papadias, D., Mamoulis, N., and Tao, Y.: Reverse nearest neighbours in large graphs. *IEEE Trans. Knowledge and Data Engineering*, 18(4), (2006) 540–553.

[17] Tao, Y., Papadias, D., and Lian, X.: Reverse kNN search in arbitrary dimensionality. In *Proceedings of the* 30th *International Conference on Very Large Data Bases* (*VLDB*), (2004) 744-755.

[18] Korn, F., Muthukrishnan, S., and Srivastava, D.: Reverse nearest neighbour aggregates over data streams. In *Proceedings of the* 28th *International Conference on Very Large Data Bases* (*VLDB*), (2002) 814–825.

[19] Yang, C., and Lin, K.: An index structure for efficient reverse nearest neighbour queries. In *Proceedings of the 17th International Conference on Data Engineering* (*ICDE*), (2001) 482–495.

[20] Stanoi, I., Agrawal, D., and Abbadi, A.: Reverse nearest neighbour queries for dynamic databases. In *Proceedings of the SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, (2000) 44–53.

[21] Korn, F., and Muthukrishnan, S.: Influence sets based on reverse nearest neighbour queries. In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD*), (2000) 201–212.

[22] Nanopoulos, A., Theodoridis, Y., and Manolopoulos, Y.: C2P: clustering based on closest pairs. In *Proceedings of the 27th International Conference on Very Large Data Bases* (*VLDB*), (2001) 331-340.

[23] Guttman, A.: R-trees: a Dynamic Index Structure for Spatial Searching, In *Proceedings of the ACM International Conference on Management of Data* (*SIGMOD*). (1984) 47-57.

[24] Bothe, S., Karras, P., and Vlachou, A.: eskyline: Processing skyline queries over encrypted data. *Proceedings of the VLDB Endowment*, 6(12), (2013) 1338-1341.

[25] Lin, X., Zhou, L., Chen, P., and Gu, J.: Privacy preserving reverse nearest-neighbour queries processing on road network. In *Proceedings of the International Conference on Web-Age Information Management*, (2012) 19-28.

[26] Y. Du,: Privacy-Aware RNN Query Processing on Location-Based Services. In *Proceedings of the 8th International Conference on Mobile Data Management*, (2007) 253–257.

[27] Wang, L., Meng, X., Hu, H., and Xu, J.: Bichromatic reverse nearest neighbour query without information leakage. In *Proceedings of the International Conference on Database Systems for Advanced Applications* (2015) 609-624.

[28] White, D.A., and Jain, R.: Similarity indexing with the SS-tree. In *Proceedings of the 12th IEEE International Conference on Data Engineering* (*ICDE*), (1996) 516-523.

[29] University of California Irvine - Machine Learning Repository: The Forest Covertype Dataset, available at: http://archive.ics.uci.edu/ml/datasets/Covertype, valid as of June 2018.

[30] Kang, J.M., Mokbel, M.F., Shekhar, S., Xia, T., and Zhang, D.: Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. *Proceedings of the 23th International Conference on Data Engineering* (*ICDE*), (2007) 806-815.