Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 8109730, 15 pages
https://doi.org/10.1155/2017/8109730

Research Article

Secure *kNN* Computation and Integrity Assurance of Data Outsourcing in the Cloud

Jun Hong,[1,2] Tao Wen,[1] Quan Guo,[3] and Zhengwang Ye[1]

[1]*School of Computer Science and Engineering, Northeastern University, Shenyang, Liaoning 110819, China*
[2]*School of Software, North University of China, Taiyuan, Shanxi 030051, China*
[3]*Department of Computer Science and Technology, Dalian Neusoft University, Dalian, Liaoning 116023, China*

Correspondence should be addressed to Jun Hong; hongjun@nuc.edu.cn

Received 20 May 2017; Revised 1 October 2017; Accepted 5 November 2017; Published 13 December 2017

Academic Editor: M. L. R. Varela

As cloud computing has been popularized massively and rapidly, individuals and enterprises prefer outsourcing their databases to the cloud service provider (CSP) to save the expenditure for managing and maintaining the data. The outsourced databases are hosted, and query services are offered to clients by the CSP, whereas the CSP is not fully trusted. Consequently, the security shall be violated by multiple factors. Data privacy and query integrity are perceived as two major factors obstructing enterprises from outsourcing their databases. A novel scheme is proposed in this paper to effectuate k-nearest neighbors *(kNN)* query and *kNN* query authentication on an encrypted outsourced spatial database. An asymmetric scalar-product-preserving encryption scheme is elucidated, in which data points and query points are encrypted with diverse encryption keys, and the CSP can determine the distance relation between encrypted data points and query points. Furthermore, the similarity search tree is extended to build a novel verifiable SS-tree that supports efficient *kNN* query and *kNN* query verification. It is indicated from the security analysis and experiment results that our scheme not only maintains the confidentiality of outsourced confidential data and query points but also has a lower *kNN* query processing and verification overhead than the MR-tree.

1. Introduction

As the spatial data resources have been developed by leaps and bounds, to be well geared into such transition, the enterprises are required to proliferate the resources of both the hardware and software resources and to recruit professionals to manage and maintain data. Accordingly, the data maintenance has been boomed overhead. On the other hand, cloud computing has become progressively popularized in recent years. This arises from their capability to offer scores of benefits, such as quick deployment, on-demand service, high scalability and cost reduction [1–4]. A growing number of companies are currently being motivated to outsource their daily business, even their core business, to the cloud service provider to eliminate the investment in hardware and software and to reduce the costs to maintain data. Moreover, using the advantages of cloud computing, the end users can use on-line software applications and access the service at any time and any place [5]. Outsourcing spatial database is progressively reflecting the trend of reality. Spatial data has scores of practical applications, such as environmental monitor, location-based services, flow control, etc. In the data outsourcing model, the cloud service provider (CSP) hosts the outsourced databases and provides query services for the clients, and the data owner loses the management and control of the outsourced data. Consequently, the confidentiality and security of data shall be violated. Data privacy and security problems count as the major factors obstructing the data owners from outsourcing their databases to the CSP [6, 7].

Data encryption is perceived as the most frequently adopted approach to maintain data confidentiality. Merely the authorized parties can conduct decryption. It is noteworthy that the destination of outsourcing data is to draw on the strong computing power and high bandwidth of the cloud service provider to offer the rapid and efficient services to users. Yet, the traditional encryption approaches, such as DES and RSA, are primarily designed to encrypt the confidential data, the encrypted data cannot support efficient query and

analysis as well as the original data. Many effective schemes [8–11] have been proposed to support how to execute queries on encrypted data.

With the exception of data privacy, query integrity, also known as query authentication, is deemed as another critical problem to be solved in the domain of data outsourcing. Since the CSP is not fully trusted, it can return incorrect or incomplete query results. Extra authentication information shall be offered to the client to ensure the correctness and completeness of query results without having to trust the CSP. Correctness bespeaks that the records in the results really exist in the owner's database and are not modified by any user. Completeness bespeaks that all the records that satisfy the query condition are included in the results. Query integrity, in particular, is crucial in the case of the results laying the foundation for critical decisions.

The k-nearest neighbors *(kNN)* query is deemed as a crucial data analysis operation which can be used as an independent query or as a core module of data mining and has been applied in many practical applications, such as geospatial technology, location-based services, and pattern recognition. Recent studies [12–16] have proposed various techniques to support either *kNN* queries on encrypted data or *kNN* query authentication. However, both privacy protection and query authentication should be provided in an insecure cloud computing environment. Thus, we focus on the *kNN* query processing and *kNN* query authentication on an encrypted spatial dataset. In this paper, we introduce an asymmetric scalar-product-preserving encryption to encrypt confidential data points and query points, and then we propose an authenticated spatial index structure based on the SS-tree [17], called verifiable SS-tree (VSS-tree), for secure *kNN* query processing and *kNN* query authentication. Our main contributions are illuminated as follows:

(1) We introduce an asymmetric scalar-product-preserving encryption through which the data owner encrypts confidential data and query points with diverse encryption keys. The cloud server can perform a *kNN* query on encrypted outsourced spatial database.

(2) We extend SS-tree [17] and propose a novel verifiable SS-tree (VSS-tree) for the *kNN* query processing and *kNN* query authentication.

(3) We perform a detailed security analysis and performance evaluation of our scheme.

The rest of this paper is organized as follows. The relevant work is reviewed in Section 2. The system model is proposed in Section 3. Section 4 specifies the encryption scheme. Section 5 elaborates on the VSS-tree. In Section 6, we perform security analysis of our scheme. In Section 7, the performance and experimental results are presented. Eventually, we conclude this paper in Section 8.

2. Related Work

The encryption approach, called "bucket-based," is proposed in [8, 9]. The domain of private data is subdivided into multiple disjoint ranges and each range is identified by a unique identifier. The cloud server performs a range query in the light of the identifiers and returns a super set of real result set. The client has to do extra processing to get the real results. Agrawal et al. [10] proposed an order-preserving encryption to support one-dimensional range query on encrypted data. The input data distribution is accordingly transformed into a user-specified target distribution. The encrypted data is kept in the same order as the original data, which simplifies the course to effectuate encrypted range query. Nevertheless, this scheme fails to resist known-plaintext attack [18]. Oliveira and Zaiane [11] proposed a distance-preserving transformation (DPT) approach. DPT transforms an original data point x into a new point $Nx + t$, where N is a $d \times d$ matrix and t is a d-dimensional vector. DPT ensures that the Euclidean distance between any two encrypted data points is equal to that between the corresponding original data points; that is, $d(x, y) = d(E(x), E(y))$. However, DPT cannot resist level 2 and level 3 attacks [19]. Man et al. [20] proposed a data transformation approach to maintain data confidentiality. Using the transforming function, the data owner and user transform their original spatial data and query ranges into encrypted ones. The cloud server performs range queries on encrypted data. Chen et al. [21] proposed a random space encryption approach to support range query on encrypted data. The outsourced data and queries are encrypted on the trust agent. The cloud server indexes the encrypted data and executes queries on it. And yet, the trust agent may become the single point of failure and network bottleneck of the system. Kalnis et al. [22] adopted k anonymity to the outsourced data. The cloud server cannot distinguish a record from at least $k - 1$ records. Obviously, the cloud server fails to perform exact query in line with this scheme. Similarly, Chow et al. [23] adopted location anonymity to hide the real location of query points. Asymmetric scalar-product-preserving encryption (ASPE) is proposed in [12] for secure *kNN* query on encrypted data. The outsourced data and query points are encrypted with diverse encryption keys, and the cloud server can execute a *kNN* query on encrypted data. However, ASPE assumes that the clients are fully trusted, which is unrealistic in real applications. The client can easily obtain the encryption key from his legal inputs and outputs. Optimized ASPE is proposed in [13, 14] in which the clients are not trusted and only the data owner knows the encryption key. Data points and query points are extended to $(2d + 2)$-dimensional points in [13], which requires more than double computation overhead than that of computing original data. Paillier homomorphic encryption is used in [14] to keep the query points confidential to the data owner. Thus, the client has to provide more computing resources to encrypt and decrypt the query points.

Digital signature chain mechanism is adopted in [24–27] for query authentication. Each record is signed with its immediate predecessor or successor record (attribute). The records and their corresponding signatures are stored together on CSP. When answering a query, the CSP returns the matched records along with their corresponding signatures to the client, and thereupon the client verifies the correctness and completeness of query results according to

the corresponding signatures. However, the computational complexity of digital signature is too high. Even though using signature aggregation [28], the client has to provide more computing power for signature verification and modular multiplications.

Being different from the signature chain, Merkle [29] first proposed Merkle Hash Tree (MHT), a memory-based binary tree with authentication information, for one-dimensional equality query. The digest rather than signature, computed by a one-way and collision-resistant hash function, is associated with each node. Only one signature is computed on the top of MHT. Devanbu et al. [30] extended MHT to support one-dimensional range query. Pang and Tan [31] extended MHT and proposed a verifiable B-tree (VB-tree). Data owner has to sign each record and node in the VB-tree, which results in high signature computation overhead. The VB-tree only guarantees the correctness of the query results. Li et al. [32] proposed Merkle B-Tree (MBT) to support disk-based query authentication. To response a range query, the CSP performs two depth-first traversals to find the leftmost and rightmost records of query results and build the verification object (VO). The VO includes the following information: the digests of entries contained in each visited internal node that do not overlap with the range; the query results along with the digests of the residual entries in the corresponding leaves; the leftmost and rightmost records of the query results for completeness verification.

In multidimensional database outsourcing, Cheng et al. [33] proposed verifiable KD-tree (VKD-tree) and verifiable R-tree (VR-tree), applying signature chain to the KD-tree and the R-tree to ensure the integrity of query results, respectively. Yang et al. [34] proposed Merkle R-tree (MR-tree) and Merkle R*-tree (MR*-tree) for query verification of spatial data. MR*-tree combines the concepts from MBT and R*-tree. The authentication information is combined with the R*-tree. Range query is performed by a depth-first traversal of MR*-tree. The VO comprises all the data entries in the visited leaf nodes and the MBRs along with the corresponding digests of the sibling nodes pruned in the visited internal nodes. Yiu et al. [15] presented a framework for authenticating moving kNN queries using the safe region approach.

In other query authentication approaches, Xie et al. [35] proposed a probabilistic query integrity authentication scheme. The data owner inserts some fabricated records into the database and outsources them together to the CSP. The CSP cannot distinguish the fabricated records from the real ones. In response to a query, the CSP returns the query results (including the real and fabricated records) to the client. The client verifies the correctness and completeness of query results by verifying whether all the qualified fabricated records are included in the results.

Both privacy protection and query authentication are realized in [36–38]. Wang et al. [36] applied duplicated encryption for query verification. Part of the original data are encrypted with two diverse encryption keys. The user transforms a query into two different queries according to the encryption schemes and probabilistically verifies the query integrity by checking the two query result sets. Shamir secret sharing is used in [37] for privacy protection and multiparty computation. k service providers collaboratively compute the aggregation results without gaining knowledge of intermediate results. The integrity of aggregate results is guaranteed by MHT. Pedersen commitment protocol and MHT are used in [38] for aggregation computation and query verification.

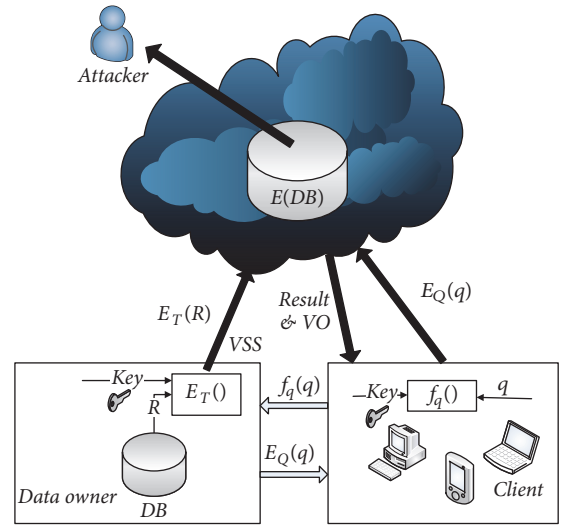


Figure 1: System model.

3. System Framework and Assumption

3.1. System Model. Figure 1 illustrates our system framework. It consists of three parts: data owner *(DO)*, client, and cloud service provider *(CSP)*. The data owner possesses two encryption keys. One is obtained from a trusted key distribution center, encompassing a private and a public key for signature. The other is generated by *DO* himself for encrypting sensitive data and query points. *DO* outsources the encrypted database D' to the *CSP*. Whenever updates occur, the corresponding encrypted data and the new signature are forwarded to *CSP*. The *CSP* hosts the outsourced database D' and provides query services for the client. To process spatial query efficiently, *CSP* maintains an authenticated spatial index structure. For each incoming query, it initiates the search algorithm to find a *kNN* query results and builds the corresponding *VO* for query authentication. To maintain the confidentiality of query points, a client transmits the processed query points to *DO* for encryption. After receiving an encrypted query point, the client transmits it to *CSP* for a *kNN* query. Once receiving *VO*, the client extracts the query results and performs query authentication. In general, the clients are located at the edge of networks and possess low network bandwidth and computing power. The clients only trust the signature information that the *DO* published.

3.2. System Assumption. We assume that the privacy data are numeric and denoted by real numbers, like the position coordinates of navigation, minimum, maximum and amount, etc. Each multidimensional piece of data is denoted as a

column vector. The *CSP* is semitrusted; thus, it can directly access the outsourced database D', fabricate or tamper with the data, and return a subset of real result set to save the computation power for providing paid services for more users. Simultaneously, *CSP* performs our scheme honestly. To maintain the confidentiality of query points, the client first transmits a processed query point to DO for encryption and thereupon extracts the encrypted query point and transmits it to *CSP* for a *kNN* query processing. The client verifies a *kNN* query results through *VO* and the public key *DO* published. Furthermore, the client is semitrusted, which may collude with *CSP* or other clients to recover the original data. Therefore, the encryption key owned by *DO* should not be revealed to the client and the CSP.

In summary, the attacks can be divided into three levels based on the knowledge the attackers can learn.

Level 1. The attacker only observes encrypted database and encrypted query points. This is known as ciphertext-only attack proposed in [18].

Level 2. With the exception of encrypted data, the attacker also knows part of the original plain data and some encryption information, such as the maximum, minimum, and data distribution of encrypted data. However, the attacker does not know the corresponding encrypted values of those plain data. This corresponds to known-sample attack [39].

Level 3. In addition to the knowledge obtained in level 2, the attacker observes a set of plain data and knows the corresponding ciphertext, and this is known as known-plaintext attack in cryptography [18].

It turns out to be evident that the knowledge of lower level that the attacker learns is a subset of what a higher-level attacker learns. If an encryption scheme can resist higher-level attacks, it can also resist lower-level attacks. Since we usually capture known-sample attack in practical applications, we design our encryption scheme against known-sample attacks.

Based on these assumptions, we should preserve the confidentiality of outsourced sensitive data and query points and provide query integrity authentication for *kNN* queries. The details are as follows:

(1) Data privacy: the confidential data should not be revealed to anyone else. Only encrypted data is outsourced to the CSP.

(2) Query privacy: query privacy bespeaks that a client's query points should be kept private to himself. Neither *DO* nor *CSP* can obtain the plain query points.

(3) Key privacy: the existing research usually shares the key with the clients. The *CSP* can easily obtain the key from the colluded or compromised client to recover the original data. Therefore, these schemes have to assume the clients are fully trusted. In our system assumption, each part of our system is semitrusted, the encryption key owned by *DO* should not be disclosed to anyone else.

Table 1: Symbol list.

Symbols	Description
p_i	A multidimensional point in the database
p_i'	The encrypted query point of p_i
q	A query point
q'	The encrypted query point of q
d	The dimensionality of spatial data
D	The private database of DO
M	A $d \times d$ matrix
M^{-1}, M^T	Inverse and transpose of matrix M
$d(p_i, q)$	The distance between p_i and q
$d(p_i', q')$	The distance between p_i' and q'
p^T, p'^T	Transposition of p, p'
$\|p\|$	Euclidean norm of point p
$E(p, Key)$	Encryption function: *Key* is the encryption key
$p_i \cdot p_j$	Scalar product of p_i *and* p_j

(4) Query authentication: based on the SS-tree, we propose a novel authenticated spatial index structure for *kNN* queries and *kNN* query authentication.

The main symbols used are listed in Table 1.

4. EASPE

4.1. Preliminary of ASPE. The basic idea of ASPE [12] is the observation that the distance between database points is not necessary for a *kNN* query. According to (1), ASPE can determine the distance relation between any two data points p_i, p_j and query point q.

$$\begin{aligned} & d(p_i, q) \le d(p_j, q) \\ & \sqrt{\|p_i\|^2 - 2p_i \cdot q + \|q\|^2} \le \sqrt{\|p_j\|^2 - 2p_j \cdot q + \|q\|^2} \\ & \Updownarrow \\ & \|p_i\|^2 - \|p_j\|^2 + 2(p_j - p_i) \cdot q \le 0, \end{aligned} \tag{1}$$

where $\|p_i\|^2$ is the scalar product of point p_i with itself, which can be computed in advance and stored with the corresponding data together for *kNN* queries. Then, ASPE does not need to preserve $\|p\|^2$. For any two encrypted points p_i', p_j', $p_i' \cdot p_j' = E(p_i, key) \cdot E(p_j, key)$. The distance between them can be computed by

$$d(p_i', p_j') = \sqrt{p_i' \cdot p_i' - 2(p_i' \cdot p_j') + p_j' \cdot p_j'}. \tag{2}$$

It is easy to see from (1) and (2) that ASPE does not keep the scalar product $p_i \cdot p_j$ to ensure that the CSP cannot compute the distance between any database points through (2). Moreover, the CSP is able to determine which data point is nearer to the query point q through (1).

Definition 1 (asymmetric scalar-product-preserving encryption (ASPE)). An encryption function E is an ASPE if and only if it satisfies the following two conditions:

(1) For any point p_i and any query point q, $p_i \cdot q = E(p_i, K) \cdot E(q, K)$.

(2) For any p_i and p_j in D, $p_i \cdot p_j \neq E(p_i, K) \cdot E(p_j, K)$.

As can be seen from Definition 1, data point and query point must be encrypted with diverse encryption keys to ensure that the encrypted value of any query point q is different from that of any data point p in D, even if $p = q$.

When encrypting a data point, ASPE randomly generates a $(d+1) \times (d+1)$ invertible matrix M_k as the encryption key and extends every data point p to a new $(d+1)$-dimensional point $\widehat{p} = (p^T, -0.5\|p\|^2)^T$ which is encrypted into $p' = M_k^T \cdot \widehat{p}$. When encrypting a query point q, the client randomly selects a positive random number r and extends the query point q to a new $(d+1)$-dimensional point $\widehat{q} = r(q^T, 1)^T$, and then he encrypts $\widehat{q}$ into $q' = M_k^{-1} \cdot \widehat{q}$, where M_k^{-1} is the encryption key of query points. To determine whether an encrypted data point p_i' is nearer to a query point q' than p_j' is, the *kNN* search algorithm checks whether $(p_i' - p_j') \cdot q' > 0$:

$$\begin{aligned}
\left(p_i' - p_j'\right) \cdot q' &= \left(p_i' - p_j'\right)^T q' = \left(M_k^T \widehat{p}_i - M_k^T \widehat{p}_j\right)^T \\
&\cdot M_k^{-1} \widehat{q} = \left(\widehat{p}_i - \widehat{p}_j\right)^T \widehat{q} \\
&= \left(\left(p_i^T, -0.5\left\|p_i\right\|^2\right)^T - \left(p_j^T, -0.5\left\|p_j\right\|^2\right)^T\right)^T \\
&\cdot r\left(q^T, 1\right)^T \\
&= \left(\left(p_i - p_j\right)^T, \left(-0.5\left\|p_i\right\|^2 + 0.5\left\|p_j\right\|^2\right)\right)(rq, r) \\
&= \left(p_i - p_j\right)^T (rq) + \left(0.5\left\|p_j\right\|^2 - 0.5\left\|p_i\right\|^2\right) r \\
&= 0.5r\left(\left\|p_j\right\|^2 - \left\|p_i\right\|^2 + 2\left(p_i - p_j\right)^T q\right) \\
&= 0.5r\left(d\left(p_j, q\right) - d\left(p_i, q\right)\right).
\end{aligned} \tag{3}$$

Since r is a positive random number, we can determine that

$$d\left(p_j, q\right) - d\left(p_i, q\right) > 0 \Longleftrightarrow d\left(p_j, q\right) > d\left(p_i, q\right). \tag{4}$$

4.2. kNN Query on ASPE. As described in Section 4.1, the client is assumed to be fully trusted by the data owner, and the encryption key and configuration information are shared with the client. However, in a more practical scenario, a client may be compromised or colludes with the CSP so that the CSP can easily obtain the key and the private configuration to decrypt the encrypted data. One plausible approach is that the DO keeps the encryption key privately and performs a secure two-party computation protocol [29, 30] with the clients. DO encrypts a processed query point $\widehat{q}$ and only transmits the encrypted query point q' to the client without disclosing the encryption key M. However, the combination of ASPE and secure two-party computation remains unable to maintain the key confidentially [14]. The encryption key shall be leaked to the others from legal outputs. The client can adequately choose enough query points $Q = (q_1, q_2, \ldots, q_{d+1})$ and obtain the corresponding encrypted query points $Q' = (q'_1, q'_2, \ldots, q'_{d+1})$, and then the client obtains $Q' = M^{-1}Q$. Obviously, if Q is an invertible matrix, the client can obtain $M^{-1} = Q'Q^{-1}$, by which the client can encrypt a new query point $q' = Q'Q^{-1}(q_{\text{new}}, 1)^T$. Therefore, the encryption key and sensitive data are exposed to the attackers.

4.3. Enhanced ASPE. We propose an enhanced ASPE (EASPE) that keeps the encryption key confidential to the clients. Being different from ASPE, it is hypothesized in this paper that the three parties in our system model are not trusted by each other. Therefore, the DO must keep the encryption key confidentially and the key cannot be obtained by anyone, while the client should keep the query points secret to the DO and the CSP. Our encryption scheme is similar to the approach proposed in [14]. However, the scheme in [14] adopted Paillier homomorphic encryption to encrypt query points which burdened the client with more computation overhead. In our scheme, we apply a 1-out-of-N oblivious transfer protocol [40] for query processing. A 1-out-of-N oblivious transfer protocol [40] is a protocol such that one party, Bob, has N inputs $X_1, \ldots, X_n$ and the other party, Alice, learns one of the inputs X_i for some $1 \leq i \leq n$ of her choice, without learning anything about the other inputs and without allowing Bob to learn anything about i. A random matrix encompassing the processed inquiry point is generated by the client and is sent to the DO for encryption.

Before encrypting data points, several artificial columns are introduced to the data points and are associated with some nonce random numbers generated independently which allows the same points to be encrypted into diverse points. Likewise, the client adds the same number of artificial columns to a query point and then perturbs the query point with some random numbers generated independently. The client sends a mixed matrix Q, encompassing the extended query point and some random vectors generated randomly, to the DO for encryption. Eventually, DO perturbs Q before matrix transformation so that the encrypted query points cannot reveal the key.

The outputs of p_i, q in the data process stage are denoted by $\widehat{p}_i$, $\widehat{q}$, respectively. *DO* finishes the encryption of $\widehat{p}_i$ and $\widehat{q}$ and outputs the ciphertexts p_i' and q' in the data encryption stage. It is noteworthy that *DO* cannot directly compute $\widehat{q}$ while encrypting the query point; nobody except the client knows the original query point. To simplify the description, $\widehat{q}$ is adopted to state our scheme in the first phase. Next, the two phases are elaborated on.

Data Processing. For each data point p, *DO* first selects a positive integer c as system security parameter in advance. In point perturbation, two random vectors ω of $(d+1)$ dimensions and χ of c dimensions are generated by *DO*, taking up the encryption key and shared by all points in the

database. The permutation function f_p changes the sequence of the extended vector randomly. As the foregoing processing is effectuated, (5) is acquired.

$$\widehat{p_i} = f_p\left(\omega_j - 2p_i, \omega_{(d+1)} + \|p_i\|^2, \chi\right), \tag{5}$$

where ω_j $(1 \le j \le d)$ indicates the jth dimension of ω.

For each query point q, firstly a positive random r is selected, and a random vector δ of c dimensions is created by the client, followed by the client's extension of q to $\dot{q} = (r(q^T, 1), \delta)$ and transmitting $\dot{q}$ to the *DO*. *DO* generates a random vector R of c dimensions to perturb the last c dimensions of $\dot{q}$. Accordingly (6) is acquired.

$$\widehat{q} = f_p\left(r\left(q, 1\right), R + \delta\right). \tag{6}$$

Since the permutation function f_p does not change the scalar product between data point and query point, then, $\widehat{p_i}\widehat{q}^T = (\omega_j - 2p_i)rq^T + (\omega_{(d+1)} + \|p_i\|^2)r + \chi(R+\delta)^T$. For any two data points p_i, p_j and a query point q, we have

$$\begin{aligned}\widehat{p_i}\widehat{q}^T - \widehat{p_j}\widehat{q}^T &= \|p_i\|^2 r - 2p_i r q^T - \|p_j\|^2 r + 2p_j r q^T \\ &= r\left(\left(\|p_i\|^2 - 2p_i q^T\right) - \left(\|p_j\|^2 - 2p_j q^T\right)\right) \\ &= r\left(d\left(p_i, q\right) - d\left(p_j, q\right)\right).\end{aligned} \tag{7}$$

Since random number r is positive, it does not affect the comparative result of (7); that is,

$$d\left(p_i, q\right) - d\left(p_j, q\right) > 0 \Longleftrightarrow d\left(p_i, q\right) > d\left(p_j, q\right). \tag{8}$$

Encryption Phase. *DO* generates an invertible matrix M as the encryption key to encrypt $\widehat{p_i}$, such that $p_i' = \widehat{p_i}M$. For each query point q, *DO* randomly generates a positive random number α to compute $q' = \alpha M^{-1}\widehat{q}^T$.

The details of encryption process are as follows.

KeyGen( ). Let $l = (d + 1 + c)$. *DO* generates a $(d + 1)$-dimensional vector ω, a c-dimensional vector χ, a $l \times l$ invertible matrix M, and a permutation function $f_p(\cdot)$ of l numbers. DO sets the quadruple $\{f_p, \omega, \chi, M\}$ as the encryption key and keeps it privately.

EncDB(D, Key). Once obtaining $\widehat{p_i} = f_p(\omega_j - 2p_i, \omega_{(d+1)} + \|p_i\|^2, \chi)^T$, *DO* computes the encrypted point $p_i' = \widehat{p_i}M$. The encrypted database is denoted as $D' = \{p_i' \mid p_i \in D\}$.

EncQuery (q, Key)

(1) After obtaining the extended $(d + 1 + c)$-dimensional point $\dot{q} = (r(q, 1), \delta)$, the client uses the 1-out-of-N oblivious transfer protocol [40] to generate a $l \times l$ matrix M_q encompassing $\dot{q}$ and other $(l - 1)$ vectors. The first d columns of the other $(l-1)$ column vectors in M_q are generated randomly and are extended to $(d + 1 + c)$-dimensional column vectors the same as the query point. The position i of column vector $\dot{q}$ is randomly selected from 1 to l and is only known to the client himself. The client transmits M_q to *DO* for encryption.

(2) For each query point q, *DO* randomly generates a random vector R of c dimensions to confuse the last c dimensions of $\dot{q}$, and then he applies the permutation function f_p to obtain $f_p(M_q)$. *DO* randomly selects a random positive number α and computes matrix $Q = \alpha M^{-1} f_p(M_q)$.

(3) After obtaining Q, the client extracts the encrypted query point q', that is, the ith column vector of matrix Q.

kNNQuery(q). To determine whether $d(p_i', q') < d(p_j', q')$, the *kNN* algorithm checks whether $(p_i' - p_j')\cdot q' < 0$ according to

$$\begin{aligned}p_i'q' - p_j'q' &= \left(p_i' - p_j'\right)q' = \left(\widehat{p_i} - \widehat{p_j}\right)M\alpha M^{-1}\widehat{q}^T \\ &= \alpha\left(\widehat{p_i} - \widehat{p_j}\right)\widehat{q}^T \\ &= \alpha r\left(\left(\|p_i\|^2 - 2p_i q^T\right) - \left(\|p_j\|^2 - 2p_j q^T\right)\right) \\ &= \alpha r\left(d\left(p_i, q\right) - d\left(p_j, q\right)\right).\end{aligned} \tag{9}$$

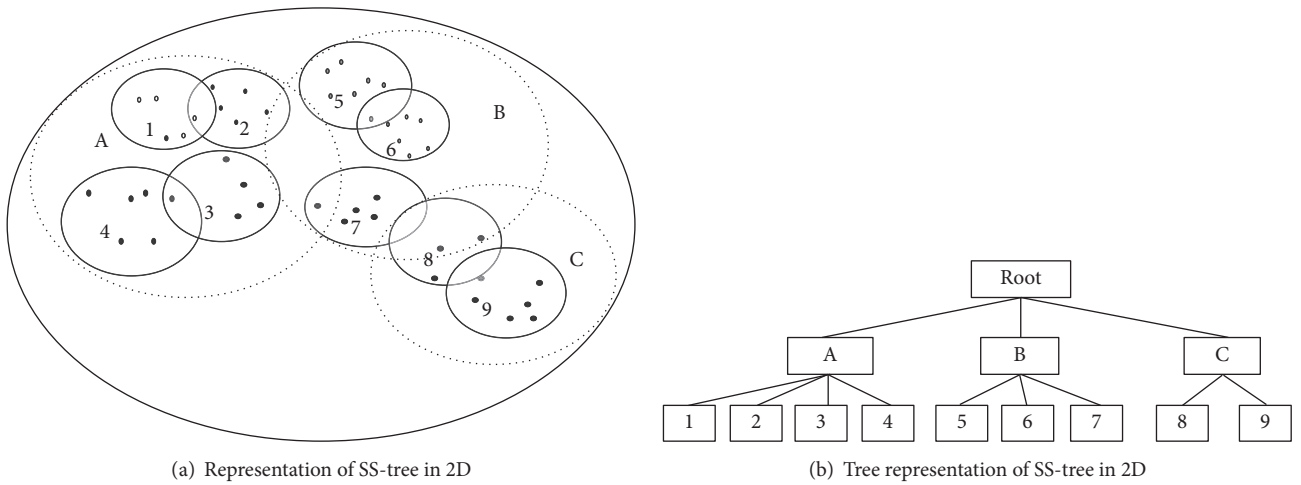
5. VSS-Tree

The simplest approach to find the results of a *kNN* query is to scan the entire database space. Yet, the query time and complexity are proportional to the data size and disk accesses, which usually cannot meet the needs of users. To improve the efficiency of spatial query, researchers build diverse spatial index structures, like R-tree [41], SS-tree [17], etc. In this section, we extend the SS-tree [17] with authentication information and build a verifiable SS-tree (VSS-tree) for *kNN* query processing and *kNN* query authentication.

5.1. VSS-Tree. Being different from the R-tree and the R*-tree, the similarity search tree (SS-tree) [17] applies bounding sphere rather than bounding rectangle for region shape. The SS-tree divides multidimensional points into isotropic neighbors. Due to the use of bounding sphere, the overlap area between regions is reduced, thereby improving *kNN* query efficiency. The structure of SS-tree is shown in Figure 2. A verifiable SS-tree (VSS-tree) is built by extending the SS-tree with authentication information, and its structure is shown in Figure 3. The center of a bounding sphere is the centroid of the underlying points of its children. Compared with the R-tree, the SS-tree only spends nearly half storage. Because a bounding sphere can be denoted by a center and a radius, its storage cost is a multidimensional point plus an integer, while a rectangle is determined by the two points at the lower left and upper right corner, its storage is twice that of dimensions. This determines that the SS-tree has more fanout and lower height.

The structure of leaf nodes is defined as follows:

$$\begin{aligned}&Leaf: \left(E_1, \ldots, E_f\right) \quad (m \le f \le M), \\ &E_i : (p, I, H),\end{aligned} \tag{10}$$



(a) Representation of SS-tree in 2D (b) Tree representation of SS-tree in 2D

Figure 2: Structure of SS-tree.

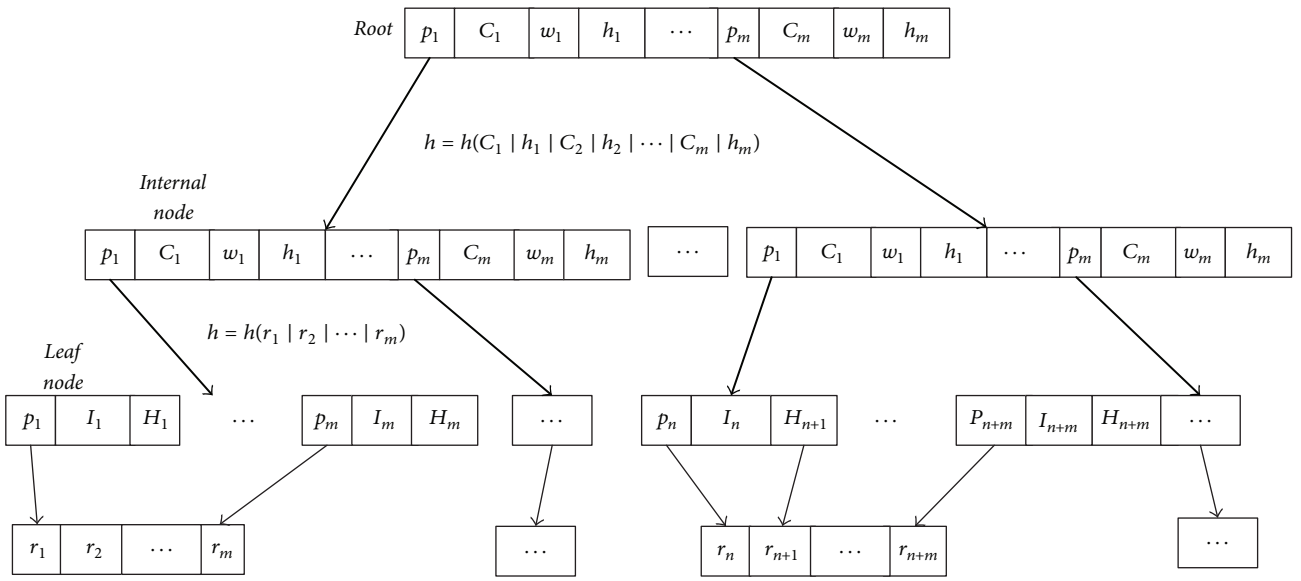


Figure 3: Structure of VSS-tree.

where m and M denote the minimum and maximum values of the entries in the leaf node, respectively. An entry of the leaf node is denoted as a triple (p, I, H), where p is a data point in database, I is the enclosing sphere of p, and H is the hash value computed on the record that p points to. An internal node of VSS-tree is elucidated as follows:

$$\begin{aligned} Node &: \left(E_1, \ldots, E_f\right) \quad (m \le f \le M), \\ E_i &: (c, p, w, h), \end{aligned} \tag{11}$$

where c indicates the minimum bounding sphere that encompasses all the regions of the ith children, consisting of a center and a radius. The pointer p points to the ith child. The variable w indicates the number of points contained in the subtree whose top is the child E_i. The hash value h summarizes all the bounding spheres and their digests of the ith child, that is, $h = h(c_1|h_1|\cdots|c_f|h_f)$. The center of a bounding sphere $x(x_1, x_2, \ldots, x_d)$ is computed according to

$$x_i = \frac{\sum_{j=1}^{n} c_j.x_i \times c_j.w}{\sum_{j=1}^{n} c_j.w} \quad (1 \le i \le d), \tag{12}$$

where j $(1 \le j \le d)$ is an index of its children, i is an index to the dimensions, $c_j.x_i$ indicates the ith dimensional coordinate of $c_j.x$, and $c_j.w$ indicates the number of its children of E_j. The radius of a bounding sphere is computed according to

$$r = \max_{1 \le j \le f} \left(\left\| c.x - c_j.x \right\| + c_j.r \right), \tag{13}$$

where $c.x$ indicates the center of the current node itself, $c_j.x$ and $c_j.r$ indicate the center and radius of the jth child node, respectively, and $\|c - c_j.x\|$ indicates the distance between the centers $c.x$ and $c_j.x$.

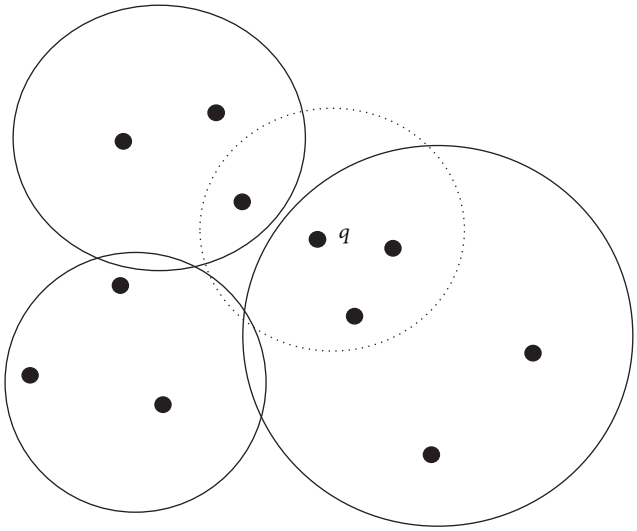


Figure 4: kNN search.

VSS-tree is built from bottom to top. All the leaf nodes are on the same level. Finally, the digest of root node is signed and published to *CSP*.

Approaching the MR*-tree, the VSS-tree also supports dynamic operations, including insertion, update, and deletion. An update can be perceived as a combination of a deletion and an insertion. The insertion adopts enforced reinsert; that is, we first add a new data object to the reinsert list and then perform insertion for the entries in the reinsert list until the list is empty. The deletion is the same as the other query authentication approaches. In our scheme, scalar product comparison rather than Euclidean distance comparison is adopted to determine where to insert a data point. When a node overflows, the split algorithm shall be revoked. Accordingly, its coordinate variance on each dimension from the centroids of its children shall be computed, and the dimension will be selected with the highest variance for splitting it.

5.2. kNN Query on VSS-Tree. Figure 4 shows a *kNN* query ($k = 3$) based on the VSS-tree. Given a query point q, *kNN* search algorithm gradually increases the search distance with the query point q as the center until the search area just encompasses three data points. We perform the *kNN* search algorithm and build the *VO* based on VSS-tree on the cloud server side. Diverse from other distance-based comparison methods, we determine the distance relationship by comparing the scalar products between the data points and the query points. The *kNN* search algorithm is shown in Algorithm 1. According to (9), we can compare the distances between any two encrypted data points and encrypted query point to determine which point is closer to a given query point. Furthermore, we introduce a sorted list *BranchList*, storing the entries of a visited internal node, to avoid unnecessary access. *KnnRes* comprises candidate *kNN* results, *KnnRes.MaxDist* denotes the maximum value in the *KnnRes*, and if *KnnRes* comprises less than k objects, *KnnRes.MaxDist* is elucidated as $+\infty$. The results shall be searched via the *kNN* search algorithm from top to bottom, and if the current visited node is an internal node, all the entries and their corresponding scalar products shall be inserted into the list *BranchList*. On that basis, the CSP shall iterate through the ordered list *BranchList* and recursively invokes the search algorithm on its visited child nodes. Once a scalar product is greater than *KnnRes.MaxDist*, the remaining entries in the *BranchList* are ignored, bespeaking that the results shall impossibly exist in the remaining entries. Thus, the minimum bounding spheres and digests of the remaining unvisited entries in the *BranchList* are inserted into *VO*. If the current visited node is a leaf node and the length of *KnnRes* is less than k, the visited entry and its scalar product will be inserted into list *KnnRes*; otherwise, whether to insert the entry into list *KnnRes* shall be determined by judging whether its scalar product is less than *KnnRes.MaxDist*. The *VO* is constructed in the search process, which consists of all the multidimensional encrypted data points in the visited leaf nodes, and the boundary spheres and their corresponding digests of the child nodes pruned of the visited internal nodes. Eventually, Algorithm 1 outputs list *KnnRes* and *VO*.

Require:
node *Node*, query point p, k
Ensure:
VO; *KnnRes*
(1) Append [to *VO*
(2) **if** *Node* is a leaf node **then**
(3) **for** each entry E in *Node* **do**
(4) VO.append(E)
(5) $dist = E.getProduct(p)$
(6) **if** $KnnRes.size() < k$ **then**
(7) $KnnRes.add(E, dist)$
(8) **else**
(9) Sort *KnnRes* in ascending order
(10) **if** $dist < KnnRes.MaxDist$ **then**
(11) $KnnRes.remove(k)$
(12) $KnnRes.add(E, dist)$
(13) **end if**
(14) **end if**
(15) **end for**
(16) **else**
(17) **for** each entry E in *Node* **do**
(18) $dist = E.getProduct(p)$
(19) $BranchList.add(E, dist)$
(20) **end for**
(21) Sort *BranchList* in ascending order
(22) **for** each entry E in *BranchList* **do**
(23) **if** $dist < KnnRes.MaxDist$ **then**
(24) $KnnSearch(E.p, q, k)$
(25) **else**
(26) $VO.append(E.C, E.h)$
(27) **end if**
(28) **end for**
(29) **end if**
(30) Append] to *VO*

Algorithm 1: *KnnSearch*.

```
Require:
    VO, q, KnnRes
Ensure:
    C, hash, result
(1)  dist = KnnRes.MaxDist
(2)  for each entry E in VO do
(3)     if E is a data object then
(4)        str = str | E
           //Enlarge the C to encompass E
(5)        C.Enlarge(E)
(6)     end if
(7)     if E.getProduct(q) ≤ KnnRes.MaxDist and
        E.id in KnnRes then
(8)        result.add(q)
(9)     end if
(10)    if E.getProduct(p) ≤ KnnRes.MaxDist
        and E.id not in KnnRes then
(11)       Alarm the client
(12)    end if
(13)    if E is a symbol [ then
(14)       (C, hash) = KnnVerify(VO, q, KnnRes)
(15)    end if
(16)    if E is a pair (C, hash) then
(17)       if C.getProduct(p) ≤ KnnRes.MaxDist
           then
(18)          Alarm the client
(19)       end if
(20)       C.Enlarge(C)
(21)       str = str|C|hash
(22)    end if
(23)    if E is a symbol ] then
(24)       Return(C, hash(str))
(25)    end if
(26) end for
```

Algorithm 2: *KnnVerify*.

Once receiving the *VO* and *KnnRes*, the client extracts the encrypted *kNN* query results from the *VO* and performs query verification. Diverse from other approaches, the client obtains the maximum *KnnRes.MaxDist* and verifies whether it is less than the other scalar products not in the list *KnnRes* to check the completeness of the *kNN* results. The verification process is as follows:

(1) The client obtains *KnnRes.MaxDist* from the list *KnnRes* and verifies that any scalar product in the *KnnRes* is less than or equal to *KnnRes.MaxDist*, while the other scalar products are greater than *KnnRes.MaxDist*.

(2) The client verifies that any scalar product between the bounding sphere in the pair *(Circle, hash)* and the query point q is greater than *KnnRes.MaxDist*.

(3) The client checks whether the reconstructed hash h_{root} agrees with s_{root}.

The *kNN* verification algorithm is shown in Algorithm 2.

The essence of the *kNN* verification algorithm is to reconstruct the VSS-tree by scanning *VO*. During the process of verification, the bounding sphere is enlarged gradually by encompassing the objects read from *VO*. Eventually, the algorithm reconstructs the bounding sphere and digest of root node, and the client validates whether the reconstructed h_{root} agrees with s_{root} for query verification.

6. Security Analysis and Integrity Verification

6.1. Security Analysis. As described in Section 3, three parties are all semitrusted. In our scheme, the privacy issues of outsourced database D', query points, and encryption key are deliberated. *CSP* can directly access the outsourced database. We need to ensure their confidentiality against *CSP*. We consider data privacy together with query privacy against *CSP* under level-2 attack.

Theorem 2. *EASPE is not distance-recoverable.*

Proof. EASPE is an enhanced ASPE, its encryption key is $\{f_p, S, \chi, M\}$, where the role of the invertible matrix M is applicable to the encryption key of ASPE. M and M^{-1} are adopted by EASPE to encrypt data points in D and query points, respectively. As ASPE proves, our EASPE is also not distance-recoverable. □

Theorem 3. *EASPE is secure against level 2 attacks.*

Proof. There are scores of types of level 2 attacks. According to the system security assumption, the following attacks are deliberated: distance-based inference attack, PCA, duplicate analysis, distribution analysis attack, and ICA-based attack. According to Theorem 2, EASPE is not distance-recoverable. Distance-based inference attack is obviously not feasible to our scheme.

Principal component analysis (PCA) has been proposed in [19] to match the correlations in the known data and the correlations in the encrypted data. Using the matched data, the attacker endeavors to reconstruct the entire original database. However, in EASPE, the values on each dimension of $E(DB)$ are a linear combination of the values on all dimensions in the original database. EASPE adds c artificial columns and generates a random vector ω to confuse the original data. Furthermore, *DO* uses permutation function f_p to change the sequence of the extended data point randomly. It turns out to be evident that EASPE does not preserve the correlations among the original dimensions in the transformed space, and thus PCA is not applicable to EASPE.

Duplicate analysis [10] is applicable to the attribute whose domain is small, such as the day of the week or the day of the month. Through the analysis of observations on encrypted data, the attacker may determine the domain of original attribute. Duplicate analysis is value-based encryption, that is, the values in each dimension are encrypted individually. However, EASPE is a tuple-based encryption, and duplicate analysis is not applicable to EASPE. Similarly, distribution analysis attack exists for estimating from Y. Observations on the encrypted database may help an attacker to determine the plain data fall into intervals $I_1, I_2, \ldots, I_n$. This attack is value-based encryption and is not applicable to EASPE.

ICA-based attack [18, 19] tries to recover the plain data X from the transformed data Y. The approach is based on the observation that the eigenvectors of Y are computed by X left-multiplied by M. Therefore, by estimating $\sum y$ and $\sum x$ and matching their eigenvectors, the attacker can produce $\widehat{M}$, an estimation of M, and then data record x_i is estimated as $\widehat{x_i} = \widehat{M^T} y_i$. This attack is on the assumption that the known samples follow the same distribution with the original data. The matrix M must be orthogonal or full rank. However, we introduce one-time random vectors χ and R for each data point and query point, respectively. Random vectors χ and R are generated independently and privately kept by *DO*, and matrix M can be generated as an invertible but nonorthogonal matrix. Hence, EASPE can impede both ICA and deriving the transformation matrix M. EASPE is therefore resilient to ICA-based Attacks.

To keep the query points confidential to *DO*, a positive number r is randomly selected and a random vector δ of c dimensions is generated to extend a query point q to a $(d + 1 + c)$-dimensional point $\dot{q} = (r(q, 1), \delta)$. And, then, 1-out-of-$N$ oblivious transfer protocol is used to generate a $l \times l$ matrix M_q including the processed query point $\dot{q}$ and other $(l - 1)$ random column vectors. The position i of column vector $\dot{q}$ is randomly selected in range from 1 to l and is only known to the client himself. *DO* cannot learn which one the client has chosen. □

Theorem 4. *The encryption key is kept confidentially against CSP and clients.*

Suppose that a client can transmit a few number of query points to *DO* for encryption, and then the encryption key is derived from the correlation between plaintext and corresponding ciphertext. If we can keep the encryption key confidential to the clients. It turns out to be evident that the key is confidential to *CSP*. Thus, we only need to prove that the encryption key is confidential to the clients.

Proof. A client transmits processed query points to *DO* and interacts with the *DO* during the query encryption stage. The encryption of query points is considered without applying permutation function in the first place. *DO* encrypts a processed query point $\widehat{q} = (r(q^T, 1), R + \delta)^T$ into $q' = \alpha M^{-1} \widehat{q}$. The ith dimension of q' is $q'_i = \alpha M^{-1}_{i*} \widehat{q}$; concretely,

$$q'_i = \alpha \left(r \sum_{j=1}^{d} M^{-1}_{ij} q_j \right) + \left(\alpha r M^{-1}_{i,d+1} + \alpha \sum_{j=d+2}^{d+1+c} M^{-1}_{ij} (\chi + R)_{j-d-1} \right). \tag{14}$$

In (14), all the values of $\{\alpha, M^{-1}_{ij}, R\}$ are kept confidential to the client. The client only knows the original query point q and its corresponding encrypted query point q'. Let $X^i_q = r \sum_{j=1}^{d} M^{-1}_{ij} q_j$, and $\varphi^i_q = (\alpha r M^{-1}_{i,d+1} + \alpha \sum_{j=d+2}^{d+1+c} M^{-1}_{ij} (\chi + R)_{j-d-1})$. The client can set up an equation $q'_i = \alpha X^i_q + \varphi^i_q$. The client can obtain enough encrypted query points by his legal input or collusion with other clients. However, the invertible matrix M is generated randomly and $\{\alpha, R\}$ are one-time random parameters selected independently for each query point. φ^i_q is entirely random to the client. Moreover, the client can learn nothing about M_{ij} $(1 \le j \le d)$ from X^i_q. Furthermore, EASPE applies permutation function f_p to the query points. The client cannot learn the correspondence of the dimensions of $\widehat{q}$ and M_{i*}. In addition, the permutation can prevent the client from setting up equation (14). Obviously, it enhances the security of our scheme. In conclusion, the encryption key is kept private against *CSP* and the clients. □

6.2. Integrity Verification. Our scheme provides correctness and completeness verification for *kNN* queries.

Theorem 5. *The correctness of kNN query results can be ensured by our scheme.*

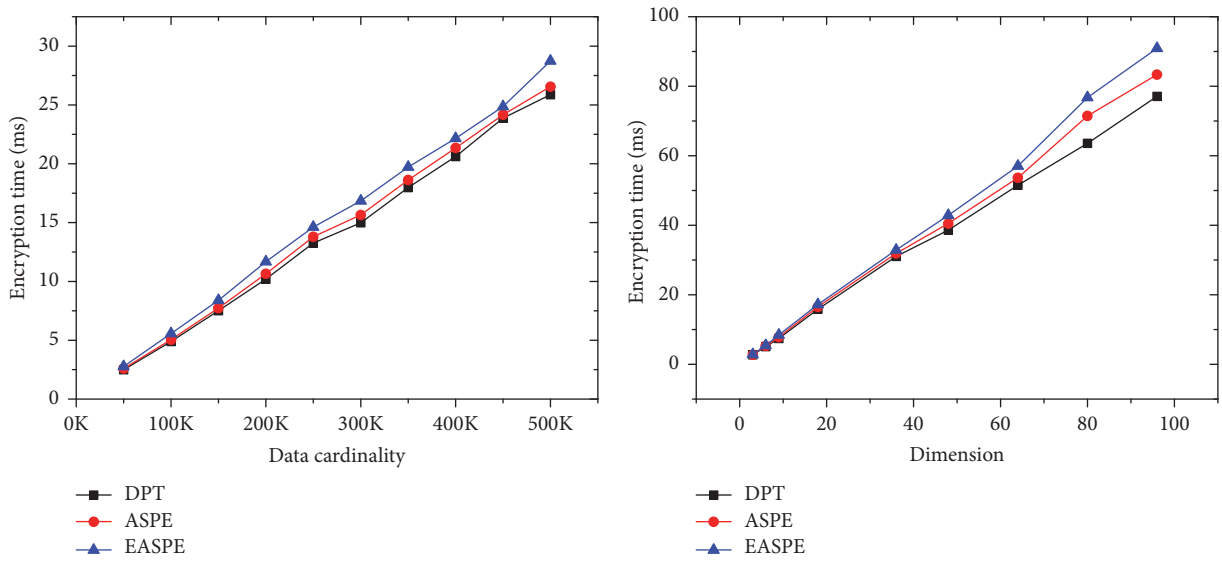
Proof. Suppose that there is one or more falsified or modified data points in the results. We note that VSS-tree is built from bottom to top. All data points in the database are involved in the construction of the root hash. As we know that the hash function is one-way and collision-resistant. The digest of any falsified or modified data must be different from the original one, and this change propagates from the leaf node to the root node which makes the reconstructed root digest h_{root} different from the original one and thus does not agree with s_{root}. Therefore, the client can detect any falsified or modified data in the results. □

Theorem 6. *The completeness of kNN query results is ensured by our scheme.*

Proof. Suppose that a data point p in a leaf node L_n is one of a *kNN* query results, but p is not involved in the results. To make the reconstructed root hash h_{root} match s_{root}, *VO* either comprises all the data entries in L_n or comprises the pair $(C, hash)$ of L_n. For the former, the client can determine that p is one of a *kNN* query results according to the verification algorithm and there exists at least one point in the results whose distance to q is farther than that of p. For the latter, the client can detect that the scalar product between L_n and q is less than *KnnRes.MaxDist*, which means that L_n comprises one or more data points that are closer to the query result q, but L_n is not visited by the search algorithm which can be detected during the verification process. □

7. Experiment Evaluation

In this section, we mainly evaluate and compare the performance between DPT and our scheme. All programs are implemented in Java. Experiments are performed on an Intel Core i7-4790 3.6 GHz computer with 8 GB RAM running Windows 7. The block size is set as 2048 KB and the default value of security parameter c as 1. The experiments are conducted on both synthetic and real datasets. The random points generated in the synthetic database are uniformly distributed in a d-dimensional space. The real dataset adopted



(a) Encryption time versus data cardinality ($d = 6$)

(b) Encryption time versus dimension ($n = 100$ K)

Figure 5: Encryption time.

Table 2: Symbol list.

	DPT	EASPE
Encryption time (ms)	2.712	2.718

is the dataset "Shuttle" from the UCI repository, which comprises 58 K points and 9 dimensions. We run each experiment 100 times and take the average to show the performance of diverse schemes. We effectuate two experiments under diverse data cardinalities and dimensions in the synthetic database. In the first experiment, data cardinality is changed from 50 K to 500 K with a fixed dimension $d = 6$. In the second experiment, the dimensions are changed from 3 to 100 with a fixed data cardinality $n = 100$ K. The performance is evaluated from the following aspects: (1) data encryption; (2) construction and storage of the VSS-tree; (3) *kNN* query; (4) query verification.

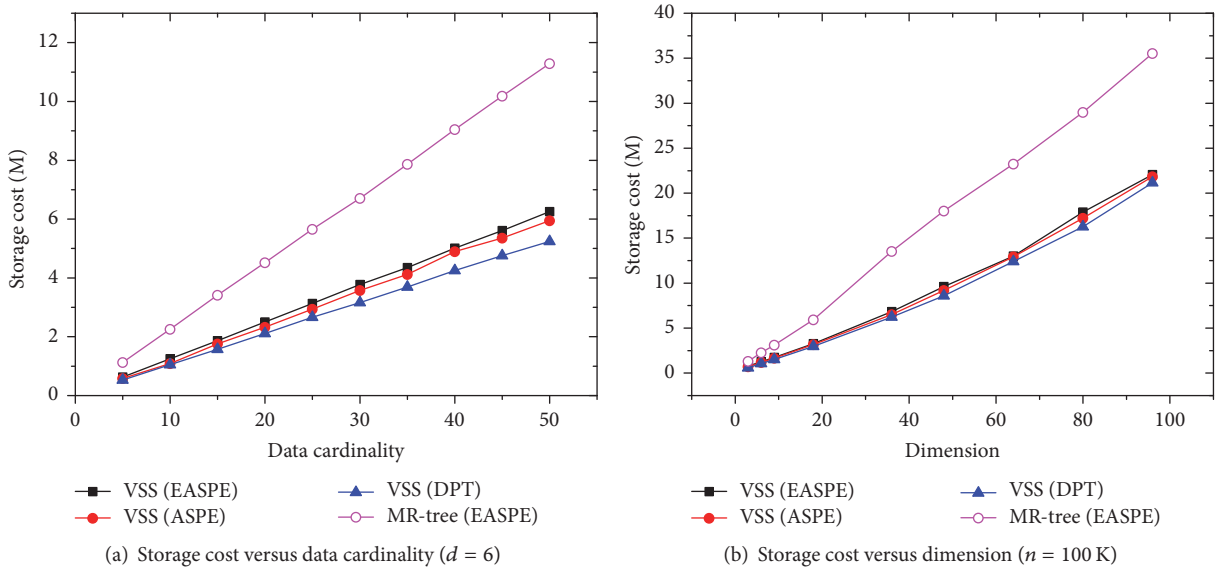
7.1. Key Generation and Data Encryption. As described in Section 4, the transition matrix used in EASPE is a $(d + 1 + c) \times (d + 1 + c)$ invertible matrix. In practical applications, the dimension d of spatial data is usually less than 100. In our experiments, we generate the encryption key only once, which takes less than 1 ms for diverse dimensions ranging from 3 to 100. Figure 5 illustrates the data encryption time on diverse data cardinalities. The encryption time includes: generating the encryption key and encrypting all the data points. The encryption time of the Shuttle dataset is shown in Table 2.

As can be seen from Figure 5, data encryption time is proportional to both data dimension and data cardinality. The encryption time of DPT is slightly shorter than that of EASPE in that EASPE performs $(d + 1 + c) \times (d + 1 + c)$ multiplications and $(d + c)$ additions, while DPT performs $(d \times d)$ multiplications and $(2d - 1)$ additions. As EASPE has c more dimensions than ASPE, the encryption time of EASPE is slightly larger than that of ASPE.

7.2. Construction and Storage Cost of VSS-Tree. The storage cost of VSS-tree is indicated from Figure 6. The storage costs under all schemes are proportional to data dimension and data cardinality. Due to the added $(c + 1)$ dimensions of EASPE, the storage cost of EASPE is larger than that of DPT and ASPE. Furthermore, the SS-tree only spends nearly half storage of that of the MR-tree as described in Section 5.1, the storage cost of the MR-tree is lager than that of the VSS-tree.

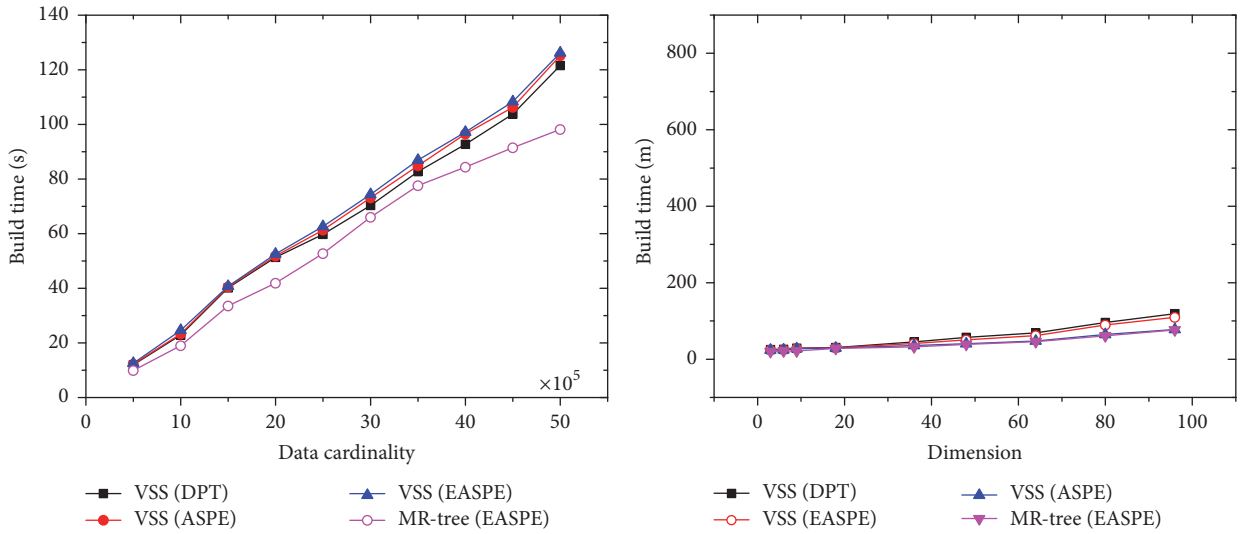
The build time of the VSS-tree is illuminated from Figure 7. The build time of the VSS-tree under both encryption schemes is proportional to both data cardinality and dimension. The build time of VSS-tree under EASPE is longer than that under DPT, this is because a d-dimensional data point is extended to a $(d + 1 + c)$-dimensional data point in EASPE which makes the computation overhead greater than that under DPT. Eventually, it should be noted that the larger the parameter c we set, the longer the time required to build the VSS-tree. The build time under the MR-tree is shorter than that under the VSS-tree, the reason is that bounding rectangle requires only comparison operations between each dimension of point, while bounding sphere needs to compute the center and radius.

The fanouts of internal node are exhibited in Figure 8 under diverse encryption schemes. Since we add $(c + 1)$ dimensions to each data point in EASPE, the fanout of VSS-tree based on it is slightly less than that based on DPT and ASPE. The fanouts under all schemes decrease with the increase of the dimensions. This is because the storage cost of a record increases as the dimension increases. Furthermore, bounding rectangle is used in MR-tree whose storage is twice



(a) Storage cost versus data cardinality ($d = 6$) (b) Storage cost versus dimension ($n = 100$ K)

Figure 6: Storage cost of VSS-tree.



(a) Build time versus data cardinality ($d = 6$) (b) Build time versus dimension ($n = 100$ K)

Figure 7: Build time of VSS-tree.

that of dimensions, the fanout of the MR-tree is less than that of the VSS-tree.

7.3. kNN Query Cost. We perform a *kNN* query on the VSS-tree and set $k = 3$. Figure 9 shows that the query processing time is proportional to both data dimension and data cardinality. The query efficiency under EASPE is higher than that under DPT. This is because the *kNN* search algorithm performs $(d + 1 + c)$ multiplications and $(d + c)$ additions to compute $p' \cdot q'$ for each visited entry under EASPE, while Euclidean distance $dist(p, q)$ is computed in DPT, the *kNN* search algorithm performs d multiplications, d subtractions and $(d - 1)$ additions for each visited entry. As described in Section 5.1, the overlap area and regions in the MR-tree are larger than those in the VSS-tree, and more nodes need to be accessed for a query. Thus, the query processing time based on the MR-tree is longer than that based on the VSS-tree.

The size of *VO* directly affects the server's response speed and network bandwidth resources. In our experiment, *VO* contains multidimensional data points of the visited leaf nodes, the bounding spheres, and corresponding digests of nodes pruned. Figure 10 illustrates that the *VO* size increases with data cardinality. Due to the use of bounding rectangle in the MR-tree, its *VO* size is larger than that under the VSS-tree.

Once receiving *VO*, the client extracts the query results from it and validates the correctness and completeness of the *kNN* query results. The verification cost includes the

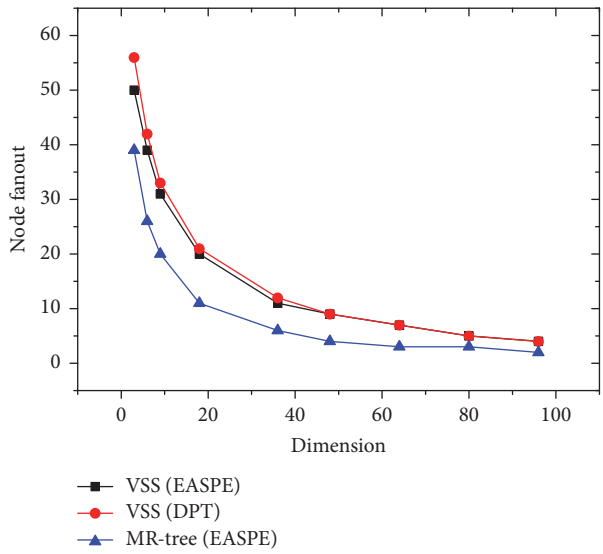
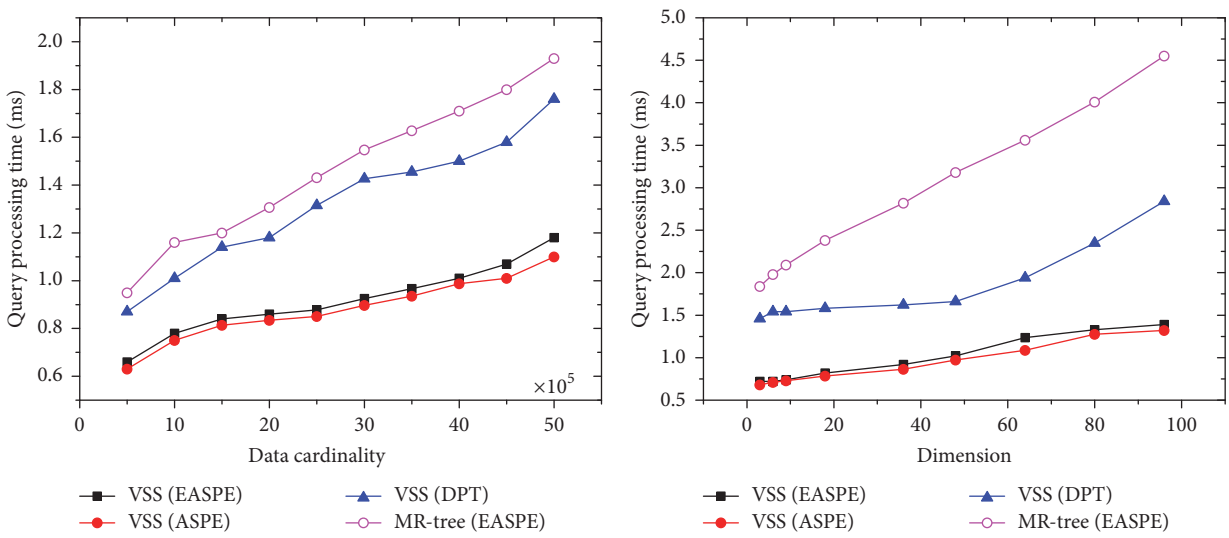


Figure 8: Fanout of internal node.



(a) Query time versus cardinality ($d = 6$)

(b) Query time versus dimension ($n = 100$ K)

Figure 9: kNN query processing time.

following: scanning *VO*, hash computation, scalar product computation and comparison, and signature verification.

The verification time is shown in Figure 11. We can see that the verification time is proportional to the data cardinality. The verification time under EASPE is shorter than that under DPT. The reason is that *kNN* verification algorithm computes $p' \cdot q'$ under EASPE, while it computes $dist(p, q)$ under DPT.

8. Conclusion

In this paper, EASPE is firstly introduced to support secure *kNN* query. EASPE is not distance-recoverable and only preserves the scalar products between data points in database and query points. In addition, we proposed a verifiable spatial data index structure VSS-tree to improve *kNN* query efficiency and provide *kNN* query verification. The security analysis and experiment results show that EASPE can resist level 2 attacks; the cloud server can efficiently perform a *kNN* query on encrypted data points and query points. The encryption cost, *kNN* query cost, and verification cost can meet the practical requirement.

In the future, the actual application scenarios shall be considered that there are more than one data source or the outsourced databases distributed on diverse cloud service providers. The VSS-tree shall be extended to support query authentication with multiple data sources or distributed databases.

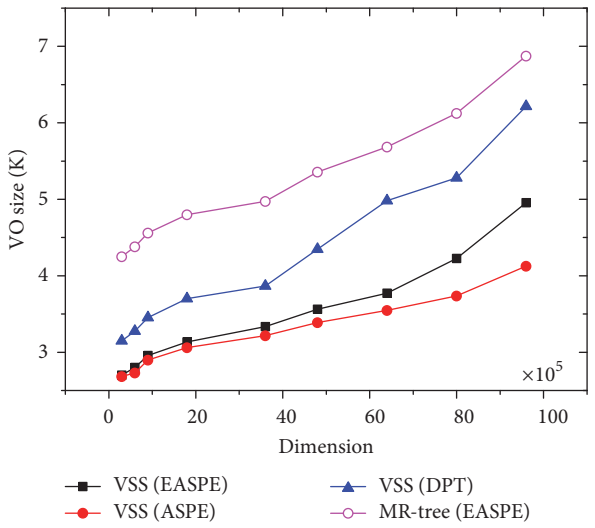


Figure 10: VO size versus data cardinality ($d = 6$).

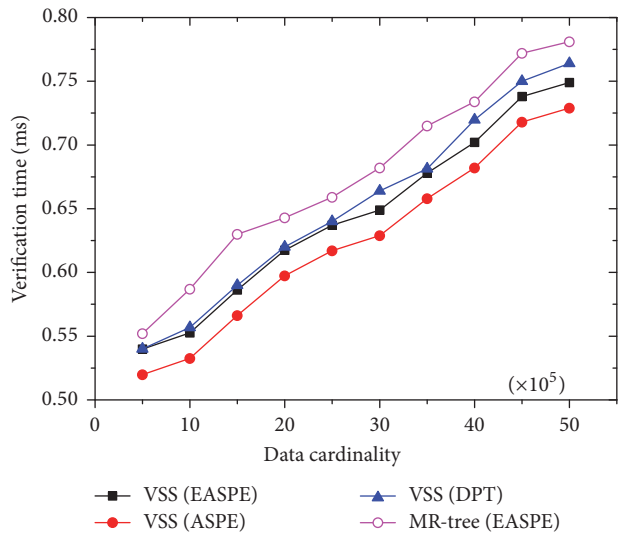


Figure 11: Verification time versus cardinality ($d = 6$).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research is supported by the National Nature Science Foundation of China (nos. 61772101, 61170169, 61170168, and 61602075).

References

[1] M. Armbrust, F. Armando, R. Griffith, D. A. Joseph, and R. H. Katz, "Technical Report UCB/EECS-2009-28," Tech. Rep., Eecs Department, University of California, Berkeley, 2009.

[2] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of The ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer Applications*, vol. 40, no. 1, pp. 325–344, 2014.

[4] P. M. Mell and T. Grance, "The NIST Definition of Cloud Computing," Tech. Rep. SP 800-145, National Institute of Standards & Technology, 2011.

[5] I. Emanuel Baciu, "Advantages and disadvantages of cloud computing services," *from the employees point of view*, 2015.

[6] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[7] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

[8] E. Mykletun and G. Tsudik, "Aggregation queries in the database-as-a-service model," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 4127, pp. 89–103, 2006.

[9] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '02)*, pp. 216–227, Madison, Wis, USA, June 2002.

[10] R. Agrawal, J. Kiernan, R. Srikant, and Y. R. Xu, "Order preserving encryption for numeric data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*, pp. 563–574, ACM, Paris, France, June 2004.

[11] S. R. M. Oliveira and O. R. Zaiane, "Privacy preserving clustering by data transformation," *Journal of Information and Data Management*, vol. 1, no. 1, p. 37, 2010.

[12] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems (SIGMOD-PODS '09)*, pp. 139–152, Providence, RI, USA, July 2009.

[13] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN computation on encrypted cloud data without sharing key with query users," in *Proceedings of the 2013 1st International Workshop on Security in Cloud Computing, Cloud Computing 2013*, pp. 55–60, China, May 2013.

[14] Y. Zhu, Z. Huang, and T. Takagi, "Secure and controllable k-NN query over encrypted cloud data with key confidentiality," *Journal of Parallel and Distributed Computing*, vol. 89, pp. 1–12, 2016.

[15] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving kNN queries," in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE 2011*, pp. 565–576, April 2011.

[16] Y. Jing, L. Hu, W.-S. Ku, and C. Shahabi, "Authentication of k nearest neighbor query on road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 6, pp. 1494–1506, 2014.

[17] D. A. White and R. Jain, "Similarity indexing with the SS-tree," in *Proceedings of the 1996 IEEE 12th International Conference on Data Engineering*, pp. 516–523, March 1996.

[18] H. Delfs and H. Knebl, *Introduction to cryptography*, Information Security and Cryptography, Springer, Berlin, Second edition, 2007.

[19] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Knowledge Discovery in Databases: PKDD 2006*, vol. 4213 of *Lecture Notes in Computer Science*, pp. 297–308, Springer, Berlin, Germany, 2006.

[20] L. Y. Man, G. Ghinita, C. S. Jensen, and P. Kalnis, "Outsourcing search services on private spatial data," in *Proceedings of the 25th IEEE International Conference on Data Engineering, ICDE 2009*, pp. 1140–1143, China, April 2009.

[21] K. Chen, R. Kavuluru, and S. Guo, "RASP: Efficient multidimensional range query on attack-resilient encrypted databases," in *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, CODASPY'11*, pp. 249–260, USA, February 2011.

[22] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 12, pp. 1719–1733, 2007.

[23] C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: query processing for location services without compromising privacy," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 4, article 24, 2009.

[24] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan, "Verifying completeness of relational query results in data publishing," in *Proceedings of the SIGMOD 2005: ACM SIGMOD International Conference on Management of Data*, pp. 407–418, USA, June 2005.

[25] M. Narasimha and G. Tsudik, "DSAC: Integrity for outsourced databases with signature aggregation and chaining," in *Proceedings of the CIKM'05 - Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pp. 235-236, Germany, November 2005.

[26] M. Zhang, C. Hong, and C. Chen, "Server transparent query authentication of outsourced database," *Journal of Computer Research and Development*, vol. 1, no. 28, pp. 182–190, 2010.

[27] J. Hong, T. Wen, Q. Gu, and G. Sheng, "Query integrity verification based-on MAC chain in cloud storage," in *Proceedings of the 2014 13th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2014 - Proceedings*, pp. 125–129, China, June 2014.

[28] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Cryptology — EUROCRYPT 2003*, vol. 2656 of *Lecture Notes in Computer Science*, pp. 416–432, Springer, Berlin, Heidelberg, 2003.

[29] R. C. Merkle, "A certified digital signature. in advances in cryptology," in *Proceedings of the International Cryptology Conference, CRYPTO 89*, pp. 218–238, Santa Barbara, Calif, Usa, 1989.

[30] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine, "Authentic data publication over the internet," *Journal of Computer Security*, vol. 11, no. 3, pp. 291–314, 2003.

[31] H. H. Pang and K. L. Tan, "Authenticating query results in edge computing," in *Proceedings of the Proceedings - 20th International Conference on Data Engineering - ICDE 2004*, pp. 560–571, USA, April 2004.

[32] F. Li, M. Hadjieftheriou, G. Kollios, and L. Reyzin, "Authenticated index structures for outsourced databases," *Handbook of Database Security: Applications and Trends*, pp. 115–136, 2008.

[33] W. Cheng, H. H. Pang, and K. L. Tan, "Authenticating multidimensional query results in data publishing," in *Lecture Notes in Computer Science*, vol. 4127, pp. 60–73, 2006.

[34] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Spatial outsourcing for location-based services," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE'08*, pp. 1082–1091, April 2008.

[35] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity auditing of outsourced data," in *Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007*, pp. 782–793, aut, September 2007.

[36] H. Wang, J. Yin, C.-S. Perng, and P. S. Yu, "Dual encryption for query integrity assurance," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM'08*, pp. 863–872, USA, October 2008.

[37] B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao, "Privacy-preserving computation and verification of aggregate queries on outsourced databases," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 5672, pp. 185–201, 2009.

[38] B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao, "Privacy-aware verification of aggregate queries on outsourced databases with applications to historic data integrity," *In Privacy Enhancing Technologies*, 2009.

[39] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *Information Security and Cryptology – ICISC 2004*, vol. 3506 of *Lecture Notes in Computer science*, pp. 104–120, Springer, 2005.

[40] C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 1592, pp. 402–414, 1999.

[41] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pp. 47–57, ACM, 1984.