International Journal of Web Services Research
Volume 21 • Issue 1

A Secure and Fast Range Query Scheme for Encrypted Multi-Dimensional Data

Zhuolin Mei, Jiujiang University, China
Huilai Zou, Zhejiang Institute of Mechanical and Electrical Engineering, China*
Jinzhou Huang, Hubei University of Arts and Science, China
Caicai Zhang, Zhejiang Institute of Mechanical and Electrical Engineering, China
Bin Wu, Jiujiang University, China
Jiaoli Shi, Jiujiang Key Laboratory of Cyberspace and Information Security, China
Zhengxiang Cheng, Jiujiang University, China

ABSTRACT

In recent years, more and more data has been stored on the cloud to provide various services. These data often contain users' private information, which inevitably raises concerns about data security. Encryption before outsourcing is a direct solution to mitigate these concerns. However, traditional encryption schemes such as block encryption make basic data services hard to support. Therefore, this paper proposes a secure and fast range query scheme for encrypted multi-dimensional data, called SFRQ. The scheme constructs a secure index over the ciphertexts of multi-dimensional data, utilizing the R-tree index, Bloom filter, and 0-1 encoding techniques. This secure index enables the cloud to provide fast range query services over the ciphertexts of multi-dimensional data. The authors have evaluated SFRQ through extensive experiments, which demonstrate its high efficiency. Additionally, the security analysis shows that no external entity, including the cloud, can obtain additional information during the entire query process.

KEYWORDS

Ciphertext Multi-Dimensional Data, Cloud Computing, Encryption, Range Search Service, Secure Index

INTRODUCTION

Cloud computing has been widely adopted by individuals, organizations, and businesses (Zeng et al., 2020; Mei et al., 2024). Many applications (Wang et al., 2022) have the capacity to leverage cloud servers for outsourcing their data and services, thereby improving the quality of the services they offer. Thus, the cloud often contains a substantial volume of data, which frequently encompasses sensitive information. Therefore, data security in the cloud becomes a popular research area in both academic and business communities (Zeng et al. 2017; Wu et al., 2020; Wu et al., 2021). To tackle these security concerns, one of the most straightforward approaches is to employ data encryption prior to outsourcing. Nevertheless, traditional encryption methods are difficult to support in some

DOI: 10.4018/IJWSR.340391
*Corresponding Author

basic data operations, such as data retrieval. Although some new encryption schemes can be used to address the problem of ciphertext search, there exist some constraints.

Agrawal et al. (2004) proposed the first order-preserving encryption (OPE), which aims to incorporate order information of plaintexts into the corresponding ciphertexts. As a result, an OPE scheme is very suitable to solve the problem of ciphertext search. However, many OPE schemes (Agrawal et al., 2004; Peng et al., 2017; Popa et al., 2011; Quan et al., 2018) mainly consider ciphertext search for single-dimensional data (Zhan et al., 2022). Moreover, due to the disclosure of order information caused by OPE schemes, this can be used to accurately deduce the plaintexts (David et al., 2004). Therefore, OPE schemes pose potential data security risks.

Bucketization schemes (Wang et al., 2013; Hore et al., 2004; Hore et al., 2012) can protect order information of ciphertexts and enable ciphertext querying. In a bucketization scheme, all the data is partitioned and placed into different buckets. The data in each bucket are treated as a unit and encrypted. Thus, the order information of ciphertexts in the same bucket can be protected well. Suppose B is a bucket and Q is a queried range. If $B \cap Q \neq \varnothing$, all the ciphertexts in B are as the results for Q and finally returned to the data user. To enhance the efficiency of bucketization schemes, researchers have proposed bucketization-based index schemes (Wang et al., 2013; Mei et al., 2018). Nevertheless, the scheme devised by Wang et al. (2013) involves many matrix operations, resulting in low efficiency. The scheme of Mei et al. (2018) exhibits suboptimal performance when dealing with datasets that have non-uniform distributions.

In this paper, we propose a secure and fast range query scheme for encrypted multi-dimensional data, namely SFRQ. In our scheme, a normal R-tree, 0-1 encoding (Gupta et al., 2001), and Bloom filter (Bloom et al., 1970) are used to construct a secure R-tree index. 0-1 encoding and a Bloom filter are used to process the minimum bounding rectangle (MBR) corresponding to each node in the R-tree. This allows each processed MBR to be securely and effectively determined whether the query range intersects with it. The data in each bucket are treated as a unit and encrypted. We conducted a large number of simulation experiments, and the results show that the proposed scheme SFRQ exhibits a high search efficiency. The contributions of this paper are as follows.

1. We have developed a secure R-tree index by leveraging a conventional R-tree, 0-1 encoding, and Bloom filter.
2. We propose a secure and fast range query scheme for encrypted multi-dimensional data, namely SFRQ, by using the proposed secure R-tree index.
3. We carry out extensive experiments to evaluate the efficiency and provide a thorough analysis of correctness and security.

RELATED WORK

An OPE scheme was first proposed by Agrawal et al. (2004). As the order information of plaintexts is preserved in the corresponding ciphertexts, i.e., larger plaintexts correspond to larger ciphertexts, OPE enables ciphertext search without decryption. A strict definition for the security of OPE was proposed by Boldyreva et al. (2009), but unfortunately, there is no OPE that satisfies the strict definition. Therefore, they propose a weaker definition, i.e., ciphertexts are indistinguishable from the values calculated by a random increment function, and then construct an instance of OPE that meets the weaker definition. Since then, many researchers have conducted extensive studies (Boldyreva et al., 2011; Dyer et al., 2017; Krendelev et al., 2014; Teranishi et al., 2014; Xiao et al., 2012) based on the work of Boldyreva et al. (2009). However, most of these OPE schemes only study the single-dimensional data. In recent years, Zhan et al. (2022) proposed a scheme that organizes all the data in a network data structure and uses prefix encoding and a Bloom filter to process the values stored in the structure, enabling the execution of range searches on encrypted multi-dimensional data (MDD). Unfortunately, the leakage of order information in OPE is likely inevitable.

A bucketization scheme was first proposed by Hacigümüş (2002). Then, Hore et al. (2004) discussed how bucketization is good for both search efficiency and security. Following Hacigümüş's and Hore's works, many studies have been done to improve bucketization schemes in many aspects. To improve the search efficiency, Lee (2014) set an order for all the buckets. In the aforementioned schemes, the buckets must be stored and searched locally on the data user side. Wang et al. (2013) adopted a matrix encryption (Wong et al., 2009) to encrypt the buckets, then organize these encrypted buckets into an index, and finally outsource the index to the cloud. Mei et al. (2018) also built an index to enable the execution of range searches on encrypted MMD. Unfortunately, their scheme is not very suitable for uniformly distributed data sets.

Background Knowledge

We outline below several key concepts that form the basis of our scheme, including the R-tree (Guttman et al., 1984), 0-1 encoding technique (Lin et al., 2005), and Bloom filter (Bloom et al., 1970).

An R-tree is a height-balanced tree. Each node of an R-tree contains an MBR. The MBR of an internal node covers the union of the MBRs of its child nodes. Each leaf node is linked to a bucket, and all the data covered by its MBR are stored in that bucket.

A Bloom filter is a probabilistic data structure utilized for membership testing of elements within a set. A Bloom filter contains a bit array A in which all the bits are initialized 0, k hash functions $h_1, h_2, \ldots, h_k$ and a data set D.

(1) When adding an element d_j in D, the Bloom filter sets $A[h_i(d_j)] = 1$ ($i \in [1, k]$ and $j \in [1, m]$).
(2) When testing whether an element $d' \in D$, the Bloom filter calculates $A[h_i(d')]$ ($i \in [1, k]$). If $A[h_i(d')] = 1$, there is $d' \in D$. Otherwise, there is $d' \notin D$.

It is important to note that a Bloom filter may produce false positives (an element is mistakenly regarded as belonging to the data set). However, according to the analysis of Graf et al. (2020), optimal parameter settings can minimize the occurrence of false positives. Specifically, a Bloom filter has a minimum probability of false positives, which is 2^{-k}, when $k = (n / m) \ln 2$. Here, n refers to the number of elements in the dataset, and m refers to the number of bits in the bit array.

0-1 encoding is a method of representing data using binary digits. Let $s = t_n t_{n-1} \ldots t_1 \in 0,1^n$ be a binary string of length n. Its 0-encoding form is defined as a set $S_s^0 = \{t_n t_{n-1} \ldots t_{i+1} 1 \mid t_i = 0, 1 \le i \le n\}$. Similarly, its 1-encoding form is defined as a set $S_s^1 = \{t_n t_{n-1} \ldots t_i \mid t_i = 1, 1 \le i \le n\}$. Suppose x and y are two integers, S_x^0 and S_x^1 are the 0-encoding and 1-encoding forms of x respectively, S_y^0 and S_y^1 are the 0-encoding and 1-encoding forms of y respectively. If and only if $S_x^1 \cap S_y^0 \ne \varnothing$, there is $x > y$. On the contrary, if and only if $S_x^1 \cap S_y^0 = \varnothing$, there is $x \le y$. Lin et al. (2005) have proved the above conclusions clearly.

Here is an example for illustrating data comparation by 0-1 encoding forms. Given two data 11 and 6, the 4-bit binary strings are $(1011)_2$ and $(0110)_2$ respectively. It is easy to calculate the 0-1 encoding forms of 11 and 6, i.e., $S_{11}^0 = \{11\}$, $S_{11}^1 = \{1, 101, 1011\}$, $S_6^0 = \{1, 0111\}$ and $S_6^1 = \{01, 011\}$. As $S_{11}^1 \cap S_6^0 = \{1\} \ne \varnothing$, there is $11 > 6$. On the contrary, as $S_6^1 \cap S_{11}^0 = \varnothing$, there is $6 \le 11$.

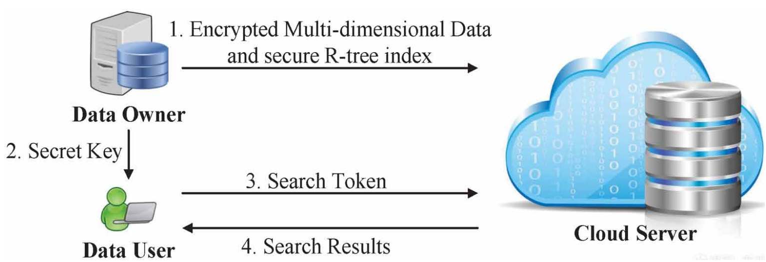
For the sake of clarity, we provide a concise overview of the symbols used in this paper in Table 1.

Figure 1 shows the system model of SFRQ. First, the data owner (DO) uses a normal R-tree (RT) to build a secure R-tree ($\overline{RT}$) by using 0-1 encoding (01E) and a Bloom filter (BF), and then encrypts all the MDD. Next, the DO outsources $\overline{RT}$ and the encrypted MDD to the cloud. Finally, the DO generates and distributes a secret key to the data user (DU), who uses it to generate

Table 1. Notations and Explanations

Notation	Explanation
DO	Data Owner
DU	Data User
01E	0-1 Encoding
0E	0 Encoding
1E	1 Encoding
BF	Bloom Filter
RT	Normal R-tree
$\overline{RT}$	Secure R-tree
MDD	multi-dimensional data

Figure 1. System Model



a search token for the queried range. The DU then sends the search token to the cloud. Upon receiving the search token, the cloud performs a range search over $\overline{RT}$ and sends the search results back to the DU. Finally, the DU decrypts the received ciphertexts using the same secret key as the DO. In our system model, we assume that the cloud is semi-trusted, meaning that it follows designated protocols and procedures but may have various reasons to be curious, including being compromised to act on behalf of a third party.

Definition 1: Correctness. Suppose $C^* = \{C_1^*, C_2^*, \ldots, C_i^*\}$ is the search results over $\overline{RT}$ by using a queried range Q. A bucketization-based range search scheme is correct if the plaintext d_j of C_j^* falls within the MBR that intersects with Q.

Definition 2: Security (Zhan et al., 2022; Guo et al., 2018). Suppose F is a leakage function. If no adversary is able to obtain information apart from F, SFRQ is considered secure. The leakage function is $F(x,y) = position_{diff}(x,y)$, where $position_{diff}(x,y)$ returns the position of the first difference between x and y.

Construction of SFRQ

In this section, we first present an overview of SFRQ, then describe the secret key generation algorithm, followed by the MBR encoding algorithm, the index construction algorithm, the search token generation algorithm and, finally, the range search algorithm in details. The scheme SFRQ begins with the DO building RT over all the MDD. Each MBR in RT is then processed using 01E and BF. Specifically, the DO first transforms each boundary information of an MBR to its binary string. Then, the DO pads a random number after the binary string. Next, the DO applies 01E to process the binary string with the random number. Finally, the DO obtains a bit array using a BF, which uses hash functions that take a binary string and the DO's secret key as the input to ensure the security of the bit array. Additionally, the DO uses a secure encryption scheme to encrypt all the MDD for data security. After processing all the MBRs and encrypting all the MDD, the DO obtains $\overline{RT}$ and all the encrypted MDD. The DO then outsources $\overline{RT}$ and encrypted MDD to the cloud. For a queried range, the DU generates several hash values as the search token using his or her secret key and hash functions in the BF, and sends the search token to the cloud. After receiving the search token, the cloud executes a range search and delivers the search results to the DU. Finally, the DU decrypts all the ciphertexts in the search results using his or her secret key, which is the same as the DO's secret key.

The construction of the SFRQ scheme involves the following probability polynomial time algorithms.

Secret key generation algorithm $KeyGen(1^\lambda) \rightarrow SK$: It takes a security parameter λ as the input and generates a secret key SK as the output, which is executed by DO.

(1). Suppose $SE = (SE.Gen, SE.Enc, SE.Dec)$ is a secure encryption scheme. $KeyGen$ executes $sk_1 = SE.Gen(1^\lambda)$. sk_1 is as the first part of SK and sk_1 is used to encrypt all the outsourced MDD before outsourcing.

(2). $KeyGen$ randomly chooses $sd_1, sd_2, \ldots, sd_k$ as the seeds for the k hash functions in BF. These seeds $sk_2 = (sd_1, sd_2, \ldots, sd_k)$ is as the second part of SK . Note that, each hash function h_i takes a value and a seed sd_i ($i \in [1, k]$) as the inputs and outputs a hash value. Without the seeds $sk_2 = (sd_1, sd_2, \ldots, sd_k)$, h_i cannot output the correct hash value. Hence, this type of hash functions in BF can ensure the security of $\overline{RT}$. This type of hash functions in BF can also ensure the security of the search tokens.

(3). $KeyGen$ outputs $SK = (sk_1, sk_2)$.

MBR encoding algorithm $MBREncoding(MBR) \rightarrow C_{MBR}$: It takes an MBR MBR as the input and generates the encoded from of MBR (denoted by C_{MBR}) as the output, which is recalled by $IndexGen$ (see the following paragraphs).

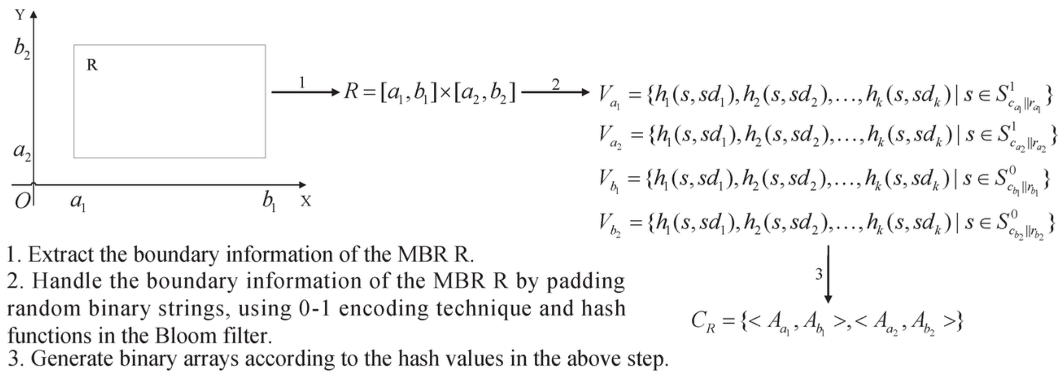
For illustration purposes, we suppose $MBR = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$, where $[a_i, b_i]$ is the range on the i -th dimension, d is the dimensionality, a_i and b_i are the minimum value and maximum value of $[a_i, b_i]$.

(1). $MBREncoding$ creates two bit arrays A_{a_i} and A_{b_i} , where each bit in A_{a_i} and A_{b_i} is initialized to 0 ($i \in [1, d]$).

(2). $MBREncoding$ encodes a_i to its binary string form, denoted by c_{a_i} , and encodes b_i to its binary string form, denoted by c_{b_i} . The length of a_i and b_i is set to l . The high positions of c_{a_i} or c_{b_i} should be padded with 0s if the length of c_{a_i} or c_{b_i} is less than l .

(3). For the security concern, $MBREncoding$ randomizes the binary string forms of c_{a_i} and c_{b_i} . Namely, $MBREncoding$ pads a l -length random binary string r_{a_i} after c_{a_i} , i.e., $c_{a_i} \,||\, r_{a_i}$. By

Figure 2. MBR Encoding



executing the same processes, $MBREncoding$ also pads another l -length random binary string r_{b_i} after c_{b_i} , i.e., $c_{b_i} \| r_{b_i}$. As r_{a_i} and r_{b_i} are both l -length random binary strings, and $a_i < b_i$, the value of $c_{a_i} \| r_{a_i}$ is smaller than the value of $c_{b_i} \| r_{b_i}$.

(4). For data comparison purposes, $MBREncoding$ calculates $S^1_{c_{a_i}\|r_{a_i}}$, which is the 1E form of $c_{a_i} \| r_{a_i}$, and then uses $h_1, h_2, \ldots, h_k$ and $sk_2 = (sd_1, sd_2, \ldots, sd_k)$ to process each element in $S^1_{c_{a_i}\|r_{a_i}}$. In particular, $MBREncoding$ calculates a set of hash values $V_{a_i} = \{h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) \mid s \in S^1_{c_{a_i}\|r_{a_i}}\}$, and then sets the bit at the v ($v \in V_{a_i}$) position of A_{a_i} to 1. By executing the similar processes, $MBREncoding$ calculates $S^0_{c_{b_i}\|r_{b_i}}$, which is the 0E form of $c_{b_i} \| r_{b_i}$. Then, $MBREncoding$ calculates a set of hash values $V_{b_i} = \{h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) \mid s \in S^0_{c_{b_i}\|r_{b_i}}\}$, and finally sets the bit at the v ($v \in V_{b_i}$) position of A_{b_i} to 1.

(5). $MBREncoding$ calculates the encoded form of MBR , and outputs $C_{MBR} = \{< A_{a_i}, A_{b_i} > \mid i \in [1, d]\}$.

Example 1. As shown in Figure 2, first, $MBREncoding$ extracts the boundary information of the MBR in the planar coordinate system, which is $R = [a_1, b_1] \times [a_2, b_2]$. Second, $MBREncoding$ calculates the binary strings of a_1 , a_2 , b_1 and b_2 respectively, chooses four random binary strings, and pads these binary strings after a_1 , a_2 , b_1 and b_2 respectively. Then, $MBREncoding$ calculates the 1E form for the minimum value a_1 and a_2 , and calculates the 0E form for the maximum value b_1 and b_2 . Next, $MBREncoding$ processes the 0E and 1E by using BF. Finally, $MBREncoding$ calculates the encoded form of $R = [a_1, b_1] \times [a_2, b_2]$, which is $C_R = \{< A_{a_1}, A_{b_1} >, < A_{a_2}, A_{b_2} >\}$.

Index construction algorithm $IndexGen(RT) \to \overline{RT}$: It takes RT as the input and constructs $\overline{RT}$ as the output, which is executed by DO.

First, $IndexGen$ recalls $MBREncoding$ to process all the MBRs of nodes in RT . Then, $IndexGen$ recalls $SE.Enc$ to encrypt all the MDD in groups which are under the leaf nodes of T .

Finally, $IndexGen$ outputs $\overline{RT}$. Hence, in $\overline{RT}$, each node contains a processed MBR and each leaf node points to a group of encrypted MDD which are covered by the MBR of the leaf node.

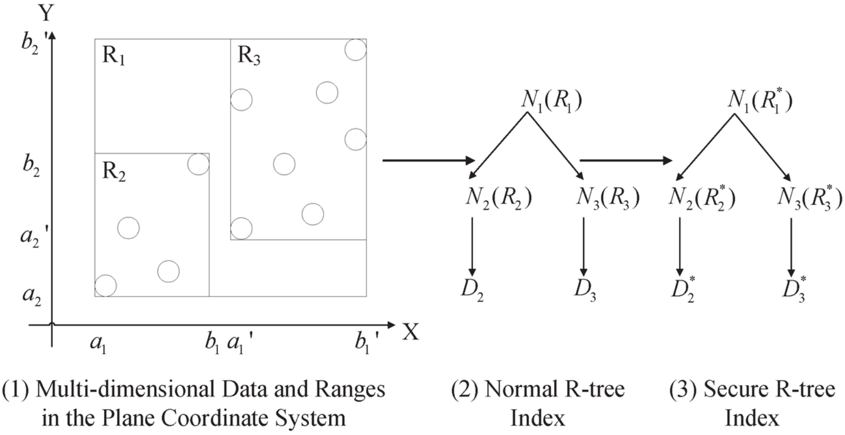
The following Example 2 shows the generation of a secure R-tree index ($\overline{RT}$).

Example 2. As shown in Figure 3 (1), all the outsourced data are two-dimensional and distributed in the planar coordinate system, which are represented by hollow circles. First, as shown in Fig. 3 (2), DO builds RT over these 2-dimensional data. In RT, each node contains an MBR. Each leaf node points to a group of 2-dimensional data. Specifically, the node N_1 contains the MBR R_1, the leaf node N_2 contains the MBR R_2 and points to a group of 2-dimensional data D_2, and the leaf node N_3 contains the MBR R_3 and points to a group of 2-dimensional data D_3. Then, DO runs $IndexGen$. $IndexGen$ recalls $MBREncoding$ to process the MBRs R_1, R_2 and R_3, and recalls $SE.Enc$ to encrypt all the 2-dimensional data in D_2 and D_3 respectively. As shown in Fig. 3 (3), the processed MBRs are denoted by R_1^*, R_2^* and R_3^*, the group of encrypted 2-dimensional data in D_2 is denoted by D_2^*, and the group of encrypted 2-dimensional data in D_3 is denoted by D_3^*. Finally, $IndexGen$ outputs $\overline{RT}$.

Search token generation algorithm $TokenGen(SK, Q) \rightarrow token_Q$: It takes the secret key SK and a queried range Q as the inputs and generates the search token $token_Q$ of Q as the output, which is executed by DU. Then, DU sends $token_Q$ to the cloud.

For illustration purposes, we suppose $Q = [p_1, q_1] \times [p_2, q_2] \times \ldots \times [p_d, q_d]$, where d is the dimensionality, $[p_i, q_i]$ is the range on the i-th dimension, d is the dimensionality and $i \in [1, d]$. $TokenGen$ encodes the minimum value p_i to its binary string form c_{p_i}, and then pads a l-length random binary string r_{p_i} after c_{p_i}. Specifically, $TokenGen$ converts p_i to $c_{p_i} \| r_{p_i}$. By using the same method, $TokenGen$ converts the maximum value q_i to $c_{q_i} \| r_{q_i}$, where c_{q_i} is the binary string form of q_i and r_{q_i} is a l-length random binary string. As r_{p_i} and r_{q_i} are both l-length random binary strings, and $p_i < q_i$, the value of $c_{p_i} \| r_{p_i}$ is smaller than the value of $c_{q_i} \| r_{q_i}$. Next, $TokenGen$ calculates the 01E forms of $c_{p_i} \| r_{p_i}$, denoted by $S^0_{c_{p_i}\|r_{p_i}}$ and $S^1_{c_{p_i}\|r_{p_i}}$ respectively. By using the hash

Figure 3. Secure R-Tree Index Construction



functions $h_1, h_2, \ldots, h_k$ in BF and the second part secret key $sk_2 = (sd_1, sd_2, \ldots, sd_k)$, $TokenGen$ calculates $token_p^0 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) > \mid s \in S^0_{c_{p_i}||r_{p_i}}, i \in [1, d]\}$ and $token_p^1 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) > \mid s \in S^1_{c_{p_i}||r_{p_i}}, i \in [1, d]\}$. By using the same method, $TokenGen$ calculates $token_q^0 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) > \mid s \in S^0_{c_{q_i}||r_{q_i}}, i \in [1, d]\}$ and $token_q^1 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) > \mid s \in S^1_{c_{q_i}||r_{q_i}}, i \in [1, d]\}$. Finally, $TokenGen$ outputs $token_Q =< token_p^0, token_p^1, token_q^0, token_q^1 >$ as the search token $token_Q$ of the queried range Q.

Range search algorithm $RangeSearch(token, \overline{RT}) \rightarrow I^*$: It takes a search token $token$ and $\overline{RT}$ as the inputs and obtain the search results I^* as the output, which is executed by the cloud server. Then, the cloud server sends I^* to DU as response.

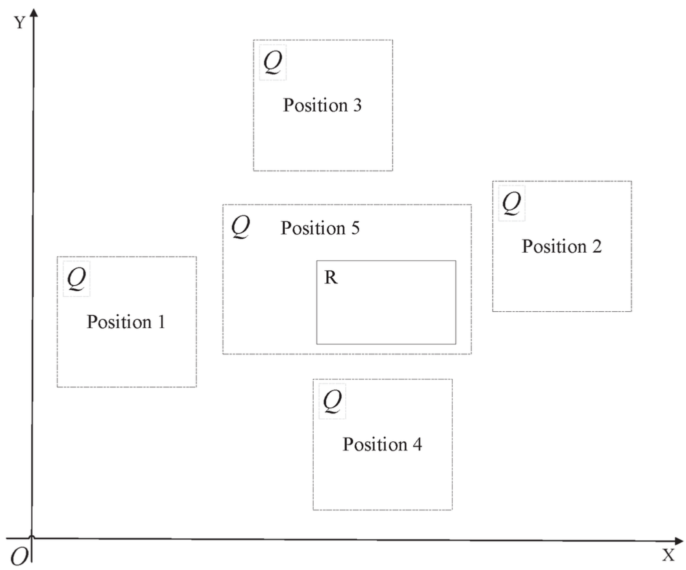
First, we introduce how to judge whether a queried range Q intersects with an MBR MBR. Then, we introduce how to perform range search over $\overline{RT}$.

For illustration purposes, we suppose $Q = [p_1, q_1] \times [p_2, q_2] \times \ldots \times [p_d, q_d]$ is a queried range and $MBR = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$ is an MBR in $\overline{RT}$, where $[p_i, q_i]$ and $[a_i, b_i]$ are the ranges on the i-th dimension respectively, d is the dimensionality, and $i \in [1, d]$. To support range search by using $\overline{RT}$, $RangeSearch$ should judge whether $Q \cap MBR \neq \varnothing$. Specifically, if $\exists i \in [1, d]$ that $q_i < a_i$ or $b_i < p_i$, there is $[p_i, q_i] \cap [a_i, b_i] = \varnothing$, i.e., there is $MBR \cap Q = \varnothing$. On the contrary, there is $Q \cap MBR \neq \varnothing$. As 01E is adopted, if $\exists i \in [1, d]$ that $S^0_{q_i} \cap S^1_{a_i} \neq \varnothing$ or $S^0_{b_i} \cap S^1_{p_i} \neq \varnothing$, there is $q_i < a_i$ or $b_i < p_i$, i.e., there is $MBR \cap Q = \varnothing$. On the contrary, there is $Q \cap MBR \neq \varnothing$. Additionally, $RangeSearch$ also judges a special intersection, i.e., $MBR \subseteq Q$. Specifically, if $\forall i \in [1, d]$ that $p_i < a_i$ and $b_i < q_i$, there is $[a_i, b_i] \subseteq [p_i, q_i] \neq \varnothing$, i.e., there is $MBR \subseteq Q$. On the contrary, there is $MBR \not\subset Q$. As 01E is adopted, if $\forall i \in [1, d]$ that $S^0_{p_i} < S^1_{a_i}$ and $S^0_{b_i} < S^1_{q_i}$, there is $p_i < a_i$ and $b_i < q_i$, i.e., there is $MBR \subseteq Q$. On the contrary there is $MBR \not\subset Q$.

As shown in Figure 4, the queried range is $Q = [p_1, q_1] \times [p_2, q_2]$ and the MBR is $R = [a_1, b_1] \times [a_2, b_2]$. When R is at the position 1, as $q_1 < a_1$ (according to 01E, $q_1 < a_1$ indicates $S^0_{c_{q_1}||r_{q_1}} \cap S^1_{c_{a_1}||r_{a_1}} \neq \varnothing$), there is $[p_1, q_1] \cap [a_1, b_1] = \varnothing$, i.e., $Q \cap R = \varnothing$. Thus, if $S^0_{c_{q_1}||r_{q_1}} \cap S^1_{c_{a_1}||r_{a_1}} \neq \varnothing$, there is $Q \cap R = \varnothing$. Similarly, when R is at the position 2, position 3 and position 4, as $b_1 < p_1$ (i.e., $[p_1, q_1] \cap [a_1, b_1] = \varnothing$ and $S^1_{c_{p_1}||r_{p_1}} \cap S^0_{c_{b_1}||r_{b_1}} \neq \varnothing$), $b_2 < p_2$ (i.e., $[p_2, q_2] \cap [a_2, b_2] = \varnothing$ and $S^1_{c_{p_2}||r_{p_2}} \cap S^0_{c_{b_2}||r_{b_2}} \neq \varnothing$) and $q_2 < a_2$ (i.e., $[p_2, q_2] \cap [a_2, b_2] = \varnothing$ and $S^0_{c_{q_2}||r_{q_2}} \cap S^1_{c_{a_2}||r_{a_2}} \neq \varnothing$), there is . Except for the above four situations, there is $Q \cap R \neq \varnothing$. Additionally, there is a special intersection between Q and R, i.e., $R \subseteq Q$. When R is at the position 5, there is $R \subseteq Q$ because $p_1 < a_1$, $b_1 < q_1$, $p_2 < a_2$ and $b_2 < q_2$ (according to 01E, these four inequalities indicate that $S^0_{c_{p_1}||r_{p_1}} \cap S^1_{c_{a_1}||r_{a_1}} \neq \varnothing$, $S^1_{c_{q_1}||r_{q_1}} \cap S^0_{c_{b_1}||r_{b_1}} \neq \varnothing$, $S^0_{c_{p_2}||r_{p_2}} \cap S^1_{c_{a_2}||r_{a_2}} \neq \varnothing$ and $S^1_{c_{q_2}||r_{q_2}} \cap S^0_{c_{b_2}||r_{b_2}} \neq \varnothing$). Thus, if $S^0_{c_{p_1}||r_{p_1}} \cap S^1_{c_{a_1}||r_{a_1}} \neq \varnothing$, $S^1_{c_{q_1}||r_{q_1}} \cap S^0_{c_{b_1}||r_{b_1}} \neq \varnothing$, $S^0_{c_{p_2}||r_{p_2}} \cap S^1_{c_{a_2}||r_{a_2}} \neq \varnothing$ and $S^1_{c_{q_2}||r_{q_2}} \cap S^0_{c_{b_2}||r_{b_2}} \neq \varnothing$, there is $R \subseteq Q$. Thus, by using the above method, $RangeSearch$ can judge whether $Q \cap R \neq \varnothing$ and $R \subseteq Q$.

Note that, to ensure the security of 01E, BF with special hash functions is adopted. As all the MBRs in $\overline{RT}$ and the queried range Q have been processed by 01E, and then processed by BF,

Figure 4. The Judgment of Queried Range and MBR



The minimal coordinate value in R is (a_1, a_2) The minimal coordinate value in R is (p_1, p_2)

The maximal coordinate value in R is (b_1, b_2) The maximal coordinate value in R is (q_1, q_2)

$RangeSearch$ can determine whether an MBR intersects with or covered by a queried range by using the corresponding binary arrays and hash values. The details are as follows.

For the queried range $Q = [p_1, q_1] \times [p_2, q_2] \times \ldots \times [p_d, q_d]$ and the MBR $MBR = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$, if $\exists i \in [1, d]$, there exists a tuple in $token_q^0 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) >| \, s \in S^0_{c_{q_i} \| r_{q_i}} \}$, such that all the bits at $h_1(s, sd_1)$, $h_2(s, sd_2)$, …, $h_k(s, sd_k)$ positions of the binary array A_{a_i} are 1, it means that $q_i < a_i$. If $\exists i \in [1, d]$, there exists a tuple in $token_p^1 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) >| \, s \in S^1_{c_{p_i} \| r_{p_i}} \}$, such that all the bits at $h_1(s, sd_1)$, $h_2(s, sd_2)$, …, $h_k(s, sd_k)$ positions of the binary array A_{b_i} are 1, it means that $p_i > b_i$. If $RangeSearch$ determines $q_i < a_i$ or $p_i > b_i$, there is $[p_i, q_i] \cap [a_i, b_i] = \varnothing$, i.e., $MBR \cap Q = \varnothing$. On the contrary, there is $MBR \cap Q \neq \varnothing$. If $\forall i \in [1, d]$, there exists a tuple in $token_p^0 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) >| \, s \in S^0_{c_{p_i} \| r_{p_i}} \}$, such that all the bits at $h_1(s, sd_1)$, $h_2(s, sd_2)$, …, $h_k(s, sd_k)$ positions of the binary array A_{a_i} are 1, it means that $p_i < a_i$. If $\forall i \in [1, d]$, there exists a tuple in $token_q^1 = \{< h_1(s, sd_1), h_2(s, sd_2), \ldots, h_k(s, sd_k) >| \, s \in S^1_{c_{q_i} \| r_{q_i}} \}$, such that all the bits at $h_1(s, sd_1)$, $h_2(s, sd_2)$, …, $h_k(s, sd_k)$ positions of the binary array A_{b_i} are 1, it means that $q_i > b_i$. If $RangeSearch$ determines $p_i < a_i$ and $q_i > b_i$, there is $[a_i, b_i] \subseteq [p_i, q_i]$, i.e., $MBR \subseteq Q$. On the contrary, there is $MBR \not\subset Q$.

According to the above method, by determining whether all the bits at the hash value positions in the corresponding BF array are 1, $RangeSearch$ can determine whether $MBR \cap Q \neq \varnothing$ and $MBR \subseteq Q$.

For ease of explanation, we suppose N is a node in $\overline{RT}$ and N is associated with the MBR MBR. If $RangeSearch$ determines $MBR \subseteq Q$, all the encrypted MDD in MBR is added to the result set. If $RangeSearch$ determines $MBR \not\subseteq Q$ and $Q \cap MBR \neq \varnothing$, the MBRs of descendant nodes of N are iteratively judged. When Q intersects with or covers the MBR of a leaf node, all the encrypted MDD in MBR of the leaf node is added to the result set. By using the search token $token_Q = < token_p^0, token_p^1, token_q^0, token_q^1 >$, $RangeSearch$ performs range search in $\overline{RT}$ in a top-down manner. Finally, $RangeSearch$ returns the search results I^* (i.e., result set) to DU as response.

Decryption algorithm $Dec(SK, I^*) \rightarrow I$: It takes the search results I^* as the inputs and outputs the plaintext I through decrypting the ciphertexts with the first part secret key sk_1, i.e. $I = SE.Dec(I^*, sk_1)$, which is executed by DU.

EXPERIMENTS

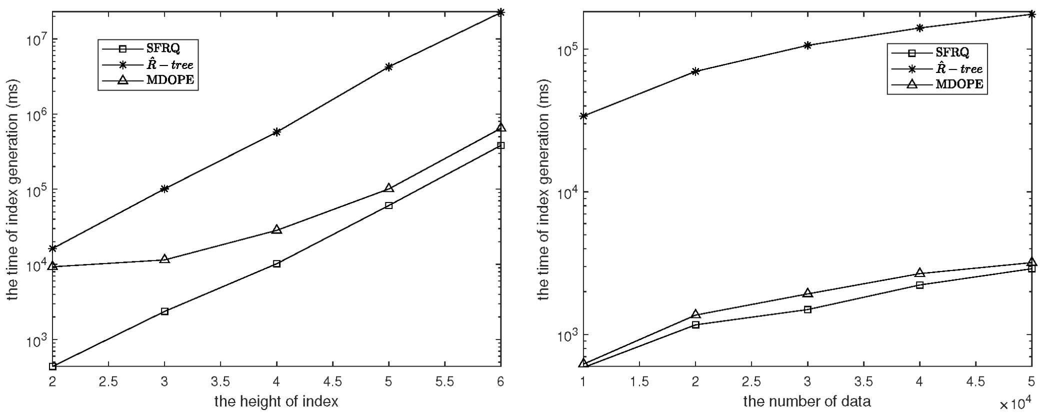
In the experiments, we compare the $\hat{R}$-tree scheme (Wang et al., 2013), the MDOPE scheme (Zhan et al., 2022) and our SFRQ scheme. These implementations were carried out on a personal computer equipped with an AMD Ryzen 5 2500U CPU and 8GB RAM, utilizing the Java programming language. For the -tree scheme, we adopt the asymmetric scalar-product preserving encryption (ASPE) of Wong et al. (2009), which is implemented using the Jama Library version 1.0.3 (Hicklin et al., 2022). In our experiments, we choose some uniformly random two-dimensional data to test the efficiency of the above schemes. In the $\hat{R}$-tree scheme and the SFRQ scheme, the fan-out of the indexes is set to six. It means that each two-dimensional range is divided into at most six smaller two-dimensional ranges. In order to achieve fairness in experimental comparisons, in the MDOPE scheme, each node on the first dimension contains only one split data, and each node on the second dimension contains two split data. This is because the range on the first dimension is divided into two smaller ranges by using one split data, and the range on the second dimension is divided into three smaller ranges by using two split data. According to the Cartesian product, in the MDOPE scheme, a two-dimensional range is divided into six smaller two-dimensional ranges. Additionally, MDOPE supports accurate range search. In order to compare the MDOPE scheme, the $\hat{R}$-tree scheme and the SFRQ scheme fairly, we set the MBR of each leaf node in the $\hat{R}$-tree scheme and the SFRQ scheme only contains one datum.

Index Construction

As shown on the left side of Figure 5, when the height of index increases, the times of index construction in the $\hat{R}$-tree scheme, the MDOPE scheme and the SFRQ scheme increase exponentially. As shown on the right side of Figure 5, when the number of data increases, the times of index construction in the $\hat{R}$-tree scheme, the MDOPE scheme and the SFRQ scheme increase linearly. Compared with the $\hat{R}$-tree scheme and the MDOPE scheme, the index construction in the SFRQ scheme is more efficient.

The indexes in the $\hat{R}$-tree scheme, the MDOPE scheme, and the SFRQ scheme are tree structure. As the number of index nodes exponentially increases with the growth of index height, the construction time of the index also increases exponentially. When the number of data increases, it needs more index nodes to index these data. In the MDOPE scheme, the index should handle each datum. According to our experimental setting, in the $\hat{R}$-tree scheme and the SFRQ scheme, the index should handle each MBR that only contains a datum. Thus, the time of index construction in the $\hat{R}$-tree scheme, the MDOPE scheme, and the SFRQ scheme increases linearly. Additionally, the index

Figure 5. Index Construction

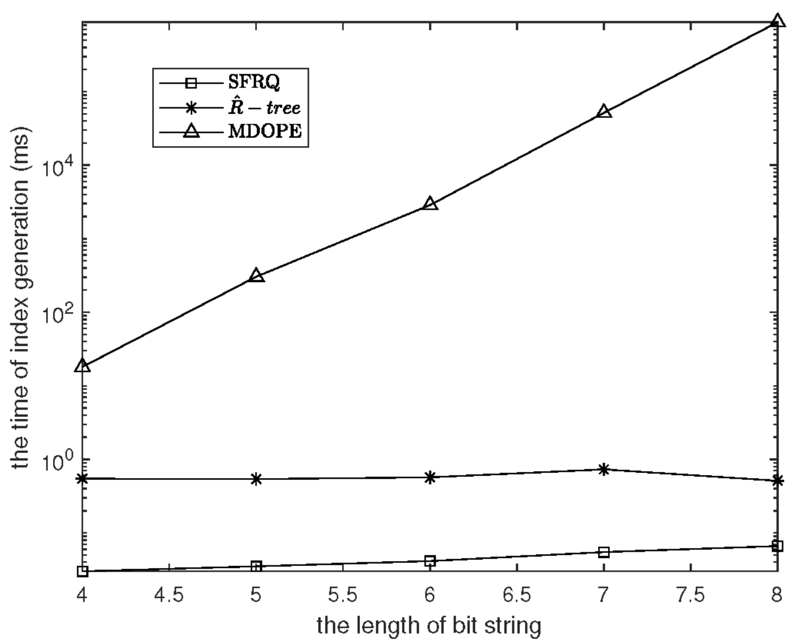


construction in the SFRQ scheme is the most efficient because the calculation of 01E in the SFRQ scheme is more efficient than that of ASPE in the $\hat{R}$-tree scheme. Although prefix encoding in MDOPE is also very efficient, there are many additional split data that should be handled. Thus, the SFRQ scheme is more efficient than the MDOPE scheme.

Search Token Generation

As shown in Figure 6, when the length of bit string increases, the time of search token generation in the MDOPE scheme increases exponentially, but the time of search token generation in the $\hat{R}$-tree scheme and the SFRQ scheme is very slowly. The search token generation in the SFRQ scheme is the most efficient.

Figure 6. Search Token Generation



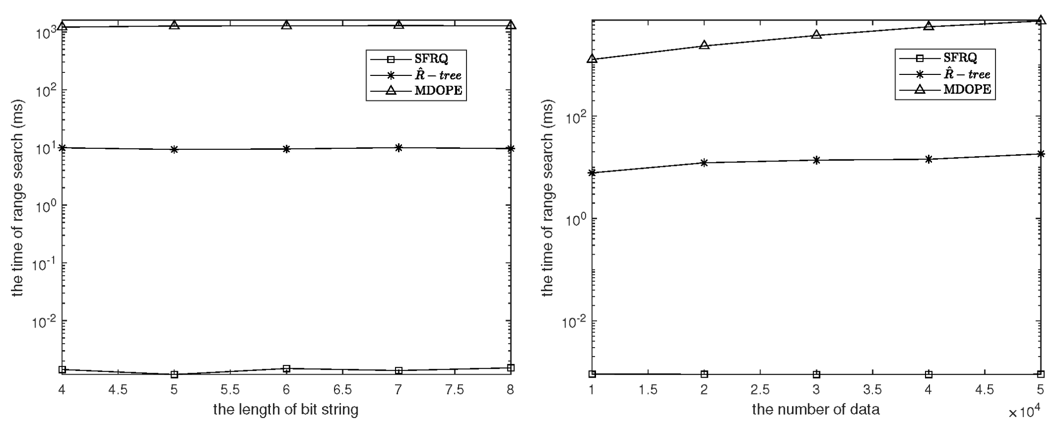
In the MDOPE scheme, a queried range is first transformed into a bit string. We suppose the length of the bit string is l. The MDOPE scheme then pads additional bit string after the original bit string. The length of the new bit string is $2l+2$. Next, the MDOPE scheme calculates the prefix encoding of the new bit string. Finally, the MDOPE scheme obtains the search token of the queried range by using BF to handle the prefix encoding of the new bit string. In the above procedure, the slowest step is that the MDOPE scheme calculates the prefix encoding of the new bit string whose length is $2l+2$. In this step, one should compare each different bit string and merge all the 2^{2l+2} different bit strings to several bit strings. Thus, the time of search token generation in the MDOPE scheme increases exponentially with the length of bit string. In the $\hat{R}$-tree scheme, a queried range is encrypted by using ASPE. The encrypted form of the queried range is as its search token. As the efficiency of ASPE is not related with the length of bit string, the time of search token generation is a constant. In the SFRQ scheme, a queried range is first transformed into a bit string. Then, the SFRQ scheme pads additional bit string after the original bit string. The length of the new bit string is $2l$. Next, the SFRQ scheme calculates 01E of the new bit string. The total number of 01E does not exceed $2l$. As the total number of bit strings is very few, the efficiency of generating the search token is remarkably high in the SFRQ scheme.

Range Search

As shown on the left side of Figure 7, when the number of data is fixed at 10000 and the length of bit string increases, the search time remains almost unchanged. As shown on the right side of Figure 7, when the number of data increases, the search times of the SFRQ scheme, the $\hat{R}$-tree scheme, and the MDOPE scheme increase. Compared with the $\hat{R}$-tree scheme and the MDOPE scheme, the SFRQ scheme is the most efficient.

As shown on the left side of Figure 7, the search time of the $\hat{R}$-tree scheme is almost unchanged because the length of bit string does not relate to the underlying encryption method ASPE. In the SFRQ scheme and the MDOPE scheme, when the length of bit string increases, the additional calculation overhead is very low, with the result that the search times almost do not increase. As shown on the right side of Figure 7, when the number of data increases, the heights of the indexes increase, with the result that the $\hat{R}$-tree scheme, the MDOPE scheme, and the SFRQ scheme should do more range search works over these indexes. Thus, the search times of these schemes increase with the volume of data. In the MDOPE scheme, many split data are inserted into the internal nodes of the index to support range search. Many comparisons work over split data result

Figure 7. Range Search



in low efficiency of range search. Additionally, the range search should be performed alone for each dimension, respectively. Thus, the range search in the MDOPE scheme is not very efficient. As the underling hash value comparison in the SFRQ scheme demonstrates superior efficiency compared to the ASPE in the $\hat{R}$-tree scheme, the SFRQ scheme demonstrates superior efficiency compared to the $\hat{R}$-tree scheme.

Analysis of Correctness and Security

Theorem 1

The SFRQ scheme complies with the correctness of Definition 1.

Proof. Suppose that $MBR = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$ is a MDR and $Q = [p_1, q_1] \times [p_2, q_2] \times \ldots \times [p_d, q_d]$ is a queried range. If $[a_1, b_1] \cap [p_1, q_1] \neq \varnothing$, $[a_2, b_2] \cap [p_2, q_2] \neq \varnothing$, …, $[a_d, b_d] \cap [p_d, q_d] \neq \varnothing$ hold, we have $\neg(a_1 > q_1 \vee b_1 < p_1) = true$, $\neg(a_2 > q_2 \vee b_2 < p_2) = true$, …, $\neg(a_d > q_d \vee b_d < p_d) = true$. Furthermore, the following equation holds. $\neg(S^1_{c_{a_1}||r_{a_1}} \cap S^0_{c_{q_1}||r_{q_1}} \neq \varnothing \vee S^0_{c_{b_1}||r_{b_1}} \cap S^1_{c_{p_1}||r_{p_1}} \neq \varnothing) = true$, $\neg(S^1_{c_{a_2}||r_{a_2}} \cap S^0_{c_{q_2}||r_{q_2}} \neq \varnothing \vee S^0_{c_{b_2}||r_{b_2}} \cap S^1_{c_{p_2}||r_{p_2}} \neq \varnothing) = true$, …, $\neg(S^1_{c_{a_d}||r_{a_d}} \cap S^0_{c_{q_d}||r_{q_d}} \neq \varnothing \vee S^0_{c_{b_d}||r_{b_d}} \cap S^1_{c_{p_d}||r_{p_d}} \neq \varnothing) = true$. Therefore, if the queried range Q intersects with the MBR MBR, there exists ciphertexts in MBR that satisfies Q. By using the algorithm $RangeSearch$ in SFRQ, all the ciphertexts in the MBRs of the leaf nodes in the secure index will be retrieved. Thus, the SFRQ scheme complies with the correctness defined in Definition 1.

Theorem 2

The SFRQ scheme adheres to the security in Definition 2.

Proof. Since the data are encrypted using a secure encryption method, the security of the data can be ensured by the encryption method. In the SFRQ scheme, the data are encrypted by using a secure encryption scheme SE. The security of the data can be guaranteed by the security of the secure encryption scheme SE. Suppose that (i) $x = x_1 x_2 \ldots x_n$ represents the boundary information of an MBR after being padded with a random value (as described in Section 5), and (ii) $y = y_1 y_2 \ldots y_n$ represents the boundary information of a queried range after being padded with a random value (as described in Section 5). If the member in the intersection of $S^0_{c_x||r_x}$ and $S^1_{c_y||r_y}$ is t, where the length of t is m, it can deduce that $x_1 = y_1$, $x_2 = y_2$, …, $x_{m-1} = y_{m-1}$, $x_m \neq y_m$. Consequently, the cloud server possesses knowledge solely of the leakage function $F(x, y) = position_{diff}(x, y)$. Hence, the SFRQ scheme adheres to the security in Definition 2.

CONCLUSION

In this paper, we propose a range search scheme, SFRQ. In the SFRQ scheme, we build a secure index $\overline{RT}$ over encrypted MDD by using a normal R-tree index RT, BF, and 01E technologies. Each node of the secure index is associated with an MBR. The boundary information of MBRs is processed by 01E. By utilizing the property of 01E, one can determine whether a queried range intersects with the MBR of a node in $\overline{RT}$. The hash functions in BF are used to ensure the security of the queried range and the MBRs of the nodes in $\overline{RT}$. Thus, the proposed SFRQ scheme can support efficient range search over ciphertexts.

ACKNOWLEDGMENT

This research was supported by the National Natural Science Foundation of China (No. 61962029, 62062045, 62262033), the Jiangxi Provincial Natural Science Foundation of China (No.20202BAB212006), the project of Zhejiang Institute of Mechanical and Electrical Engineering (No. A-0271-22-201), the Hubei Natural Science Foundation Innovation and Development Joint Fund Project (No. 2022CFD101, 2022CFD103), the Xiangyang High-tech Key Science and Technology Plan Project (No. 2022ABH006848), and the Hubei Superior and Distinctive Discipline Group of "New Energy Vehicle and Smart Transportation".

REFERENCES

Agrawal, R., Kiernan, J., Srikant, R., & Xu, Y. (2004, June). Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on management of data* (pp. 563-574). doi:10.1145/1007568.1007632

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58. doi:10.1145/1721654.1721672

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), 422–426. doi:10.1145/362686.362692

Boldyreva, A., Chenette, N., Lee, Y., & O'Neill, A. (2009). Order-preserving symmetric encryption. *Advances in cryptology-Eurocrypt 2009: 28th annual international conference on the theory and applications of cryptographic techniques*, *Cologne, Germany, April 26-30, 2009. Proceedings*, *28*, 224–241. doi:10.1007/978-3-642-01001-9_35

Boldyreva, A., Chenette, N., & O'Neill, A. (2011). Order-preserving encryption revisited: Improved security analysis and alternative solutions. *Advances in cryptology–CRYPTO 2011: 31st annual cryptology conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, *31*, 578–595. doi:10.1007/978-3-642-22792-9_33

Boneh, D., & Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. *Theory of cryptography: 4th theory of cryptography conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings*, *4*, 535–554. doi:10.1007/978-3-540-70936-7_29

Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M. C., & Steiner, M. (2013). Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in cryptology–crypto 2013: 33rd annual cryptology conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I* (pp. 353-373). doi:10.1007/978-3-642-40041-4_20

David, H. A., & Nagaraja, H. N. (2004). *Order statistics*. John Wiley & Sons. doi:10.1002/0471722162

Demertzis, I., Papadopoulos, S., Papapetrou, O., Deligiannakis, A., & Garofalakis, M. (2016, June). Practical private range search revisited. In *Proceedings of the 2016 international conference on management of data* (pp. 185-198). doi:10.1145/2882903.2882911

Dyer, J., Dyer, M., & Xu, J. (2017). Order-preserving encryption using approximate integer common divisors. In *Data privacy management, cryptocurrencies and blockchain technology: ESORICS 2017 international workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings* (pp. 257-274). doi:10.1007/978-3-319-67816-0_15

Graf, T. M., & Lemire, D. (2020). Xor filters: Faster and smaller than bloom and cuckoo filters. *ACM Journal of Experimental Algorithmics*, *25*, 1–16. doi:10.1145/3376122

Guo, J., Wang, J., Zhang, Z., & Chen, X. (2018, September). An almost non-interactive order preserving encryption scheme. In *International conference on information security practice and experience* (pp. 87-100). doi:10.1007/978-3-319-99807-7_6

Gupta, P., & McKeown, N. (2001). Algorithms for packet classification. *IEEE Network*, *15*(2), 24–32. doi:10.1109/65.912717

Guttman, A. (1984, June). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on management of data* (pp. 47-57). doi:10.1145/602259.602266

Hacigümüş, H., Iyer, B., Li, C., & Mehrotra, S. (2002, June). Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on management of data* (pp. 216-227). doi:10.1145/564691.564717

Hicklin, J., Moler, C., Webb, P., Boisvert, R. F., Miller, B., Pozo, R., & Remington, K. (2000). *Jama: A Java matrix package*. http://math. nist. gov/javanumerics/jama

Hore, B., Mehrotra, S., Canim, M., & Kantarcioglu, M. (2012). Secure multidimensional range queries over outsourced data. *The VLDB Journal*, *21*(3), 333–358. doi:10.1007/s00778-011-0245-7

Hore, B., Mehrotra, S., & Tsudik, G. (2004, August). A privacy-preserving index for range queries. In *Proceedings of the thirtieth international conference on very large data bases*. (*Vol. 30*, pp. 720-731). Academic Press.

Joan, D., & Vincent, R. (2002). *The design of Rijndael: AES-the advanced encryption standard*. Information Security and Cryptography. doi:10.1007/978-3-662-04722-4

Karras, P., Nikitin, A., Saad, M., Bhatt, R., Antyukhov, D., & Idreos, S. (2016, June). Adaptive indexing over encrypted numeric data. In *Proceedings of the 2016 international conference on management of data* (pp. 171-183). doi:10.1145/2882903.2882932

Katz, J., & Lindell, Y. (2020). *Introduction to modern cryptography*. CRC Press. https://dl.acm.org/doi/10.5555/2700550

Krendelev, S. F., Yakovlev, M., & Usoltseva, M. (2014, September). *Order-preserving encryption schemes based on arithmetic coding and matrices. In 2014 federated conference on computer science and information systems*. IEEE. doi:10.15439/2014F186

Lee, Y. (2014). Secure ordered bucketization. *IEEE Transactions on Dependable and Secure Computing*, *11*(3), 292–303. doi:10.1109/TDSC.2014.2313863

Li, R., Liu, A. X., Wang, A. L., & Bruhadeshwar, B. (2015). Fast and scalable range query processing with strong privacy protection for cloud computing. *IEEE/ACM Transactions on Networking*, *24*(4), 2305–2318. doi:10.1109/TNET.2015.2457493

Lin, H. Y., & Tzeng, W. G. (2005). An efficient solution to the millionaires' problem based on homomorphic encryption. *Applied cryptography and network security: Third international conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings*, *3*, 456–466. doi:10.1007/11496137_31

Mei, Z., Wu, B., Tian, S., Ruan, Y., & Cui, Z. (2017). Fuzzy keyword search method over ciphertexts supporting access control. *KSII Transactions on Internet and Information Systems*, *11*(11), 5671–5693. doi:10.3837/tiis.2017.11.027

Mei, Z., Yu, J., Huang, J., Wu, B., Zhao, Z., Zhang, C., & Wu, Z. (2024). Secure multi-dimensional data retrieval with access control and range query in the cloud. *Information Systems*, *102343*, 102343. Advance online publication. doi:10.1016/j.is.2024.102343

Mei, Z., Zhu, H., Cui, Z., Wu, Z., Peng, G., Wu, B., & Zhang, C. (2018). Executing multi-dimensional range query efficiently and flexibly over outsourced ciphertexts in the cloud. *Information Sciences*, *432*, 79–96. doi:10.1016/j.ins.2017.11.065

Peng, Y., Li, H., Cui, J., Zhang, J., Ma, J., & Peng, C. (2017). hOPE: Improved order preserving encryption with the power to homomorphic operations of ciphertexts. *Science China. Information Sciences*, *60*(6), 1–17. doi:10.1007/s11432-016-0242-7

Popa, R. A., Redfield, C. M., Zeldovich, N., & Balakrishnan, H. (2011, October). CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the twenty-third ACM symposium on operating systems principles* (pp. 85-100). doi:10.1145/2043556.2043566

Quan, H., Wang, B., Zhang, Y., & Wu, G. (2018). Efficient and secure top-k queries with top order-preserving encryption. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 31525–31540. doi:10.1109/ACCESS.2018.2847307

Reviriego, P., Sánchez-Macian, A., Walzer, S., Merino-Gómez, E., Liu, S., & Lombardi, F. (2022). On the privacy of counting Bloom filters. *IEEE Transactions on Dependable and Secure Computing*, *20*(2), 1488–1499. doi:10.1109/TDSC.2022.3158469

Shi, E., Bethencourt, J., Chan, T. H., Song, D., & Perrig, A. (2007, May). *Multi-dimensional range query over encrypted data. In 2007 IEEE symposium on security and privacy*. IEEE. doi:10.1109/SP.2007.29

Teranishi, I., Yung, M., & Malkin, T. (2014). Order-preserving encryption secure beyond one-wayness. In *in cryptology–ASIACRYPT 2014: 20th international conference on the theory and application of cryptology and information security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings*, *20*(Part II), 42–61. doi:10.1007/978-3-662-45608-8_3

Wang, P., & Ravishankar, C. V. (2013, April). Secure and efficient range queries on outsourced databases using Rp-trees. In *2013 IEEE 29th international conference on data engineering* (pp. 314-325). IEEE. doi:10.1109/ICDE.2013.6544835

Wang, X., Hong, H., Zeng, J., Sun, Y., & Liu, G. (2022, December). EIMDC: A new model for designing digital twin applications. In *International conference on internet of things* (pp. 19-32). doi:10.1007/978-3-031-23582-5_2

Wong, W. K., Cheung, D. W. L., Kao, B., & Mamoulis, N. (2009, June). Secure kNN computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD international conference on management of data* (pp. 139-152). doi:10.1145/1559845.1559862

Wu, Z., Li, G., Shen, S., Lian, X., Chen, E., & Xu, G. (2021). Constructing dummy query sequences to protect location privacy and query privacy in location-based services. *World Wide Web (Bussum)*, *24*(1), 25–49. doi:10.1007/s11280-020-00830-x

Wu, Z., Li, R., Zhou, Z., Guo, J., Jiang, J., & Su, X. (2020). A user sensitive subject protection approach for book search service. *Journal of the Association for Information Science and Technology*, *71*(2), 183–195. doi:10.1002/asi.24227

Xiao, L., & Yen, I. L. (2012, March). Security analysis for order preserving encryption schemes. In *2012 46th annual conference on information sciences and systems* (pp. 1-6). IEEE. doi:10.1109/CISS.2012.6310814

Zeng, J., Yang, L. T., Lin, M., Ning, H., & Ma, J. (2020). A survey: Cyber-physical-social systems and their system-level design methodology. *Future Generation Computer Systems*, *105*, 1028–1042. doi:10.1016/j.future.2016.06.034

Zeng, J., Yang, L. T., Lin, M., Shao, Z., & Zhu, D. (2017). System-level design optimization for security-critical cyber-physical-social systems. *ACM Transactions on Embedded Computing Systems*, *16*(2), 1–21. doi:10.1145/2925991

Zhan, Y., Shen, D., Duan, P., Zhang, B., Hong, Z., & Wang, B. (2022). MDOPE: Efficient multi-dimensional data order preserving encryption scheme. *Information Sciences*, *595*, 334–343. doi:10.1016/j.ins.2022.03.001

Jinzhou Huang received the PhD degree in computer system architecture from Huazhong University of Science and Technology (HUST), in 2015. He is currently an associate professor in the School of Computer Engineering, Hubei University of Arts and Science, China. His research interests include online social networking, intelligent transportation, internet of things, computer networking and information security.