Fast Range Query on Encrypted Multi-dimensional Data in Cloud Environment

Zhuolin Mei[1], Jing Zeng[2], Caicai Zhang[3*], Shimao Yao[1,4], Jiaoli Shi[1,4], and Bin Wu[1,4]
[1] School of Computer and Big Data Science, Jiujiang University, China.
[2] China Gridcom Co., Ltd., China.
[3] School of Modern Information Technology, Zhejiang Polytechnic University of Mechanical and Electrical Engineering, China.
[4] Jiujiang Key Laboratory of Cyberspace and Information Security, China.
[e-mail: zhangcaicai@zime.edu.cn]
[*]Corresponding author: Caicai Zhang

Received March 10, 2024; revised July 5, 2024; accepted July 31, 2024; published September 30, 2024

Abstract

Cloud computing has extensively grown in recent years. A large amount of data is stored in cloud servers. To ensure confidentiality, these data is often encrypted and then stored in cloud servers. However, encryption makes range queries difficult to perform. To solve this issue, we present a scheme that facilitates fast range queries on encrypted multi-dimensional data in scenarios involving multiple users. In our scheme, we construct a tree index on encrypted multi-dimensional data, and each node is linked to a secure enhanced multi-dimensional range (MDR). To support efficient range query on the tree index, we adopt bloom filter technique. Additionally, users' privileges are designed in a one-way calculation manner to support that different users can only perform range queries within their own privileges. Finally, we conduct extensive experiments which show the efficiency of our scheme, and also conduct a thorough analysis of its security.

Keywords: Cloud Environment, Multi-dimensional data, Range Query, Data Privacy

1. Introduction

Cloud computing is a very attractive technology[1]. By leveraging cloud servers for outsourcing data and services, data owner and users can enjoy various advantages[2]. However, as the outsourced data stored on remote cloud servers is not directly under the control of data owners, data privacy emerges as a paramount concern[3, 4]. An effective approach to address this concern is to encrypt the data before outsourcing it. However, as ciphertexts are indistinguishable with each other, basic data manipulations become very difficult, such as queries[5].

Numerous novel encryption schemes have been proposed to facilitate range queries in recent years, but there still exist limitations. Order preserving encryption methods support efficient comparison between the ciphertexts of numerical data[6]. However, according to the ordering of ciphertexts, plaintexts can be precisely estimated. To preserve the ordering of ciphertexts, researchers have proposed bucketization methods[7, 8, 9, 10]. These methods involve dividing the data into multiple buckets, encrypting them in each bucket collectively. This ensures that the ordering of ciphertexts within a bucket remains secure. However, a limitation of these methods is that they do not support scenarios involving multiple users. To solve this issue, public-key based methods are proposed[11, 12, 13]. However, these public key-based methods inherently involve a considerable amount of intricate computations. Thus, the main shortage of these public key-based methods is inefficient. As the technique of bloom filter[14] can be utilized to efficiently judge whether a datum belongs to a set or not, many methods[15, 16, 17, 18] adopt bloom filter to achieve secure range query on ciphertexts. Many works[19, 20] focus on proposing frameworks and heightened security for range query over ciphertexts. Existing searchable keyword encryption schemes can be applied in these frameworks and then provide secure and fast range queries on ciphertexts. However, the above bloom filter based methods and the proposed frameworks for range queries on ciphertexts fail to account for the handling of multi-dimensional data.

In this paper, we propose a scheme which enables secure and fast range queries on encrypted multi-dimensional data in scenarios involving multiple users. In our scheme, we construct a tree index, and each node is linked to a secure enhanced multi-dimensional range (MDR). All the data covered by the MDR of each leaf node is encrypted by using a secure encryption scheme (e.g. AES) and stored under the leaf node. Then, the data owner enhances the security of the MDR of each node in the tree index by using one-way hash functions (e.g. SHA families). Next, the data owner generates a binary string for each security enhanced MDR by using bloom filter, and stores the binary string in the corresponding node. After all the MDRs associated with the nodes have been transformed to binary strings, the data owner deletes all the information in these nodes (except the binary strings and the links between nodes) to form a secure and efficient interval tree index (denoted by SEITI). Additionally, the data owner sets different privilege values for different nodes of SEITI and constructs a one-way calculation between these privilege values. Finally, the data owner outsources encrypted multi-dimensional data and SEITI to the cloud server, and distributes different privilege values to users. A user can use his/her privilege value to efficiently and securely perform range query on SEITI only within his/her privilege.

We summarize our scheme into the following three aspects: (1) We propose a tree index SEITI based on MDR encoding, one-way hash functions and bloom filter. (2) We propose an efficient and secure range query method over encrypted multi-dimensional data by using SEITI. (3) We conduct many experiments which demonstrate the efficiency of our scheme, and we also analyze its security.

2. Related Work

There are primarily four categories of secure and fast range query methods over encrypted data: order preserving encryption methods, bucketization methods, bloom filter based methods and searchable symmetric encryption methods.

In OPE, data is encrypted in an ordered manner, allowing for fast ciphertext retrieval[6]. Popa et al.[21] build a mutable OPE that achieves the ideal security of OPE. In this method, a binary tree is a state of the encryption which is maintained on a server. Intensive client-to-server interactions and mutations on previous ciphertexts are required when encrypting new numeric data. In [22], authors have improved mutable OPE by using random binary trees.

In bucketization methods, data is divided into different buckets. Each bucket is assigned an identity. Thus, the ordering of ciphertexts in a bucket can be well protected. When the queried range overlaps with a bucket, the ciphertexts in the bucket will be returned as part of the query results. Wang et al.[8] construct an index $\hat{R}-tree$. Each node of $\hat{R}-tree$ is linked to an encrypted minimum bounding rectangle (MBR). The adopted encryption scheme is asymmetric scalar-product preserving encryption (ASPE)[23]. Lee[9] proposes an ordered bucketization method. According to the ordering of buckets, query results can be efficiently found. However, Lee's method only considers the ciphertexts of single dimensional data. Given a range, ASPE can be used to judge whether the range intersects with an MBR. Thus, Wang's method allows a user to find the query results in a top-down manner by using $\hat{R}-tree$. Mei et al.[24] construct key derivation in their tree index to support multiple users with different search privileges.

Li et al.[15] propose a scheme called PBtree to perform secure range query. According to prefix encoding and bloom filter, the index in this scheme achieves index indistinguishability. In each node of the index, all the data are encoded and then form a binary string by using one-way hash function and bloom filter. When there is a large amount of data, the binary strings are very long, which incur a large space overhead of PBtree. In [16], Wang et al. propose a method which supports range query for arbitrary geometry over ciphertexts. Each data point and geometric range query are represented as binary strings of bloom filters. Whether a point lies within a geometric area is determined by the inner product of two binary strings. However, as the data or query scope increases, it becomes necessary to create large bloom filters that covers all potential points within a specific query range. This, in turn, results in increased storage and computational costs. Then, Wang et al.[26] propose an improved method based on [16]. However, as the query scope increases, it also requires exhausting all the data that may be searched during the entire query process. Cui et al.[17] propose a privacy preserving solution to support boolean range query on ciphertexts. They present a spatio-textual data encoding technique based on bloom filter and assess the effectiveness of bloom filters using ASPE[23]. Mahdikhani et al.[18] employs Paillier homomorphic cryptosystem and ingenious bloom filter data structure for simultaneously achieving better privacy and higher efficiency in the count aggregation in a privacy-preserving range query scenario.

Faber et al.[27] employ a binary tree structure to handle dataset. They leverage a static SSE to support range queries over ciphertexts. Demertzis et al.[28] also utilize a similar index as used in [27], and introduce several range query schemes. Each scheme offers a distinct trade-off between security and efficiency. Zuo et al.[19] introduce two schemes incorporating range-covering technique to facilitate range queries on ciphertexts. Additionally, their schemes offer flexible update functionality. Wang et al.[20] propose a scheme which supports range queries over dynamic database with forward/backward security. However, these schemes only consider the range query on the ciphertexts of single dimensional data.

3. System and Security Model

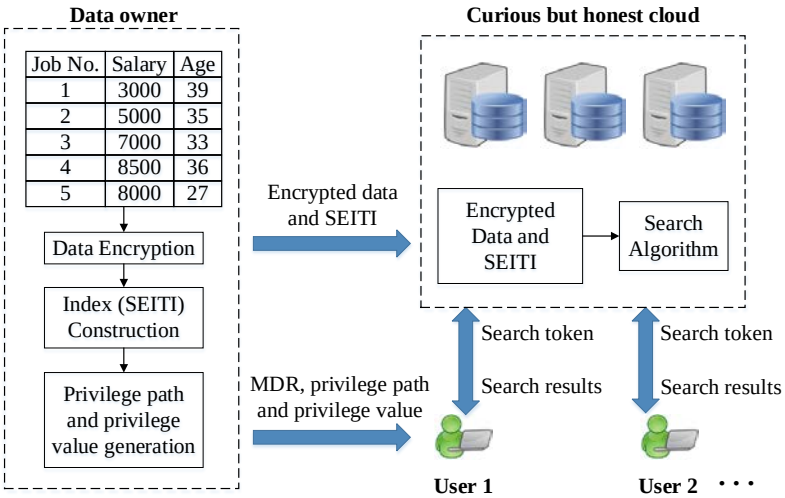

Fig. 1. Architecture of Our Scheme.

Fig. 1 shows the architecture of our scheme. Data owner (DO) outsources encrypted multi-dimensional data and a secure and efficient interval tree index (SEITI) to the cloud server (CS). Then, DO distributes privilege values, privilege paths and the information about MDRs to users (Us). If a user (U) maintains the privilege value and the privilege path of an MDR (denoted by MDR), he/she can generate the search token for any sub-MDR (denoted by MDR', $MDR' \subseteq MDR$). Then, U sends the search token of MDR' to CS. After receiving the search token, CS performs range query on SEITI and finds all the encrypted multi-dimensional data in MDR'. Finally, CS returns the search results to U.

In our scheme, we adopt the honest but curious model for CS. The term "honest" implies that CS (i) faithfully adheres to the designated protocols and procedures to fulfill its role as a service provider, and (ii) refrains from modifying the ciphertexts or secure index stored on it. On the other hand, the term "curious" indicates that CS may exhibit curiosity, either due to its own inherent curiosity or as a result of being compromised to act on behalf of a third party. In our scheme, the threat model relies on the information available to adversaries (such as CS), which is also used in [29, 30].

Definition 1 (Security [31]). Given a leakage function F, all adversaries cannot reveal more information except the leakage function F, then the range query scheme is secure. The leakage function F is defined as $F(m_i, m_j) = position_{diff}(m_i, m_j)$, where $position_{diff}(m_i, m_j)$ gives the different first position of m_i and m_j.

4. Construction of Secure and Efficient Interval Tree Index (SEITI)

In our scheme, DO first builds an interval tree over the outsourced data. Second, DO assigns multi-dimensional range (MDR) to each node of the interval tree. In each leaf node, DO uses a secure encryption algorithm (e.g. AES) to encrypt all the outsourced data covered by the MDR of the leaf node, and then stores these ciphertexts under the leaf node. Next, DO encodes each MDR into an MDR code, and then enhances the security of each MDR code to generate a secure and efficient interval tree index (SEITI). Finally, DO outsources all the ciphertexts

and SEITI to CS.

In the following paragraphs, we present the construction of SEITI. The construction of SEITI consists of three parts: Multi-Dimensional Range (MDR) assignment, MDR code generation and security enhancement of MDR code.

MDR Assignment. In our scheme, SEITI is a full n-ary tree, where n is an integer and calculated by division parameters (please refer to the next paragraph). The structure of SEITI is helpful for Us to calculate appropriate privilege values and then generate the corresponding search tokens for their queried ranges (please refer to Section 5 and 6). Each node of SEITI is associated with an MDR. In SEITI, the MDR of the root node covers all the outsourced data. For each internal node, its MDR is uniformly divided into n smaller MDRs by division parameters. These smaller MDRs are assigned to the child nodes of the internal node respectively. For each leaf node, all the outsourced data covered by its MDR are encrypted as a unit by using a secure encryption scheme, and stored under the leaf node.

For the i^{th} dimension, DO defines the division parameter div_i (div_i is a positive integer). According to div_i, a range $R_i = (a_i, b_i]$ on the i^{th} dimension can be uniformly divided into div_i smaller ranges, $(a_i, a_i + \Delta_i]$, $(a_i + \Delta_i, a_i + 2\Delta_i]$, … , $(a_i + (div_i - 1)\Delta_i, a_i + div_i\Delta_i]$, where $\Delta_i = (b_i - a_i) / div_i$. Similarly, an MDR $MDR = (a_0, b_0] \times (a_1, b_1] \times \ldots \times (a_{d-1}, b_{d-1}]$ can be uniformly divided into $n = \prod_{i=0}^{d-1} div_i$ smaller MDRs as follows.

$$\{(a_0 + o_0\Delta_0, b_0 + (o_0+1)\Delta_0] \times (a_1 + o_1\Delta_1, b_1 + (o_1+1)\Delta_1] \times \ldots \times (a_{d-1} + o_{d-1}\Delta_{d-1}, b_{d-1} + (o_{d-1}+1)\Delta_{d-1}] \mid i = 0,1,\ldots,d-1, o_i \in \{0,1,\ldots,div_i - 1\}, \Delta_i = (b_i - a_i)/div_i\}.$$

Suppose A is the root node of SEITI and associated with MDR_A. According to the division parameters, MDR_A is uniformly divided into $n = \prod_{i=0}^{d-1} div_i$ smaller MDRs and each smaller MDR is assigned to a child node of A. By using the method iteratively, each node of SEITI can be assigned to an MDR. Thus, if the MDR of a node in SEITI is known, all the MDRs of its descendant nodes can be obtained efficiently. Example 1 illustrates the MDR assignment for the nodes in SEITI.

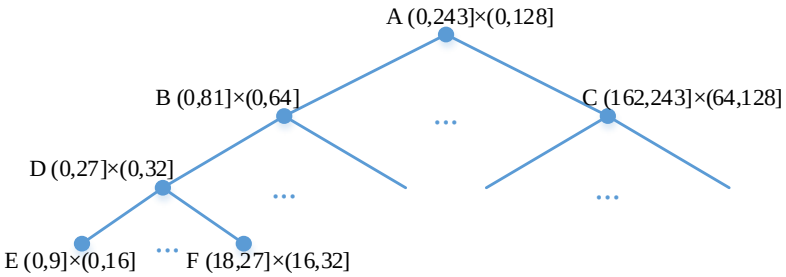

Fig. 2. MDR Assignment for the Nodes in SEITI.

Example 1. As shown in **Fig. 2**, the division parameters are $div_0 = 3$ and $div_1 = 2$. SEITI is a full 6-ary tree ($6 = div_0 \times div_1$). The height of SEITI is 4. The root node A is associated with $(0,243] \times (0,128]$. $(0,243] \times (0,128]$ is uniformly divided into smaller MDRs iteratively, and each smaller MDR is assigned to the descendant nodes of A.

MDR Code Generation. MDR code generation involves two steps, range encoding and MDR encoding.

Range Encoding. Suppose $(x_i, y_i]$ is a range which can be obtained by using div_i to divide $(a_i, b_i]$ iteratively and the length of range code is l. To encode the range $(x_i, y_i]$, one can employ the subsequent steps.

(1) DO builds a full div_i-ary tree (denoted by T_i). The height of T_i is $l+1$.

(2) DO assigns $(a_i, b_i]$ to the root node (denoted by A_i) of T_i. Then, DO sets the range code of $(a_i, b_i]$ to $**\ldots*$, where each $*$ can be any value in $0,1,\ldots,div_i-1$ (e.g. $**\ldots*$ can be $00\ldots0$, $00\ldots1$, … , $00\ldots(div_i-1)$, … , $(div_i-1)(div_i-1)\ldots(div_i-1)$, etc.).

(3) DO uniformly divides $(a_i, b_i]$ into div_i smaller ranges, which are $(a_i, a_i+\Delta_i]$, $(a_i+\Delta_i, a_i+2\Delta_i]$, … , $(a_i+(div_i-1)\Delta_i, a_i+div_i\Delta_i]$, where $\Delta_i=(b_i-a_i)/div_i$. Then, DO assigns these smaller ranges to the child nodes of A_i, and sets their range codes to $0*\ldots*$, $1*\ldots*$, … , $(div_i-1)*\ldots*$, respectively.

(4) By executing the above procedure iteratively, each node of T_i is associated with a smaller range and the range can be encoded to a range code. As the smaller range $(x_i, y_i]$ can be obtained by using div_i and $(a_i, b_i]$, some node in T_i is associated with $(x_i, y_i]$. Thus, the range code of $(x_i, y_i]$ can be easily obtained.

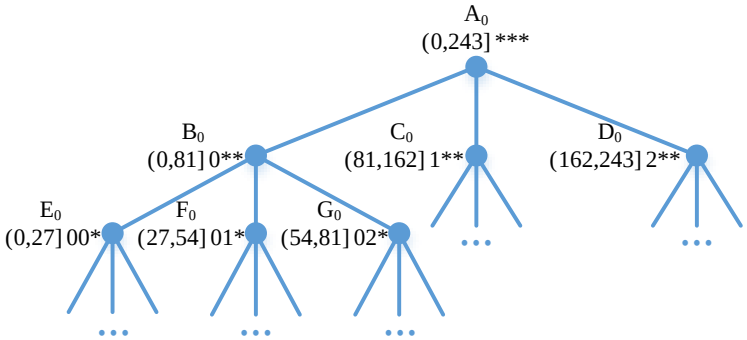

Fig. 3. Range Encoding in (0,243].

As shown in **Fig. 3**, T_0 is a tree over the range $(0,243]$. The division parameter is $div_0=3$. The length of range codes is set to $l=3$. The height of T_0 is set to 4 ($4=l+1$). The root node A_0 is associated with (0,243] and its range code is $***$, where each $*$ can be any value in 0, 1, 2 ($2=div_0-1$). (0,243] is uniformly divided into 3 ($div_0=3$) smaller ranges, (0,81], (81,162] and (162,243]. These smaller ranges are associated with the nodes B_0, C_0 and D_0, and their range codes are $0**$, $1**$ and $2**$ respectively. By executing the above procedure iteratively, the smaller range of each node in T_0 can be encoded, e.g. the smaller ranges (0,27], (27,54] and (54,81] are encoded to 00*, 01* and 02* respectively.

It is easy to find the relationship between a range and its range code. For simplicity, let $c_0c_1\ldots c_{l-1}$ be the range code of $(x_i, y_i]$, where $c_j = *, 0, 1, \ldots, div_i - 1$ ($j = 0, 1, \ldots, l-1$).

$$x_i = a_i + S_i \times (c_0 \times div_i^{l-1} + c_1 \times div_i^{l-2} + \ldots + c_{l-1} \times div_i^0) \quad (1)$$

$$y_i = a_i + S_i \times (c_0 \times div_i^{l-1} + c_1 \times div_i^{l-2} + \ldots + c_{l-1} \times div_i^0) + S_i \quad (2)$$

, where $(a_i, b_i]$ is the range associated with the root node of T_i and $S_i = (b_i - a_i) / div_i^l$.

According to the formula (1) and (2) above, one can calculate $(x_i, y_i]$ by using $c_0c_1\ldots c_{l-1}$. For $c_0c_1\ldots c_{l-1}$, if $c_j \neq *$ ($j = 0, 1, \ldots, l-1$), one takes c_j into the formula (1) and (2) to calculate x_i and y_i. Finally, one can obtain $(x_i, y_i]$. If $c_j = *$, (i) one takes $c_j = 0$ into the formula (1) to calculate x_i, which is the minimal value represented by $c_0c_1\ldots c_{l-1}$. (ii) One takes $c_j = div_i - 1$ into the formula (2) to calculate y_i, which is the maximal value represented by $c_0c_1\ldots c_{l-1}$. Finally, one can obtain the range $(x_i, y_i]$.

One can also calculate $c_0c_1\ldots c_{l-1}$ by using $(x_i, y_i]$. First, one calculates $\frac{x_i - a_i}{S_i} = c_0 \times div_i^{l-1} + c_1 \times div_i^{l-2} + \ldots + c_{l-1} \times div_i^0$ by using the formula (1). Then, one divides $\frac{x_i - a_i}{S_i}$ by the power of div_i to calculate the reminder iteratively. Finally, one can obtain the code of x_i. By using the same method, one can use the formula (2) to obtain the code y_i. According to the codes of x_i and y_i, one can obtain the range code $c_0c_1\ldots c_{l-1}$.

According to the analysis above, it is easy to know that (i) there is no need to build a full div_i-ary tree T_i, and (ii) by using the formular (1) and (2), the range code $c_0c_1\ldots c_{l-1}$ can be efficiently calculated when the range $(x_i, y_i]$, div_i and $(a_i, b_i]$ are known. (iii) By using the formular (1) and (2), the range $(x_i, y_i]$ can be efficiently calculated when the range code $c_0c_1\ldots c_{l-1}$, div_i and $(a_i, b_i]$ are known.

MDR Encoding. To encode the MDR $MDR = (a_0, b_0] \times (a_1, b_1] \times \ldots \times (a_{d-1}, b_{d-1}]$, one can employ the subsequent steps.

(1) DO calculates the range codes for $R_0 = (a_0, b_0]$, $R_1 = (a_1, b_1]$, …, $R_d = (a_d, b_d]$. The range codes are C_{R_0}, C_{R_1}, …, $C_{R_{d-1}}$ respectively. DO sets a privilege value pv for MDR (please refer to the next section).

(2) Suppose (i) the range code C_{R_i} is $c_0^i c_1^i \ldots c_{l-1}^i$ (the length of C_{R_i} is l), where c_j^i is the j^{th} value of C_{R_i}, and (ii) the MDR code C_{MDR} is $v_0v_1\ldots v_{l-1}$, where v_j is the j^{th} value of C_{MDR}. Then, DO calculates $k_j = H(j, pv) \bmod d$, where H is a one-way hash function (e.g. SHA families). According to k_j, DO chooses $C_{R_{k_j}}$ and sets v_j to the j^{th} value $c_j^{k_j}$ of $C_{R_{k_j}}$, i.e. $v_j = c_j^{k_j}$. Finally, after all the values v_0, v_1, …, v_{l-1} are determined, the MDR code $C_{MDR} = v_0v_1\ldots v_{l-1}$ is obtained.

In our scheme, the MDR code C_{MDR} can be seen as randomly calculated, because (i) the privilege value pv is calculated by using a random number and a one-way hash function (please refer to the next section), (ii) k_j is calculated by using $H(j, pv) \bmod d$ (thus k_j can be seen as calculated by using the pseudo-random number pv), and (iii) v_j is set to the j^{th} value $c_j^{k_j}$ of $C_{R_{k_j}}$, where $j = 0,1,...,l-1$. Additionally, as the one-way hash function H is used in the procedure of MDR encoding, the privilege value pv is very safe even if the MDR code C_{MDR} is leaked.

Security Enhancement for MDR Code. For security concern, the MDR code C_{MDR} should be protected. This is because the values in C_{MDR} are chosen from the range codes C_{R_0}, C_{R_1}, …, $C_{R_{d-1}}$ and the number of values in C_{MDR} is not many. Attackers can guess the MDR code C_{MDR}. Thus, DO should enhance the security of C_{MDR}.

(1) DO enhances the security of each value v_j ($j = 0,1,\ldots,l-1$) in C_{MDR}. As (i) v_j is chosen from the range code $C_{R_{k_j}}$ and (ii) the values in $C_{R_{k_j}}$ are chosen from $*,0,1,\ldots,div_{k_j}-1$ (please refer to Section 4), v_j is one of the values $*,0,1,\ldots,div_{k_j}-1$. Thus, (i) if $v_j \neq *$, DO calculates $H(j,H(v_j, pv))$. (ii) If $v_j = *$, DO calculates $H(j,H(0, pv))$, $H(j,H(1, pv))$, … , $H(j,H(div_{k_j}-1, pv))$, because $*$ can be any value from $0,1,\ldots,div_{k_j}-1$ (please refer to Section 4).

Note that, as H is a one-way hash function, all the information about C_{MDR} (including the position j, the value v_j and the privilege value pv) can be safely protected even if attackers know the value $H(j,H(v_j, pv))$ ($j = 0,1,\ldots,l-1$).

(2) To support range query, DO uses bloom filter (denoted by BF) to handle $H(j,H(v_j, pv))$. If $v_j \neq *$, DO calculates $v_j' = BF(H(j,H(c_j, pv)))$. If $c_j = *$, as $*$ can be any value from $0,1,\ldots,div_{k_j}-1$, DO calculates $v_j' = BF(H(j,H(0, pv)))\,|\,BF(H(j,H(1, pv)))\,|\ldots|\,BF(H(j,H(div_{k_j}-1, pv)))$, where $|$ denotes bitwise OR operation.

(3) Finally, DO calculates the security enhanced MDR code of C_{MDR}, which is $[C_{MDR}] = v_0'\,|\,v_1'\,|\ldots|\,v_{l-1}'$.

In summary, first, DO builds a full n-ary tree and each node is associated with an MDR in the form of plaintext. Second, DO encodes each MDR to an MDR code and then enhances the security of the MDR code. Then, DO replaces the MDR in each node with its corresponding security enhanced form, which is as the SEITI in our scheme. Finally, DO outsources SEITI along with the encrypted multi-dimensional data to CS. Note that, different privilege values are used to enhance the security of different MDR codes (please refer to Section 5).

5. Privilege Assignment and Revocation

In this section, we first introduce privilege value. Privilege values is used to enhance the security of an MDR and describes the privilege of U. If U maintains the privilege value of a node in SEITI, he/she can perform range queries within the smaller MDRs covered by the MDR of the node. Then, we introduce privilege path, which can help CS to efficiently find the MDR which U has the privilege to search. Finally, we present the privilege assignment and revocation.

Privilege Value. For the sake of clarification, suppose A is a node in SEITI, the MDR of A is MDR_A, the privilege value of A is pv_A, B is a child node of A, the MDR of B is MDR_B and the privilege value of B is pv_B.

(1) DO chooses a hash function H_{ID} (which is public) to calculated the identities of nodes in SEITI, e.g., the identity of A can be calculated $ID_A = H_{ID}(MDR_A)$. As the MDRs of nodes in SEITI are different, their identities are different. If one knows the MDR of a node, he/she can efficiently calculate the smaller MDRs of its descendant nodes (please refer to Section 4). Thus, if one knows MDR_A, he/she can calculate MDR_B and then calculate the identity of B, which is $ID_B = H_{ID}(MDR_B)$.

(2) DO randomly chooses an integer pv as the privilege value of the root node in SEITI. For an internal node, its privilege value is calculated by its parent node. Specifically, if one knows MDR_A and pv_A, he/she calculates MDR_B and $ID_B = H_{ID}(MDR_B)$, and then calculates $pv_B = H(ID_B, pv_A)$, where H is a one-way hash function. By using the same method, if one knows MDR_A and pv_A, he/she can calculate the MDR and the privilege value of any descendent node of A. Thus, given the privilege value pv and the MDR of the root node in SEITI, DO can set the privilege value for any node in SEITI.

If DO distributes MDR_A and pv_A to U, as U can calculate MDR_C ($MDR_C \subseteq MDR_A$) and pv_C of C (C is a descendant node of A), U can search the ciphertexts covered by MDR_C (please refer to Section 7). As H is a one-way hash function, U cannot calculate the MDRs and the privilege values of the nodes which are not the descendant nodes of A, which guarantees that U can only perform range queries within his/her privilege.

Privilege Path. The privilege path of A can help CS securely and efficiently find A in SEITI.

In SEITI, there exists a path from the root node to A. The path is denoted by $\langle N_0, N_1, \ldots, N_{m-1} \rangle$, where N_0 denotes the root node, N_{m-1} denotes the parent node of A and N_i is the parent node of N_{i+1} ($0 \le i < i+1 \le m-1$). For the nodes on the path, their security enhanced MDR codes can be obtained from SEITI, which is $\langle [C_{MDR_{N_0}}], [C_{MDR_{N_1}}], \ldots, [C_{MDR_{N_{m-1}}}] \rangle$. DO randomly generates $m-1$ binary strings $bs_{N_0}, bs_{N_1}, \ldots, bs_{N_{m-1}}$, s.t. $bs_{N_0} \& [C_{MDR_{N_0}}] = bs_{N_0}$, $bs_{N_1} \& [C_{MDR_{N_1}}] = bs_{N_1}$, …, $bs_{N_{m-1}} \& [C_{MDR_{N_{m-1}}}] = bs_{N_{m-1}}$, where & denotes bitwise AND operation. Finally, DO

generates $PP_A = \langle bs_{N_0}, bs_{N_1}, \ldots, bs_{N_{m-1}} \rangle$, which is the privilege path of A.

Privilege Assignment. DO distributes $\langle (PP_A, ID_u)_{sig_{DO}}, PP_A, MDR_A, pv_A \rangle$ to a user u (it means u has the privilege to search the ciphertexts in MDR_A), where $(PP_A, ID_u)_{sig_{DO}}$ is the signature of DO. The signature is used to verify the legality of u's privilege when he/she submits query requests to CS.

Privilege Revocation. To revoke the privilege of u, DO should update SEITI in CS and assign new secret parameters to the other Us. First, DO chooses a random number for the root node of SEITI, and then calculates the pseudo-random number for each internal node. Specifically, if r_A is the random number of the node A, the random number r_B of the node B (A is the parent of B) can be calculated, $r_B = H(ID_B, r_A)$, where H is a one-way hash function and ID_B is the identity of B. Thus, all the random numbers of the nodes in SEITI can be calculated iteratively. For each node in SEITI, if the random number is r and the security enhanced MDR code is $[C_{MDR}]$, DO calculates a binary string $s_r = BF(r)$, and then calculates $s_r \& [C_{MDR}]$. Next, DO replaces $[C_{MDR}]$ with $s_r \& [C_{MDR}]$ to update SEITI. Finally, if the privilege of u (suppose u holds $\langle (PP_A, ID_u)_{sig_{DO}}, PP_A, MDR_A, pv_A \rangle$) is not revoked, DO generates a new privilege path PP_A*, and sends $\langle (PP_A*, ID_u)_{sig_{DO}}, PP_A*, MDR_A, pv_A, r_A \rangle$ to u. Hence, Us whose privileges have been revoked are unable to continue performing queries on SEITI, while Us whose privileges have not been revoked can still carry out queries on SEITI.

6. Search Token Generation

After receiving $\langle (PP_A, ID_u)_{sig_{DO}}, PP_A, MDR_A, pv_A \rangle$, u can generate the search token for the queried multi-dimensional range MDR_Q by using the subsequent steps.

(1) u calculates $MDR_A \cap MDR_Q$. If $MDR_A \cap MDR_Q = \varnothing$, u has no privilege to search any ciphertexts in MDR_Q. Otherwise, u generates the search token for $MDR_A \cap MDR_Q$.

(2) u builds an n-ary tree T_{SRCH}, where $n = \prod_{i=0}^{d-1} div_i$. The height of T_{SRCH} is $l+1-m$. There is sub-tree T_A in SEITI. The root node of T_A is associated with MDR_A. The height of T_A is $l+1-m$. It is easy to know that T_{SRCH} and T_A have the same structure.

(3) u assigns MDR_A to the root node of T_{SRCH} and sets pv_A as its privilege value. Then, for the other nodes of T_{SRCH}, u iteratively calculates the smaller MDRs and privilege values by using division parameters and one-way hash functions (u can utilize similar method as DO used in Section 4 and Section 5).

(4) According to the MDRs associated with nodes in T_{SRCH}, u calculates the minimal cover of $MDR_A \cap MDR_Q$, which is denoted by $MC_{MDR_A \cap MDR_Q}$ (please refer to the following Definition 2). Then, u prunes T_{SRCH}. For each node N in T_{SRCH}, if $N \notin MC_{MDR_A \cap MDR_Q}$ or N is not the ancestor node of any node in $MC_{MDR_A \cap MDR_Q}$, u deletes N and its corresponding links in T_{SRCH}. The pruned T_{SRCH} is denoted by T_{SRCH}'. It is obvious that the set of all the leaf nodes in T_{SRCH}' equals to $MC_{MDR_A \cap MDR_Q}$.

Definition 2 (Minimal Cover). Minimal Cover is a set of some nodes in a tree. Suppose A is the root node of the tree, MDR_A is the multi-dimensional range associated with A, MDR is a multi-dimensional range ($MDR \subseteq MDR_A$). MC_{MDR} is the minimal cover of MDR iff (i) $MDR \subseteq \bigcup_{N \in MC_{MDR}} MDR_N$, where N is a node of the tree and MDR_N is associated with N; (ii) $\forall MC \in \{MC \mid MC \subseteq MC_{MDR}\}$, $\bigcup_{N \in MC} MDR_N \subseteq MDR$.

In the following paragraphs, we present the steps for u to handle T_{SRCH}'. For ease of explanation, we suppose N is a node in T_{SRCH}', its MDR is $MDR_N = (a_0, b_0] \times (a_1, b_1] \times \ldots \times (a_{d-1}, b_{d-1}]$, its privilege value is pv_N and its identity is $ID_N = H_{ID}(MDR_N)$.

(5) For each node N in T_{SRCH}', u randomly generates a d-dimensional data $data_N = (r_0, r_1, \ldots, r_{d-1})$ in MDR_N. According to the formula $\frac{x_i - a_i}{S_i} = c_0 \times div_i^{l-1} + c_1 \times div_i^{l-2} + \ldots + c_{l-1} \times div_i^0$ (please refer to Section 4), u sets $x_i = r_i$, and divides $\frac{r_i - a_i}{S_i}$ by the power of div_i. Finally, u iteratively calculates the reminder to obtain the code of r_i, which is denoted by $c_0^i c_1^i \ldots c_{l-1}^i$ ($i = 0,1,\ldots,d-1$).

(6) u chooses some values from these codes $c_0^i c_1^i \ldots c_{l-1}^i$ ($i = 0,1,\ldots,d-1$) to encode $data_N$, denoted by $C_{data_N} = t_0 t_1 \ldots t_{l-1}$. First, u calculates $k_i = H(i, pv_N) \bmod d$. Then, u chooses the code $r_{k_i} = c_0^{k_i} c_1^{k_i} \ldots c_{l-1}^{k_i}$. Next, u extracts the ith value in $r_{k_i} = c_0^{k_i} c_1^{k_i} \ldots c_{l-1}^{k_i}$ and sets t_i to $c_i^{k_i}$. Finally, u calculates all the values in $C_{data_N} = t_0 t_1 \ldots t_{l-1}$, where $t_i = c_i^{k_i}$ and $i = 0,1,\ldots,d-1$.

Note that, the encoding procedures of a data and an MDR are very similar, but they have two differences: (i) the MDR is encoded by DO to construct SEITI, but the data is encoded by Us to generate search token; (ii) There may exist the symbol * in the code of the MDR, but there is no * in the code of the data, i.e. $t_i \neq *$ ($i = 0,1,\ldots,d-1$).

(7) u enhances the security of $C_{data_N} = t_0 t_1 \ldots t_{l-1}$. For the value t_i ($t_i \neq *$), u calculates $t_i' = BF(H(i, H(t_i, pv_N)))$, where H is the one-way hash function and BF is the bloom filter. Finally, u obtains the security enhanced C_{data_N}, which is $\left[C_{data_N}\right] = t_0' \mid t_1' \mid \ldots \mid t_{l-1}'$ (|

denotes the bitwise OR operation).

(8) For each node N in T_{SRCH}', u deletes all the information (including ID_N, MDR_N, pv_N and C_{data_N} etc.), retaining only $\left[C_{data_N}\right]$. Then, u obtains the search token of $MDR_A \cap MDR_Q$, denoted by $\left\langle (PP_A, ID_u)_{sig_{DO}}, (T_{SRCH}', ID_u)_{sig_u}, PP_A, T_{SRCH}' \right\rangle$, where $(T_{SRCH}'*, ID_u)_{sig_u}$ is the signature of u. Upon receiving $\left\langle (PP_A, ID_u)_{sig_{DO}}, (T_{SRCH}', ID_u)_{sig_u}, PP_A, T_{SRCH}' \right\rangle$, CS can verify the legality of the privilege of u through $(PP_A, ID_u)_{sig_{DO}}$, and verify the legality of the query token through $(T_{SRCH}', ID_u)_{sig_u}$.

Note that, if the privileges of some Us have been revoked, DO should update the privileges of the rest Us. Suppose that after the update, u holds $\left\langle (PP_A*, ID_u)_{sig_{DO}}, PP_A*, MDR_A, pv_A, r_A \right\rangle$. To generate a search token, u firstly generates T_{SRCH}'. Second, for each node N of T_{SRCH}', u calculates r_N by using r_A (please refer to Section 5) and calculates the binary string $s_{r_N} = BF(r_N)$. Then, u extracts $\left[C_{data_N}\right]$ in N and calculates s_{r_N} &$\left[C_{data_N}\right]$. Next, for each node N, u replaces s_{r_N} &$\left[C_{data_N}\right]$ with $\left[C_{data_N}\right]$ (the updated T_{SRCH}' is denoted as $T_{SRCH}'*$). Finally, u obtains the search token of $MDR_A \cap MDR_Q$, which is $\left\langle (PP_A*, ID_u)_{sig_{DO}}, (T_{SRCH}'*, ID_u)_{sig_u}, PP_A*, T_{SRCH}'* \right\rangle$.

7. Range query and Data Update

Privilege Assignment. To search the ciphertexts in $MDR_A \cap MDR_Q$, u sends $\left\langle (PP_A, ID_u)_{sig_{DO}}, (T_{SRCH}', ID_u)_{sig_u}, PP_A, T_{SRCH}' \right\rangle$ to CS. According to PP_A, CS can find the node A in SEITI. Then, by using T_{SRCH}', CS can find all the ciphertexts in $MDR_A \cap MDR_Q$. The details of range query are as follows.

(1) As $PP_A = \left\langle bs_{N_0}, bs_{N_1}, \ldots, bs_{N_{m-1}} \right\rangle$, CS extracts bs_{N_0} from PP_A. Then CS extracts $\left[C_{MDR_{M_0}}\right]$ from the root node of SEITI, where $\left[C_{MDR_{M_0}}\right]$ is the secure enhanced MDR code of M_0 (suppose the root node of SEITI is M_0). Then, CS tests whether bs_{N_0} &$\left[C_{MDR_{M_0}}\right] = bs_{N_0}$, where & denotes bitwise AND operation. If bs_{N_0} &$\left[C_{MDR_{M_0}}\right] = bs_{N_0}$, CS continues to test bs_{N_1} and $\left[C_{MDR_{M_1}}\right]$, where $\left[C_{MDR_{M_1}}\right]$ is the secure enhanced MDR code of M_1 (suppose M_1 is a child node of M_0). By using the method iteratively, if PP_A is legal, CS can find the node A in SEITI by using PP_A.

(2) CS tests the child nodes of A with the root node of T_{SRCH}' in turn. Suppose B is a child node of A and B' is the root node of T_{SRCH}', CS tests whether $\left[C_{data_{B'}}\right]\&\left[C_{MDR_B}\right]=\left[C_{data_{B'}}\right]$. If the equation holds, it means that the node B in SEITI corresponds to the node B' in T_{SRCH}'. Then, CS tests each child node of B with each child node of B' by using the above method iteratively. Finally, CS can find all the nodes in SEITI which correspond to the leaf nodes in T_{SRCH}'. As the set of all the leaf nodes in T_{SRCH}' is the minimal cover of $MDR_A \cap MDR_Q$ (please refer to Section 6), these corresponding nodes in SEITI cover all the ciphertexts covered by $MDR_A \cap MDR_Q$. Finally, CS returns all the ciphertexts under these corresponding nodes in SEITI to u as the search results.

Note that, if the distribution of the outsourced data is approximately uniform, our scheme can effectively handle the range query. In light of security considerations, DO can employ data padding method or data compression techniques to ensure that the sizes of encrypted multi-dimensional data under leaf nodes are the same. Namely, the leaf nodes of SEITI cannot be distinguished according to the volumes of ciphertexts stored under them.

Data Update. If a user u wants to update some data indexed by SEITI, he/she should first obtain the update authorization from DO. Then, u generates a search token to find the leaf nodes of SEITI which covers the data that is needed to be updated. With the assistance of CS, the data that is needed to be updated is replaced with the new data.

8. Experiment

In our experiments, we conduct a comparative analysis of our scheme with [24] and [31]. In [24], DO builds an Efficient Interval Tree (EIT) as index, which is constructed based on Asymmetric Scalar-product Preserving Encryption with Noise (ASPEN[32]). ASPEN consists of many matrix calculations, which are implemented by using Jama Library version 1.0.3. To ensure the security of ASPEN, the dimensionality of matrixes is extended to 100 by using extra artificial dimensions[23, 24, 32]. In [31], DO proposes a multi-dimensional data order preserving encryption method (MDOPE). MDOPE utilizes a network data structure to efficiently organize multi-dimensional data. It employs prefix encoding and bloom filter techniques to process the values stored in the network data structure, thereby enabling queries on encrypted multi-dimensional data.

Our experiments are conducted on a Windows 10 computer with an AMD Ryzen 5 2500U CPU and 8GB of RAM. We generate a collection consisting of 2.5×10^5 data entries. These data entries are evenly distributed within the 2-dimensional range $(0,2^{10}]\times(0,3^{10}]$. Over the range, we build EIT, MDOPE and SEITI separately. In EIT and SEITI, the division parameters for the first and second dimensions are set to 2 and 3 respectively. Thus, the fan-outs of EIT and SEITI are 6 ($6=2\times3$). In MDOPE, each node on the first dimension contains only one split data, while on the second dimension, each node contains two split data. As a result, the ranges on the first and second dimensions are divided into two and three smaller ranges respectively. Consequently, the fan-out of MDOPE is 5.

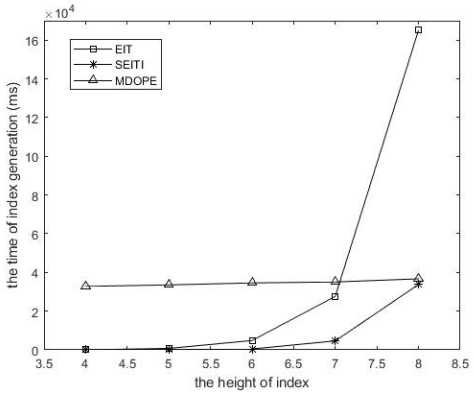

Fig. 4. Index Construction.

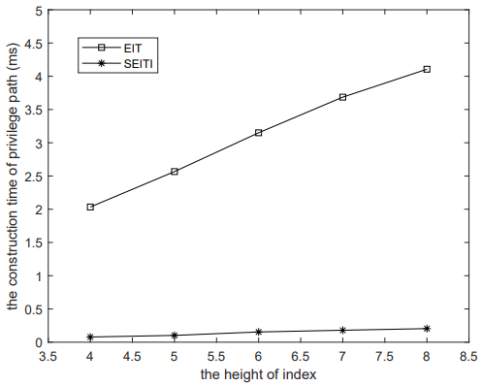

Fig. 5. Privilege Path Generation.

Index Construction. Fig. 4 shows the construction times of EIT and SEITI which increase exponentially. The construction time of MDOPE increase linearly. As EIT (and SEITI) is tree structure, the total number nodes in EIT (and SEITI) increases exponentially with the height of EIT (and SEITI). Thus, the construction time of EIT (and SEITI) increases exponentially. In EIT, the MDR of each node is encrypted by using a matrix. In SEITI, the MDR of each node is processed by one-way hash function and bloom filter. As one-way hash function and bloom filter are more efficient than matrix calculation, the index construction of SEITI is more efficient than that of EIT. In MDOPE, the split data in each index node only needs to be converted to a binary code, then padded with a random binary code, and finally processed by a bloom filter. Thus, the calculation is very efficient. The primary time overhead lies in inserting the encrypted multi-dimensional data into the index. Thus, when the height of the index increases (the amount of encrypted multi-dimensional data is fixed), the time of index construction grows slowly.

Privilege Path Generation. As shown in Fig. 5, the generations of privilege paths in EIT and SEITI increase linearly and are very efficient. Compared with EIT, the privilege path generation in SEITI is more efficient. When the height of EIT (and SEITI) increases, the average length of these privilege paths increases linearly. Thus, the average time for generating these privilege paths also increases linearly. In EIT, all the values in a privilege path should be encrypted by using matrices. In SEITI, all the values in a privilege path are randomly chosen according to the secure enhanced MDR codes stored in the nodes of SEITI. Therefore, due to the efficiency of the underlying calculation method, the privilege path generation in SEITI is more efficient than in EIT. Compared with EIT and SEITI, as MDOPE does not consider the scenario of multiple users, there is no privilege path generation in MDOPE.

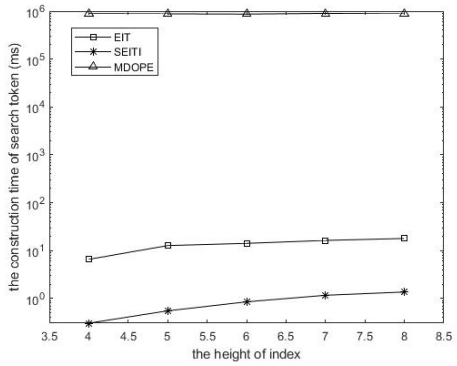

Fig. 6. Search Token Generation.

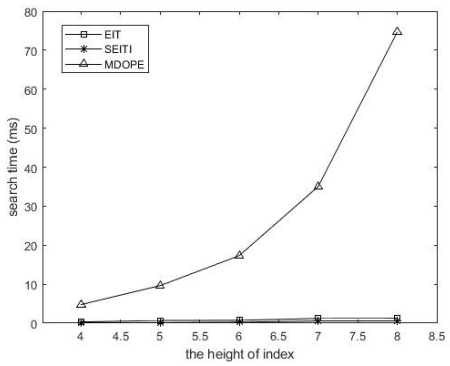

Fig. 7. Range query.

Search Token Generation. Fig. 6 shows the search token generation in MDOPE which is very inefficient. The average time of search token generation increases with the height of EIT (and SEITI). Compared with MDOPE and EIT, the search token generation in SEITI is more efficient. In EIT (and SEITI), the search token is constructed according to the minimal cover of a queried range. When the number of elements in the minimal cover increases, a user should take more time to process these elements. Thus, it takes more time to generate the search token for a queried range. In the experiments, we randomly generate thousands of search tokens for queried ranges. When the height of EIT (and SEITI) increases, the average size of these queried ranges increases. This results that the average number of elements in the minimal covers of the queried ranges increases. Thus, the average time of search token generation increases with the height of EIT (and SEITI). In SEITI, each value in the search token is encrypted by a matrix. In EIT, each value in the search token is processed by using one-way hash function and bloom filter. As one-way hash function and bloom filter are more efficient than matrix calculations, the search token generation in SEITI is more efficient than EIT. In MDOPE, a queried range is first transformed into a l-length bit string. Then, the l-length bit string is padded by a 2-length bit string for comparison purpose, and a l-length random bit string for security purpose. Thus, the queried range is transformed into a $(2l+2)$-length bit string. Next, the $(2l+2)$-length bit string is processed by using prefix encoding. This process is very inefficient, because the $(2l+2)$-length bit string implies that there are 2^{2l+2} different bit strings which should be merged one by one to finally form only a few bit strings. Thus, the search token generation in MDOPE is very inefficient.

Range query. The SEITI scheme utilizes a secure index to achieve efficient querying. In SEITI, the secure index is a tree structure where leaf nodes are associated with MDRs. The data within an MDR is encrypted as a unit. As the total amount of data increases, the amount of data in MDRs associated with leaf nodes also increases. During the querying process, the search token is compared only with the nodes of the secure index. Ultimately, the MDRs associated with the leaf nodes that intersect the query range are located, and the encrypted data within them is returned as the query result. Consequently, when the height of the secure index is fixed, an increase in the total amount of data does not affect the query efficiency of SEITI. Therefore, in our experiments, we keep the total amount of data unchanged and test the query efficiency by varying the height of the secure index. Fig. 7 shows the range query in SEITI which is more efficient than EIT and MDOPE. In EIT (and SEITI), a search token consists of a privilege path and a search tree. CS should find some nodes of EIT (and SEITI) which correspond to the nodes in the privilege path and the search tree according to the values stored in these nodes. Thus, if there are more nodes in a search token, it requires more calculations for CS to find the corresponding nodes in EIT (and SEITI). When the height of EIT (and SEITI) increases, the average amount of nodes in the search tokens increases, resulting that the average time of range query in EIT (and SEITI) increases. As (i) EIT and SEITI have similar tree structure and (ii) the search tokens in EIT and SEITI are both constructed by using the minimal covers of queried ranges, the average number of the nodes in search tokens are nearly the same. However, CS needs to execute multiple matrix calculations in EIT. In SEITI, CS only needs to execute bitwise AND calculations. Thus, the range query of SEITI is more efficient than that of EIT. In MDOPE, CS is unable to accurately determine whether an index node satisfies a search token. Therefore, CS may be required to tests the search token with the major split values in an index node and also the major child nodes of the index node. Therefore, if the height of index increases, the number of these split values and index nodes (that need to be tested) increases exponentially. Thus, the time of range query in MDOPE increases

exponentially with the height of index. In summary, the range query in SEITI is more efficient than EIT and MDOPE.

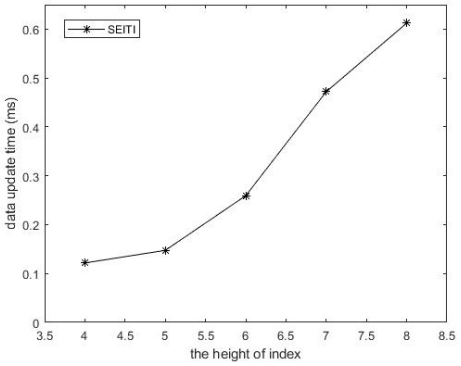

Fig. 8. Data update.

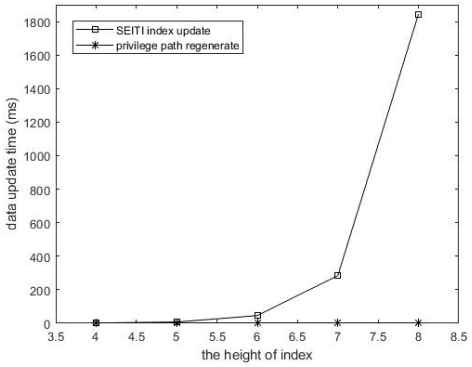

Fig. 9. Privilege Revocation.

Data update and privilege revocation. We only test the data update and privilege revocation in our scheme because both the EIT scheme and the MDOPE scheme do not provide data update and privilege revocation. SEITI supports efficient queries, which enables the cloud server to fast locate the data which should be updated and then replace the data with the new data. Thus, the efficiency of data update is related to its query efficiency. Consequently, as shown in **Fig. 8**, the efficiency of data update is also very efficient. To revoke the privileges of users, CS should update the index of SEITI, then regenerate and distribute new privilege paths to Us whose privileges have not been revoked. Thus, we test the efficiency of index update in SEITI and privilege path regeneration. As shown in **Fig. 9**, because the total number of index nodes increases exponentially with the index height, the time for updating the index increases exponentially. In SEITI, the computation overhead for regenerating a privilege path is very low. Thus, the regeneration of privilege path is very efficient. In our experiments, we only give the average time for regenerating a single privilege path. However, in practical applications, if the privileges of some users are revoked, DO needs to regenerate new privilege paths for the remaining users. Therefore, in multi-user scenarios, the time required for regenerating privilege paths depends on the total number of users whose privileges have not been revoked.

9. Security Analysis of SEITI

In this section, we first analyze the security of the index and the ciphertexts of multi-dimensional data stored in CS. Then, we proceed to analyze the security of the range query.

Theorem 1. The proposed SEITI scheme is secure in terms of the leak function F .

Proof 1. In SEITI, the multi-dimensional data is encrypted with a secure encryption scheme. Thus, the security of all the multi-dimensional data is guaranteed by the security of the encryption scheme. Given an MDR of an index node and a data of a search tree node (in the search token), their encoding lengths are both l (please refer to Section 4 and 6). The MDR code and the data code are then processed by a one-way hash function (e.g. SHA families) with privilege values (privilege values are pseudo-random numbers). The numbers of the output elements from the one-way hash function are the same. Then, these elements are as the inputs of the bloom filter. Suppose two inputs of the bloom filter are $a =< a_1, a_2, \ldots, a_n >$ and

$b =< b_1, b_2, \ldots, b_n >$ respectively and there are m elements in a and b are the same, it can deduce that $a_1 = b_1$, $a_2 = b_2$, …, $a_m = b_m$, $a_{m+1} \neq b_{m+1}$, …, $a_n \neq b_n$. According to the outputs of the bloom filter, an adversary only knows the leakage function $F(m_i, m_j) = position_{diff}(m_i, m_j)$. In summary, our scheme SEITI is secure with respect to the leakage function F.

Known Ciphertext Model: Adversaries can only access the ciphertexts of multi-dimensional data, the index SEITI, the submitted search tokens and the retrieved ciphertexts. According to the known ciphertext model, if an adversary records the search tokens and the retrieved ciphertexts, the adversary can obtain access patterns. Therefore, nothing beyond the access pattern should be leaked in the known ciphertext model. We adapt the definitions in [29, 30] for our proofs.

Definition 3. Search Pattern (SP): Let $Q = \{R_1, \ldots, R_m\}$ be the set of m consecutive queried ranges, SP be a $m \times m$ binary matrix s.t. if $MC_{R_i} = MC_{R_j}$, then $SP[i, j] = 1$. Otherwise, $SP[i, j] = 0$ ($i, j = 1, \ldots, m$, MC_{R_i} and MC_{R_j} are the minimal covers of R_i and R_j respectively).

Definition 4. Access Pattern (AP). Let $Q = \{R_1, \ldots, R_m\}$ be the set of m consecutive queried ranges, $T = \{T_{R_1}, \ldots, T_{R_m}\}$ be the search tokens for $Q = \{R_1, \ldots, R_m\}$, $C = \{C_1, \ldots, C_m\}$ be the retrieved ciphertext sets for $T = \{T_{R_1}, \ldots, T_{R_m}\}$. The access pattern for Q is defined as $AP = \{AP(T_{R_1}) = C_1, \ldots, AP(T_{R_m}) = C_m\}$.

Definition 5. History (H_m). Let $Q = \{R_1, \ldots, R_m\}$ be the set of m consecutive queried ranges and $C = \{C_1, \ldots, C_m\}$ be the retrieved ciphertext sets for $Q = \{R_1, \ldots, R_m\}$. Then, $H_m = (Q, C)$ is defined as a m-query history.

Definition 6. Trace (γ): Let $A_p(H_m)$ be the access pattern of H_m, $T = \{T_{R_1}, \ldots, T_{R_m}\}$ be the search tokens for $Q = \{R_1, \ldots, R_m\}$, $C = \{C_1, \ldots, C_m\}$ be the retrieved ciphertext sets for $Q = \{R_1, \ldots, R_m\}$. Then, the trace of H_m is defined as $\gamma(H_m) = \{AP(H_m), \{T_{R_1}, \ldots, T_{R_m}\}, \{C_1, \ldots, C_m\}\}$.

Definition 7. View (V): Let $SEITI$ be the index, $\mathbb{C}$ be all the ciphertexts under $SEITI$, $T = \{T_{R_1}, \ldots, T_{R_m}\}$ be the search tokens for $Q = \{R_1, \ldots, R_m\}$, $C = \{C_1, \ldots, C_m\}$ be the retrieved ciphertext sets for $Q = \{R_1, \ldots, R_m\}$. Then $V(H_m) = \{\mathbb{C}, SEITI, \{T_{R_1}, \ldots, T_{R_m}\}, \{C_1, \ldots, C_m\}\}$ is defined as the view of H_m. $V(H_m)$ is the information that is accessible to an adversary.

We employ a simulator-based proof method similar to the one widely used in [29, 30]. In essence, if the adversary cannot differentiate between two histories with the same trace, it implies that the adversary cannot gain any additional information about the ciphertexts, the index, the search tokens and the query results[29, 30].

Theorem 2. Our scheme is secure in the known ciphertext model.

Proof 2. The notation S denotes the simulator which can simulate a view $V(H_m)^*$. From an adversary's view, $V(H_m)^*$ and $V(H_m)=\{\mathbb{C}, SEITI, \{T_{R_1},\ldots,T_{R_m}\}, \{C_1,\ldots,C_m\}\}$ exhibit computational indistinguishability. To achieve this, the simulator S does the followings:

(1) Multi-dimensional data are encrypted by using a secure encryption method. The ciphertexts output by the secure encryption method are computationally indistinguishable from random values. The set of these ciphertexts is $\mathbb{C}$ which is available in $SEITI$. S randomly chooses $|\mathbb{C}|$ values as $\mathbb{C}^*$. Thus, $\mathbb{C}^*$ and $\mathbb{C}$ exhibit computational indistinguishability.

(2) The privilege value pv of the root node in $SEITI$ is randomly chosen. The privilege value of each node in $SEITI$ is calculated by using pv and the hash function H. Thus, the privilege value of each node can also be seen as randomly chosen. Then, for each node in $SEITI$, its security enhanced MDR code $[C_{MDR}]$ is calculated by using its privilege value, one-way hash function and bloom filter. As $[C_{MDR}]$ is calculated by using its privilege value, $[C_{MDR}]$ can also be seen as a binary string in which each bit is randomly set to 0 or 1. As the index $SEITI$ is available, S constructs an index $SEITI^*$ s.t. (i) The structures of $SEITI$ and $SEITI^*$ are the same; (ii) Each node of $SEITI^*$ is associated with a binary string in which each bit is randomly set to 0 or 1; (iii) The lengths of the binary strings in the nodes of $SEITI$ and $SEITI^*$ are the same. Thus, $SEITI^*$ and $SEITI$ exhibit computational indistinguishability.

(3) The search token T_{R_i} ($i=1,2,\ldots,m$) is available. T_{R_i} consists of a privilege path PP and a tree T_{SRCH}'. S constructs a search token $T_{R_i}{}^*$ by using the next two steps. First, S constructs a privilege path PP^*. The lengths of PP and PP^* are the same. Additionally, as (i) the binary strings in the nodes of $SEITI^*$ and $SEITI$ are computationally indistinguishable and (ii) the values in PP^* and PP are generated according to the binary strings in the nodes of $SEITI^*$ and $SEITI$, the values in PP and PP^* are computationally indistinguishable. Thus, PP and PP^* exhibit computational indistinguishability. Second, S constructs a tree $T_{SRCH}'^*$. T_{SRCH}' and $T_{SRCH}'^*$ have the same structure. S finds a subtree T^* in $SEITI^*$: (i) T^* and $T_{SRCH}'^*$ have the same structure; (ii) Each node in $T_{SRCH}'^*$ corresponds to a node in T^*. For each node in $T_{SRCH}'^*$, S randomly sets its binary string to a s.t. $a\,\&\,b=a$, where a is the binary string in the node of $T_{SRCH}'^*$ and b is the binary string in the corresponding node of T^* ($\&$ denotes bitwise AND operation). As the binary strings in the nodes of T_{SRCH}' are generated according to the privilege values and these privilege values can be seen as randomly chosen, these binary strings can be seen as randomly chosen. As the binary strings in the nodes of $T_{SRCH}'^*$ are also generated according to the privilege values and these privilege values can be seen as randomly chosen, these binary strings can also be seen as randomly chosen. As (i) the binary strings in the nodes of T_{SRCH}' and $T_{SRCH}'^*$ are computationally indistinguishable and (ii) T_{SRCH}' and $T_{SRCH}'^*$ have the same

structure, T_{SRCH} ' and T_{SRCH} '* are computationally indistinguishable. As PP and T_{SRCH} ' in T_{R_i} are computationally indistinguishable from PP^* and $T_{R_i}{}^*$ respectively ($i=1,\ldots,m$), the search tokens $\{T_{R_1},\ldots,T_{R_m}\}$ and $\{T_{R_1}{}^*,\ldots,T_{R_m}{}^*\}$ exhibit computational indistinguishability.

(4) S generates $\{C_1^*,\ldots,C_m^*\}$, where C_i^* ($i=1,\ldots,m$) denotes an empty set. The ciphertext set $\mathbb{C}$ and the set of retrieved ciphertext result sets $\{C_1,\ldots,C_m\}$ are available. For each ciphertext $c\in\mathbb{C}$, if $c\in C_1$, S chooses a ciphertext c^* in $\mathbb{C}^*$ and then adds c^* to C_1^*. Next, S judges whether $c\in C_2$. If $c\in C_2$, S also adds c^* to C_2^*. By using the method iteratively, S can obtain $\{C_1^*,\ldots,C_m^*\}$. It is obviously that $|C_i|=|C_i^*|$. As (i) $|C_i|=|C_i^*|$ and (ii) the ciphertexts in $\mathbb{C}$ and $\mathbb{C}^*$ are computationally indistinguishable, $\{C_1,\ldots,C_m\}$ and $\{C_1^*,\ldots,C_m^*\}$ exhibit computational indistinguishability.

Given that each element of $V(H_m)$ and $V(H_m)^*$ exhibit computational indistinguishability, we can conclude that our scheme meets the security definition outlined in Theorem 2.

10. Conclusion and Discussion

In this work, we propose SEITI. SEITI can be used to perform range queries on the ciphertexts of multi-dimensional data in the scenario of multiple users. Compared with order-preserving encryption schemes[21, 22], as the ciphertexts in the same MDR are computationally indistinguishable, SEITI can effectively preserve the ordering of ciphertexts. Compared with bucketization methods[7, 8, 9, 10], as the privilege values can only calculated by using one-way hash function, SEITI can support multiple users by distributing different privilege values to different users. Compared with public-key based ciphertexts search methods[11, 12, 13], as the bitwise AND operations are used, the range query in SEITI is more efficient. Compared with bloom filter based methods[15,16,17,18], SEITI can support multiple users to perform range queries over ciphertexts. SEITI also exists some drawbacks, including the absence of dynamic data update, secret parameter management and efficient privilege revocation. In future endeavors, we will prioritize addressing these critical issues.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61962029, 62262033, 62062045, 62341206), the Zhejiang Province Visiting Engineer Cooperation Project (No. FG2023061).

References

[1] S. Zhang, L. T. Yang, J. Feng et al., "A tensor-network-based big data fusion framework for Cyber–Physical–Social Systems (CPSS)," *Information Fusion*, vol.76, pp.337-354, 2021. Article (CrossRefLink)

[2] Z. Mei, B. Wu, S. Tian et al., "Fuzzy Keyword Search Method over Ciphertexts supporting Access Control," *KSII Transactions on Internet and Information Systems (TIIS)*, vol.11, no.11, pp.5671-5693, 2017. Article (CrossRefLink)

[3] J. Feng, L. T. Yang, Q. Zhu, and K.-K. R. Choo, "Privacy-Preserving Tensor Decomposition Over Encrypted Data in a Federated Cloud Environment," *IEEE Transactions on Dependable and Secure Computing*, vol.17, no.4, pp.857-868, 2020. Article (CrossRefLink)

[4] J. Feng, L. T. Yang, R. Zhang, W. Qiang, and J. Chen, "Privacy Preserving High-Order Bi-Lanczos in Cloud–Fog Computing for Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol.18, no.10, pp.7009-7018, 2022. Article (CrossRefLink)

[5] Z. Wu, R. Wang, Q. Li, X. Lian, G. Xu, E. Chen, X. Liu, "A Location Privacy-Preserving System Based on Query Range Cover-Up or Location-Based Services," *IEEE Transactions on Vehicular Technology*, vol.69, no.5, pp.5244-5254, 2020. Article (CrossRefLink)

[6] D. Sharma, "Searchable encryption : A survey," *Information Security Journal: A Global Perspective*, vol.32, no.2, pp.76-119, 2023. Article (CrossRefLink)

[7] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol.21, pp.333-358, 2012. Article (CrossRefLink)

[8] P. Wang, C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using Rp-trees," in *Proc. of 2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp.314-325, 2013. Article (CrossRefLink)

[9] Y. Lee, "Secure Ordered Bucketization," *IEEE Transactions on Dependable and Secure Computing*, vol.11, no.3, pp.292-303, 2014. Article (CrossRefLink)

[10] W. Lin et al., "Towards Secure and Efficient Equality Conjunction Search Over Outsourced Databases," *IEEE Transactions on Cloud Computing*, vol.10, no.2, pp.1445-1461, 2022. Article (CrossRefLink)

[11] Y. Lu, "Privacy-Preserving Logarithmic-time Search on Encrypted Data in Cloud," in *Proc. of NDSS Symposium 2012*, 2012. Article (CrossRefLink)

[12] Z. Mei et al., "Secure multi-dimensional data retrieval with access control and range query in the cloud," *Information Systems*, vol.122, 2024. Article (CrossRefLink)

[13] H. Qiao et al., "Ciphertext Range Query Scheme Against Agent Transfer and Permission Extension Attacks for Cloud Computing," *IEEE Internet of Things Journal*, vol.11, no.10, pp.17975-17988, 2024. Article (CrossRefLink)

[14] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol.13, no.7, pp.422-426, 1970. Article (CrossRefLink)

[15] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast and Scalable Range Query Processing With Strong Privacy Protection for Cloud Computing," *IEEE/ACM Transactions on Networking*, vol.24, no.4, pp.2305-2318, 2016. Article (CrossRefLink)

[16] B. Wang, M. Li, and H. Wang, "Geometric Range Search on Encrypted Spatial Data," *IEEE Transactions on Information Forensics and Security*, vol.11, no.4, pp.704-719, 2016. Article (CrossRefLink)

[17] N. Cui, J. Li, X. Yang, B. Wang, M. Reynolds, and Y. Xiang, "When Geo-Text Meets Security: Privacy-Preserving Boolean Spatial Keyword Queries," in *Proc. of 2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp.1046-1057, 2019. Article (CrossRefLink)

[18] H. Mahdikhani et al., "Achieving Efficient and Privacy-Preserving Range Query in Fog-enhanced IoT with Bloom Filter," in *Proc. of ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp.1-6, 2020. Article (CrossRefLink)

[19] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao and J. Pieprzyk, "Dynamic Searchable Symmetric Encryption Schemes Supporting Range Queries with Forward (and Backward) Security," in *Proc. of 23rd European Symposium on Research in Computer Security, ESORICS 2018*, LNSC, vol.11099, pp.228-246, 2018. Article (CrossRefLink)
[20] J. Wang, and S. S. M. Chow, "Forward and Backward-Secure Range-Searchable Symmetric Encryption," in *Proc. of Privacy Enhancing Technologies Symposium*, vol.2022, no.1, pp.28-48, 2022. Article (CrossRefLink)
[21] R. A. Popa, F. H. Li, and N. Zeldovich, "An Ideal-Security Protocol for Order-Preserving Encoding," in *Proc. of 2013 IEEE Symposium on Security and Privacy*, pp.463-477, 2013. Article (CrossRefLink)
[22] F. Kerschbaum, and A. Schroepfer, "Optimal Average-Complexity Ideal-Security Order-Preserving Encryption," in *Proc. of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp.275-286, 2014. Article (CrossRefLink)
[23] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. of the 2009 ACM SIGMOD International Conference on Management of data*, pp.139-152, 2009. Article (CrossRefLink)
[24] Z. Mei, H. Zhu, Z. Cui, Z. Wu, G. Peng, B. Wu, and C. Zhang, "Executing multi-dimensional range query efficiently and flexibly over outsourced ciphertexts in the cloud," *Information Sciences*, vol.432, pp.79-96, 2018. Article (CrossRefLink)
[25] L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial keyword query processing: an experimental evaluation," in *Proc. of the VLDB Endowment*, vol.6, no.3, pp.217-228, 2013. Article (CrossRefLink)
[26] B. Wang, M. Li, and L. Xiong, "FastGeo: Efficient Geometric Range Queries on Encrypted Spatial Data," *IEEE Transactions on Dependable and Secure Computing*, vol.16, no.2, pp.245-258, 2019. Article (CrossRefLink)
[27] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich Queries on Encrypted Data: Beyond Exact Matches," in *Proc. of 20th European Symposium on Research in Computer Security*, LNSC, vol.9327, pp.123-145, 2015. Article (CrossRefLink)
[28] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis, "Practical Private Range Search Revisited," in *Proc. of the 2016 International Conference on Management of Data (SIGMOD 2016)*, pp.185-198, 2016. Article (CrossRefLink)
[29] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient Similarity Search over Encrypted Data," in *Proc. of 2012 IEEE 28th International Conference on Data Engineering*, pp.1156-1167, 2012. Article (CrossRefLink)
[30] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. of IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp.2112-2120, 2014. Article (CrossRefLink)
[31] Y. Zhan, D. Shen, P. Duan et al., "MDOPE: Efficient multi-dimensional data order preserving encryption scheme," *Information Sciences*, vol.595, pp.334-343, 2022. Article (CrossRefLink)
[32] S. Su, Y. Teng, X. Cheng, K. Xiao, G. Li, and J. Chen, "Privacy-Preserving Top-k Spatial Keyword Queries in Untrusted Cloud Environments," *IEEE Transactions on Services Computing*, vol.11, no.5, pp.796-809, 2018. Article (CrossRefLink)

Zhuolin Mei received the B.S. degree from the Wuhan University of Technology, China, in 2009, and the Ph.D. degree from the Huazhong University of Science and Technology, China, in 2017. He is currently a Lecturer with Jiujiang University, China. His research interests include cloud computing, data security, data search, and access control.

Jing Zeng received his Ph.D. degree from the Huazhong University of Science and Technology. He is currently a bigdata expert in China Gridcom Co.,Ltd. His research interests include power bigdata, E-government big data, AI, AIoT, Security and Privacy.

Caicai Zhang received the Ph.D. degree from the Huazhong University of Science and Technology, China, in 2016. She is currently a Lecturer with Zhejiang Polytechnic University of Mechanical and Electrical Engineering, China. Her research interests include deep learning, data security, and uncertain data management.

Shimao Yao received the B.S. degree from the Nanjing University of Aeronautics and Astronautics, China, in 2003, the M.S. degree from the Dalian University of Technology, China, in 2010, and the Ph.D. degree from the Kunsan National University, South Korea, in 2021. He is currently a Lecturer with Jiujiang University, China. His research interests include information security and computer networks.

Jiaoli Shi received the B.S. degree from the Guilin University of Electronic Technology, China, in 2000, the M.S. degree from the Huazhong University of Science and Technology, China, in 2006, and the Ph.D. degree from Wuhan University, China, in 2017. She is currently an Associate Professor with Jiujiang University, China. Her research interests include cloud storage security and privacy preserving.

Bin Wu received the Ph.D. degree from the Huazhong University of Science and Technology, China, in 2017. He is currently a Lecturer with Jiujiang University. His research interests include privacy preserving, big data security, cloud security, information security, and information query.