Tree-Based Multi-dimensional Range Search on Encrypted Data with Enhanced Privacy

Boyang Wang[1(✉)], Yantian Hou[1], Ming Li[1], Haitao Wang[1], Hui Li[2], and Fenghua Li[3]

[1] Department of Computer Science, Utah State University, Logan, USA
{bywang.usu,houyantian}@gmail.com, {ming.li,haitao.wang}@usu.edu
[2] State Key Laboratory of Integrated Services Networks, Xidian University, Xián, China
lihui@mail.xidian.edu.cn
[3] State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing, China
lfh@iie.ac.cn

Abstract. With searchable encryption, a data user is able to perform meaningful search on encrypted data stored in the public cloud without revealing data privacy. Besides handling simple queries (e.g., keyword queries), complex search functions, such as multi-dimensional (conjunctive) range queries, have also been studied in several approaches to provide search functionalities over multi-dimensional data. However, current works supporting multi-dimensional range queries either only achieve linear search complexity or reveal additional private information to the public cloud. In this paper, we propose a tree-based symmetric-key searchable encryption to support multi-dimensional range queries on encrypted data. Besides protecting data privacy, our proposed scheme is able to achieve faster-than-linear search, query privacy and single-dimensional privacy simultaneously compared to previous solutions. More specifically, we formally define the security of our proposed scheme, prove that it is selectively secure, and demonstrate its faster-than-linear efficiency with experiments over a real-world dataset.

Keywords: Multi-dimensional range search · Encrypted data

1 Introduction

With the low-priced data storage and computation services offered by cloud providers, people outsource their large-scale data to the cloud to reduce their cost spending on local devices. While enjoying data services in the public cloud, the leakage of private data has always been one of the major concerns to users [21]. Using *traditional encryption on the client side*, such as the example of implementing AES-256 (on the client side) in a cloud data storage application named Wuala [1], people can preserve their private data, even from the public

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 374–394, 2015.
DOI: 10.1007/978-3-319-23829-6_26

cloud. However, the implementation of traditional encryption on the client side will prevent clients to utilize and compute their cloud data efficiently.

For instance, a client using Wuala has no way to operate meaningful search on its encrypted cloud data, unless it first retrieves/syncs all the data from the cloud side and decrypts those ciphertexts. This process is resource-consuming to a client, especially for medical data or financial data with large-scale data size. The client will need to face the same awkward and painful situation when it simply encrypts data with traditional encryption on the client side, and stores those ciphertexts in Google Drive or Amazon S3. Of course, sharing the secret key with the cloud, which is equivalent to *traditional encryption on the cloud side* (e.g., Dropbox), is another option to conduct search on outsourced data, but that will totally reveal confidential data to the public cloud.

To enable users to search their encrypted cloud data without retrieving the entire data or revealing private data to the public cloud, the techniques of searchable encryption were proposed. Most of the current works [5,7,8,10,12,14,15,17,24,26,27,30] focus on supporting simple search functions, such as keyword queries. However, they are not suitable for handling complex search operations, such as *multi-dimensional range queries*, on encrypted data in real datasets, where plain data are generally presented with numerical values in multiple dimensions.

Some previous schemes or the extensions of them [6,23] can support multi-dimensional range queries, but the search complexity of these schemes is *linearly* increasing with the number of data records in a dataset. Moving a step forward, several schemes [18,31] proposed to utilize multi-dimensional tree structures, such as kd-trees [4] and R-trees [13], to achieve *faster-than-linear* search regarding to the total number of data records. However, as pointed out in [28], these solutions reveal *single-dimensional privacy* to the cloud, which simply allows the public cloud to reveal additional privacy by performing range search in every single dimension correctly and independently while only granted with a search token of a multi-dimensional range query. To protect this privacy leakage while still maintaining faster-than-linear search, Wang et al. [28] recently designed a tree-based public-key multi-dimensional range searchable encryption based on Hidden Vector Encryption [6] and R-trees. Unfortunately, as a trade-off, this scheme inherently loses *query privacy (i.e., the public cloud learns the content of the queries submitted by the client)* due to its public-key-based design [22].

In this paper, to overcome the limitations and enhance users' privacy in previous solutions, we design Elm[1], a tree-based symmetric-key multi-dimensional range searchable encryption. With this proposed scheme, a data owner is able to index its data records with an R-tree, encrypt all the nodes/data in the tree, and outsource the encrypted tree to the public cloud. The public cloud is able to correctly perform multi-dimensional range search on encrypted data without

[1] We name it Elm because it is a tree-based solution and it can enhance users' privacy for multi-dimensional range queries. For the ease of description, when we mention a scheme is faster-than-linear in the rest of this paper, it indicates that the search complexity of it is faster-than-linear with regard to the number of data records.

revealing query privacy, data privacy or single-dimensional privacy. The main contributions of this paper are summarized as follows:

1. We formally describe the definition of a tree-based symmetric-key multi-dimensional range searchable encryption, and present the formal security of it in terms of query privacy, data privacy and single-dimensional privacy.
2. Besides preserving data privacy, our scheme achieves faster-than-linear search, query privacy and single-dimensional privacy *simultaneously* compared to previous solutions (as shown in Table 1). Specifically, we leverage a symmetric-key predicate encryption [22] (denoted as SSW in this paper) to encrypt all the nodes in an R-tree, so that the public cloud is able to still follow the original search algorithm (i.e., the one in the plaintext domain) of an R-tree by testing corresponding geometric relations on encrypted data.
3. We prove our scheme is selectively secure and demonstrate its efficiency on a real dataset. Moreover, compared to [28], our scheme can securely support *dynamic data* of an R-tree for some simple cases due to the enhancement of query privacy.

Note that the use of an R-tree in this paper is two-fold: (1) it achieves faster-than-linear search; (2) it is more suitable for maintaining single-dimensional privacy compared to other multi-dimensional tree structures (further explanations about this privacy issue with different tree structures can be found in [28]).

Table 1. Comparison among Different Solutions.

	[18]	[31]	[28]	[23]	[6]	Ours
Faster-than-linear Search	√	√	√	×	×	√
Query Privacy	√	√	×	×	×	√
Single-Dimensional Privacy	×	×	√	√	√	√

2 Related Work

Keyword Search. Song et al. [24] proposed the first symmetric-key searchable encryption. Golle et al. [12] designed a scheme for processing conjunctive keyword queries. Curtmola et al. [10] rigorously defined and discussed the security of searchable symmetric encryption for keyword queries, and also studied the multi-user setting. Kamara et al. [14,15] and Stefanov et al. [26] presented keyword search over dynamic encrypted data. Sun et al. [27] designed a multi-keyword search scheme, which can support similarity-based ranking on encrypted data. Cash et al. [8] recently proposed a sublinear searchable encryption to support conjunctive keyword search and boolean keyword search, and they further studied the dynamic version of their work in [7].

Keyword search on encrypted data have also been studied in the public-key setting. Boneh et al. [5] designed the first public-key encryption with keyword

search (PEKS). Abdalla et al. [3] further studied the connections between anonymous Identity-Based Encryption and PEKS. Lai et al. [17] proposed a public-key searchable encryption to perform expressive keyword queries. However, these works discussed above mainly focus on keyword search, which is not sufficient to handle multi-dimensional range queries on encrypted data.

More recently, a scheme [33] with data interoperability has been proposed to flexibly enable a set of SQL queries (including keyword search, range search, etc.) on encrypted data. Unfortunately, it fails to achieve faster-than-linear search or preserve single-dimensional privacy for multi-dimensional range queries. On the other hand, Pappas et al. [20] introduced a scheme (named Blind Seer) to flexibly support arbitrary boolean queries with sublinear search by using Bloom-Filter-based tree structure. However, a large amount of client-server interactions are required to finish the entire search process (essentially, one round of client-server interaction is needed to make search decision at each node in the tree).

Range Search. Boneh et al. [6] designed a general public-key approach to support comparison queries, subset queries and range queries on encrypted data by leveraging Hidden Vector Encryption (HVE). Shi et al. [23] studied a public-key scheme, which can improve the search complexity of each data record to $O(w \log T)$ compared to $O(wT)$ in [6]. Unfortunately, these two approaches are public-key-based, which fail to provide query privacy [22].

Lu [18] proposed a logarithmic range search scheme on encrypted data, named LSED, by utilizing segment trees, predicate encryption (i.e., SSW [22]) and B^+ trees. The extension of it, denoted as $LSED^+$, can support multi-dimensional range queries by replacing B^+ trees with kd-trees in their design. However, as pointed out by the author himself, this extension reveals single-dimensional privacy. Wang et al. [31] presented a scheme for performing multi-dimensional range queries with the use of R-trees and Asymmetric Scalar-product Preserving Encryption [32]. Unfortunately, this scheme leaks single-dimensional privacy and lacks the formal security definition.

Recently, Wang et al. [28] designed a tree-based public-key MDRSE based on HVE and multi-dimensional tree structures (i.e., R-trees). This scheme is able to achieve faster-than-linear search. More importantly, the authors explained that some similar tree structures, such as kd-trees and range-trees, are inherently not able to achieve single-dimensional privacy. However, since it is a public-key scheme, it loses query privacy as well for the same reason as [6,23].

3 Preliminaries

Predicate Encryption. Predicate encryption is able to test whether plain data (e.g., u) satisfies a predicate (i.e., $f(u) = 1$ or $f(u) = 0$) without revealing plain data. SSW [22] is a symmetric-key predicate encryption and is able to support inner product queries. Specifically, data is described as a vector $\boldsymbol{u}$ and a predicate can be denoted as a vector $\boldsymbol{v}$, and the evaluation on encrypted data reveals $f(\boldsymbol{u}) = 1$ iff $\boldsymbol{v} \circ \boldsymbol{u} = 0$, where $\boldsymbol{v} \circ \boldsymbol{u} = \sum_{i=1}^{n} v_i \cdot u_i$ denotes the inner product of these two vectors. Besides protecting data privacy, SSW can also preserve query privacy. The details of SSW are presented in Fig. 1.

- $\texttt{Setup}(1^\lambda, T)$: Given a security parameter λ and T, output a secret key $\texttt{SK}$.
- $\texttt{Enc}(\texttt{SK}, \mathbf{u})$: Given $\texttt{SK}$ and a plaintext $\mathbf{x} \in \mathcal{U}$, where $\mathbf{u} = (u_1, ..., u_T)$ and $\mathcal{U}$ is the plaintext space, output a ciphertext C.
- $\texttt{GenToken}(\texttt{SK}, \mathbf{v})$: Given $\texttt{SK}$ and a query $\mathbf{v} \in \mathcal{V}$, where $\mathbf{v} = (v_1, ..., v_T)$ and $\mathcal{V}$ is the query space, output a token $\texttt{TK}$.
- $\texttt{Query}(\texttt{TK}, C)$: Given $\texttt{TK}$ and C, output 1 iff $\mathbf{v} \circ \mathbf{u} = 0$ and 0 otherwise.

Correctness: SSW is correct, for all λ, all $\mathbf{u} \in \mathcal{U}$, all $\mathbf{v} \in \mathcal{V}$, all $\texttt{SK} \leftarrow \texttt{Setup}(1^\lambda, T)$, all $C \leftarrow \texttt{Enc}(\texttt{SK}, \mathbf{u})$, all $\texttt{TK} \leftarrow \texttt{GenToken}(\texttt{SK}, \mathbf{v})$,

- If $\mathbf{v} \circ \mathbf{u} = 0$, $\texttt{Query}(\texttt{TK}, C) = 1$;
- If $\mathbf{v} \circ \mathbf{u} \neq 0$, $\Pr[\texttt{Query}(\texttt{TK}, C) = 0] \geq 1 - \texttt{negl}(\lambda)$;

where $\texttt{negl}(\lambda)$ is a negligible function in λ.

Fig. 1. Details of SSW.

R-trees. R-trees are height-balanced tree structures to index data with multiple dimensions. It can improve the search efficiency of range queries on multi-dimensional data. An example of an R-tree in two dimensions can be found in Fig. 3. The essential idea of indexing data in an R-tree is to group nearby elements (points or hyper-rectangles) on the same level and include them into a *minimal bounding hyper-rectangle* in a higher-level of the tree. Every leaf node in an R-tree represents a point, and every non-leaf node describes a bounding hyper-rectangle. Clearly, the root node of an R-tree is the largest bounding hyper-rectangle that covers all the elements.

With this structure, the search of range queries in an R-tree can be efficiently conducted from the root node by recursively checking geometric relations, including whether two hyper-rectangles (a non-leaf node and a query) intersect or whether a point (a leaf node) is inside a hyper-rectangle (a query). Specifically, for each non-leaf node, if it interacts with the query, continue to search its child nodes; otherwise, stop search on this path. For each leaf node, if it is inside the query, return this node; otherwise, do not return.

4 Problem Statement

System Model. In the system model of a searchable encryption, we have two entities, a data owner and the cloud server (which are illustrated in Fig. 2). A data owner outsources its data (i.e., a large set of data records) to the cloud server in order to save local storage cost. In addition, this data owner still would like to use its outsourced data correctly and efficiently. Specifically, in the study of this paper, that means this data owner should be able to retrieve the correct results of its data from the cloud server for each multi-dimensional range query. The cloud server is considered as an *honest-but-curious* party. It means the cloud server is believed to be able to provide reliable services, but it may be curious about the content of data records stored in the cloud and the content of queries submitted by the data owner. In order to preserve users' privacy, data and queries are in an encrypted form in the cloud.

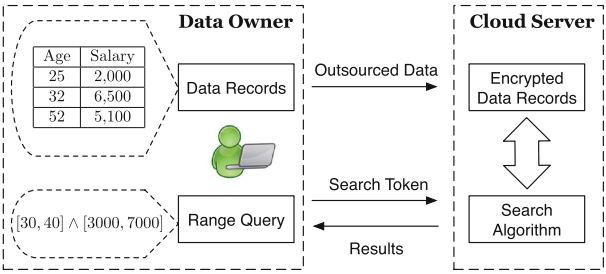

Fig. 2. The system model includes a data owner and the cloud server.

Definitions. We first briefly present some basic definitions for data in multiple dimensions, which will be frequently used in the rest of this paper.

- ***Lattice****: Let* $\Delta = (T_1, ..., T_w)$*, where* T_i *is the upper bound in the* i*-th dimension and* $1 \leq i \leq w$*. A lattice* $\mathbb{L}_\Delta$ *is defined as* $\mathbb{L}_\Delta = [T_1] \times \cdots \times [T_w]$*, where* $[T_i] = \{1, ..., T_i\}$*.*
- ***Point****: A point* X *in* $\mathbb{L}_\Delta$ *is defined as* $X = (x_1, ..., x_w)$*, where* x_i *is a value in the* i*-th dimension,* $x_i \in [T_i]$*,* $\forall i \in [1, w]$*.*
- ***Hyper-Rectangle****: A hyper-rectangle* $\mathbf{HR}$ in $\mathbb{L}_\Delta$ *is defined as* $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$*, where* $\mathbf{R}_i$ *is a range in the* i*-th dimension,* $\mathbf{R}_i \subseteq [1, T_i]$*,* $\forall i \in [1, w]$*.*

Considering the preceding system model, a data record is essentially a point and a multi-dimensional range query is actually a hyper-rectangle.

We now introduce the formal definition of a symmetric-key Multi-Dimensional Range Searchable Encryption (MDRSE). In addition, we leverage a tree structure Γ (more specifically, an R-tree in the design of this paper), which is able to index data records and improve the search complexity of multi-dimensional range queries. Compared to the recent work [28], which is also a tree-based solution supporting multi-dimensional range search, the major difference of our scheme is that it is a symmetric-key approach while the previous one is a public-key scheme. This change from a public-key design to a symmetric-key one will enhance query privacy, which will be further discussed later.

Definition 1 ***(Symmetric-Key Multi-Dimensional Range Searchable Encryption).*** *A tree-based symmetric-key MDRSE is a tuple of five polynomial-time algorithms* $\Pi = (\texttt{GenKey}, \texttt{BuildTree}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ *such that:*

- $\texttt{SK} \leftarrow \texttt{GenKey}(1^\lambda, \Delta)$*: is a probabilistic key generation algorithm that is run by the data owner to setup the scheme. It takes as input a security parameter* λ *and* $\Delta = (T_1, ..., T_w)$*, and outputs a secret key* SK*.*
- $\Gamma \leftarrow \texttt{BuildTree}(\mathbf{D})$*: is a deterministic algorithm run by the data owner to build a multi-dimensional tree to index data records. It takes as input* n *data records* $\mathbf{D} = \{D_1, ..., D_n\}$*, where each data record* $D_i = (d_{i,1}, ..., d_{i,w})$ *is essentially a point in* $\mathbb{L}_\Delta$*, and outputs a multi-dimensional tree* $\Gamma = \{D_1, ..., D_n, N_1, ..., N_m, \mathbf{P}\}$*, where* D_i *is a leaf node, for* $1 \leq i \leq n$*, and* N_j *is a non-leaf*

node, for $1 \leq j \leq m$*, and* $\mathbf{P}$ *is the set of pointers covering all the parent-child relations in tree* Γ*.*

- $\Gamma^* \leftarrow \texttt{Enc}(\texttt{SK}, \Gamma)$*: is a probabilistic algorithm run by the data owner to encrypt a multi-dimensional tree. It takes as input a secret key* SK*, multi-dimensional tree* Γ*, and outputs an encrypted multi-dimensional tree* $\Gamma^* = \{C_1, ..., C_n, E_1, ..., E_m, \mathbf{P}\}$*, where* C_i *is an encrypted leaf node, for* $1 \leq i \leq n$*,* E_j *is an encrypted non-leaf node, , for* $1 \leq j \leq m$*, and* $\mathbf{P}$ *is the set of pointers covering all the parent-child relations in tree* Γ^**.*
- $\texttt{TK} \leftarrow \texttt{GenToken}(\texttt{SK}, Q)$*: is a probabilistic algorithm run by the data owner to generate a search token for a given range query. It takes as input a secret key* SK *and a range query (i.e., a hyper-rectangle)* Q*, and outputs a search token* TK*.*
- $\mathbf{I} \leftarrow \texttt{Search}(\Gamma^*, \texttt{TK})$*: is a deterministic algorithm run by the server to search over an encrypted multi-dimensional tree. It takes as input an encrypted multi-dimensional tree* Γ^* *and a search token* TK*, and outputs a set* $\mathbf{I}$ *of identifiers (memory locations of data records in the cloud server), where* $I_i \in \mathbf{I}$*, if data record* $D_i \in Q$*.*

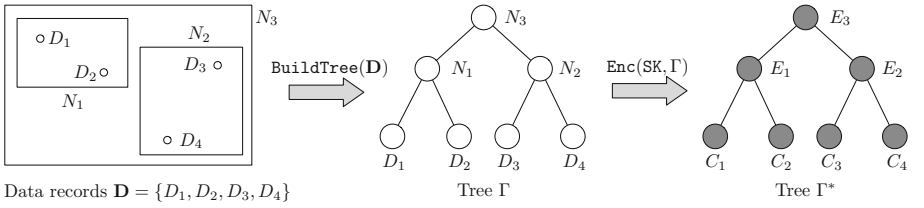

Fig. 3. A set of data records is indexed by a tree Γ, and is further encrypted into an encrypted tree Γ^* where $\Gamma \simeq \Gamma^*$.

In the above tree-based scheme, we describe a tree (also its encrypted version) with a set of nodes and a set of pointers covering all the parent-child relations in the tree. During encryption, the algorithm only encrypts every node (including every leaf node D_i and every non-leaf node N_j) in tree Γ while keeping all the pointers (i.e., $\mathbf{P}$) unchanged. It means although the nodes are denoted with ciphertexts in the encrypted tree Γ^*, the graph structures of the original tree Γ and its encrypted version Γ^* are *isomorphic*, which is denoted as $\Gamma \simeq \Gamma^*$. An example of the encryption on a tree is described in Fig. 3.

Informally, we say that the encryption algorithm in our scheme encrypts nodes only (*in order to protect data privacy*), but does not change the tree structure (*so faster-than-linear search can still be functional*). The encryption on nodes will be carried by multiple instances (i.e., *one instance for each node*) of predicate encryption (i.e. SSW [22] presented in Sect. 3).

Correctness. *We say that the above tree-based symmetric-key MDRSE is correct if for all* $\lambda \in \mathbb{N}$*, all* SK output by $\texttt{GenKey}(1^\lambda, \Delta)$*, all* $D_i \in \mathbb{L}_\Delta$*, all* Γ *output*

by $\texttt{BuildTree}(\mathbf{D})$, *all* Γ^* *output by* $\texttt{Enc}(\texttt{SK}, \Gamma)$, *all* $Q \subseteq \mathbb{L}_\Delta$, *all* $\texttt{TK}$ *output by* $\texttt{GenToken}(\texttt{SK}, Q)$, *for any* $i \in [1, n]$

- *If* $D_i \in Q$, *then* $\texttt{Search}(\Gamma^*, \texttt{TK}) = \mathbf{I}$, *where* $I_i \in \mathbf{I}$;
- *If* $D_i \notin Q$, *then* $\Pr[\texttt{Search}(\Gamma^*, \texttt{TK}) = \mathbf{I}$, *where* $I_i \notin \mathbf{I}] \geq 1 - \texttt{negl}(\lambda)$;

where $\texttt{negl}(\lambda)$ *denotes a negligible function in* λ.

Informally, the correctness of the searchable encryption described above means that it will definitely return the identifier of a data record if this data record indeed satisfies a given query; on the other hand, it will return the identifier with a *negligible* probability if this data record actually fails to match a given query.

5 Security Definitions

In this section, we first capture all the possible privacy leakage with a leakage function, and then we formally define the security of a tree-based symmetric-key MDRSE based on this leakage function.

Leakage Function. A *leakage function* includes all the privacy leakage in a searchable encryption. The leakage function in a tree-based symmetric-key MDRSE introduced by a set of data records $\mathbf{D}$, its tree structure Γ and a query Q can be described as $\mathcal{L}(\mathbf{D}, \Gamma, Q)$, which includes

- ***Size Pattern****: the cloud server learns the number of data records* n *in the dataset, the size of each dimension* $|T_i|$, *and the number of queries submitted by the data owner.*
- ***Access Pattern****: the cloud server reveals the identifiers of data records that are returned for each submitted query.*
- ***Search Pattern****: the cloud server learns if the same data record is retrieved by two different queries.*
- ***Path Pattern****: the cloud server learns how exactly the search algorithm traverses from the root node to the matched leaf nodes for each given query, i.e., the identifiers of all the nodes in the paths traversed by the search of each given query.*

Note that most of the searchable encryption schemes do not protect size pattern, access pattern or search pattern. Path pattern is recently introduced in [20,28] and defined specifically for tree-based solutions, because the original definition of access pattern is not sufficient to capture all the privacy leakage in some tree structures. Essentially, it is a special type of access pattern in trees [20]. The leakage of path pattern in a tree-based MDRSE is actually not hard to explain. Since the encryption algorithm does not modify the structure of a tree (see Fig. 3 again), which makes the cloud server easily reveals path pattern.

Theoretically speaking, the use of Oblivious RAMs [11,25] can preserve access pattern and search pattern from the cloud server. Unfortunately, compared to searchable encryption, the efficiency of Oblivious RAMs is still a major concern. How to particularly preserve the privacy defined in the above leakage function is out of scope of this paper.

Query Privacy. The main security objective of a tree-based symmetric-key MDRSE in this paper is to achieve *query privacy*, *data privacy* and *single-dimensional privacy*. Each of these three privacy can be rigorously defined in a *selective* manner [22]. We start with query privacy first. Informally, *selective query privacy* means by submitting two multi-dimensional range queries Q_0 and Q_1, a computationally bounded adversary is able to *adaptively* issue a number of ciphertext queries and token queries restricted by Q_0, Q_1 and leakage function $\mathcal{L}$. However, it is not able to distinguish this two range queries.

Definition 2 (Selective Query Privacy). *Let* $\Pi = (\texttt{GenKey}, \texttt{BuildTree}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ *be a tree-based symmetric-key MDRSE scheme over lattice* $\mathbb{L}_\Delta$, $\lambda \in \mathbb{N}$ *be the security parameter:*

- **Init:** *The adversary* $\mathcal{A}$ *submits two range queries* Q_0 *and* Q_1 *to the challenger, where* Q_0, $Q_1 \subseteq \mathbb{L}_\Delta$.
- **Setup:** *The challenger runs* $\texttt{GenKey}(1^\lambda, \Delta)$ *to generate a secret key* SK, *and it keeps* SK *private.*
- **Phase 1:** *The adversary* $\mathcal{A}$ *adaptively requests a number of queries, where each query is one of the two following types:*
  - *Ciphertext Query: On the* j*th ciphertext query, the adversary* $\mathcal{A}$ *outputs a tree* $\Gamma_j = \texttt{BuildTree}(\mathbf{D}_j)$, *where* $\mathbf{D}_j$ *is a set of data records described as* $\mathbf{D}_j = (D_{j,1}, ..., D_{j,n})$. *The challenger responses with an encrypted tree* $\Gamma_j^* = \texttt{Enc}(\texttt{SK}, \Gamma_j)$, *where* $\mathbf{D}_j$ *is subjected to the two following restrictions:*
    1. $\mathcal{L}(\mathbf{D}_j, \Gamma_j, Q_0) = \mathcal{L}(\mathbf{D}_j, \Gamma_j, Q_1)$;
    2. *And for* $1 \leq i \leq n$, *either* $(D_{j,i} \in Q_0) \wedge (D_{j,i} \in Q_1)$, *or* $(D_{j,i} \notin Q_0) \wedge (D_{j,i} \notin Q_1)$.
  - *Token Query: On the* j*th token query, the adversary* $\mathcal{A}$ *outputs a range query* Q'_j, *where* $Q'_j \subseteq \mathbb{L}_\Delta$. *The challenger responds with a search token* $\texttt{TK}'_j = \texttt{GenToken}(\texttt{SK}, Q'_j)$.
- **Challenge:** *With* Q_0, Q_1 *selected in* ***Init****, the challenger flips a coin* $b \in \{0, 1\}$ *and returns* $\texttt{TK}_b = \texttt{GenToken}(\texttt{SK}, Q_b)$ *to the adversary.*
- **Phase 2:** *The adversary* $\mathcal{A}$ *continues to adaptively request a number of queries, which are still subjected to the same restrictions in* ***Phase 1****.*
- **Guess:** *The adversary takes a guess* b' *of* b.

The advantage of adversary $\mathcal{A}$ *in the above selective query security game is defined as* $\mathbf{Adv}^{SQP}_{\Pi,\mathcal{A}}(1^\lambda, \Delta)$. *We say that scheme* Π *is selectively query secure if for all polynomial time adversaries have at most negligible advantage*

$$\mathbf{Adv}^{SQP}_{\Pi,\mathcal{A}}(1^\lambda, \Delta) = |\Pr[b' = b] - 1/2| \leq \texttt{negl}(\lambda).$$

where $\texttt{negl}(\lambda)$ *denotes a negligible function in* λ.

Since our scheme is symmetric-key-based, the challenger in the security game is able to response to the adversary with two types of queries, including ciphertext queries and token queries. While the recent work [28] only needs to consider token queries in its security game, because it is public-key-based, where the adversary possesses the encryption key and is allowed to obtain any ciphertexts

by itself (*i.e., the selection of data records for encryption has no restrictions compared to the ciphertext queries in* ***Phase 1*** *of the above security game*). In fact, as indicated in [22], this kind of ability that the adversary is capable of in the security game makes public-key solutions eventually reveal query privacy.

Data Privacy. Similarly like query privacy, data privacy can also be defined in a selective security game between the adversary and challenger. Informally, *selective data privacy* indicates by submitting two datasets $\mathbf{D}_0$ and $\mathbf{D}_1$, a computationally bounded adversary is able to *adaptively* issue a number of ciphertext queries and token queries restricted by $\mathbf{D}_0$, $\mathbf{D}_1$ and leakage function $\mathcal{L}$. However, it is not able to distinguish this two datasets.

Definition 3 (Selective Data Privacy). *Let* $\Pi = (\mathtt{GenKey}, \mathtt{BuildTree}, \mathtt{Enc}, \mathtt{GenToken}, \mathtt{Search})$ *be a tree-based symmetric-key MDRSE scheme over lattice* $\mathbb{L}_\Delta$, $\lambda \in \mathbb{N}$ *be the security parameter:*

- **Init:** *The adversary* $\mathcal{A}$ *submits two data record sets* $\mathbf{D}_0$ *and* $\mathbf{D}_1$ *with the same length and isomorphic tree structure* $\Gamma_0 \simeq \Gamma_1$, *where* $\mathbf{D}_0 = \{D_{0,1}, ..., D_{0,n}\}$, $\mathbf{D}_1 = \{D_{1,1}, ..., D_{1,n}\}$, $D_{0,i}$, $D_{1,i} \in \mathbb{L}_\Delta$, *for* $1 \leq i \leq n$, $\Gamma_0 = \mathtt{BuildTree}(\mathbf{D}_0)$ *and* $\Gamma_1 = \mathtt{BuildTree}(\mathbf{D}_1)$.
- **Setup:** *The challenger runs* $\mathtt{GenKey}(1^\lambda, \Delta)$ *to generate a secret key* SK, *and it keeps* SK *private.*
- **Phase 1:** *The adversary* $\mathcal{A}$ *adaptively requests a number of queries, where each query is one of the two following types:*
  - *Ciphertext Query: On the* j*th ciphertext query, the adversary* $\mathcal{A}$ *outputs a tree* $\Gamma'_j = \mathtt{BuildTree}(\mathbf{D}'_j)$, *where* $\mathbf{D}'_j$ *is a set of data records described as* $\mathbf{D}'_j = (D'_{j,1}, ..., D'_{j,n})$. *The challenger responses with an encrypted tree* $\Gamma'^*_j = \mathtt{Enc}(\mathtt{SK}, \Gamma'_j)$.
  - *Token Query: On the* j*th token query, the adversary* $\mathcal{A}$ *outputs a range query* Q_j, *where* $Q_j \subseteq \mathbb{L}_\Delta$. *The challenger responds with a search token* $\mathtt{TK}_j = \mathtt{GenToken}(\mathtt{SK}, Q_j)$, *where* Q_j *is subjected to the two following restrictions:*
    1. $\mathcal{L}(\mathbf{D}_0, \Gamma_0, Q_j) = \mathcal{L}(\mathbf{D}_1, \Gamma_1, Q_j)$;
    2. *And for* $1 \leq i \leq n$, *either* $(D_{0,i} \in Q_j) \wedge (D_{1,i} \in Q_j)$, *or* $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$.
- **Challenge:** *With* $\mathbf{D}_0$, $\mathbf{D}_1$ *selected in* ***Init****, the challenger flips a coin* $b \in \{0, 1\}$ *and returns* $\Gamma^*_b = \mathtt{Enc}(\mathtt{SK}, \Gamma_b)$ *to the adversary.*
- **Phase 2:** *The adversary* $\mathcal{A}$ *continues to adaptively request a number of queries, which are still subjected to the same restrictions in* ***Phase 1****.*
- **Guess:** *The adversary takes a guess* b' *of* b.

The advantage of adversary $\mathcal{A}$ *in the above selective data privacy game is defined as* $\mathbf{Adv}^{SDP}_{\Pi,\mathcal{A}}(1^\lambda, \Delta)$. *We say that scheme* Π *is selectively data secure if for all polynomial time adversaries have at most negligible advantage*

$$\mathbf{Adv}^{SDP}_{\Pi,\mathcal{A}}(1^\lambda, \Delta) = |\Pr[b' = b] - 1/2| \leq \mathtt{negl}(\lambda).$$

where $\mathtt{negl}(\lambda)$ *denotes a negligible function in* λ.

Single-Dimensional Privacy. Now let us define the last piece of privacy (*i.e., single-dimensional privacy*) in our design. Informally, single-dimensional privacy means given a search token of multi-dimensional range query Q, a computationally bounded adversary is not able to independently obtain the exact search results for any single-dimensional query Q^k, for $1 \leq k \leq n$, where Q^k denotes the single-dimensional query of Q in the k-th dimension. For instance, if $Q = ([30, 40] \wedge [400, 700])$, then $Q^1 = [30, 40]$ and $Q^2 = [400, 700]$.

In fact, we can actually capture an adversary's capability for attacking single-dimensional privacy *consistently* in the preceding selective security games we presented. Since we have both selective query security game and selective data security game, we need to particularly capture single-dimensional privacy for each of them. For the selective single-dimensional query security game, it is the same as the selective query security game in Definition 2 except that it has an additional ***third*** restriction for responding ciphertext queries, which can be rigorously defined as follows:

– *Ciphertext Query: the description is the same as in Definition 2 with:*
  1. $\mathcal{L}(\mathbf{D}_j, \Gamma_j, Q_0) = \mathcal{L}(\mathbf{D}_j, \Gamma_j, Q_1)$;
  2. *And for* $1 \leq i \leq n$, *either* $(D_{j,i} \in Q_0) \wedge (D_{j,i} \in Q_1)$, *or* $(D_{j,i} \notin Q_0) \wedge (D_{j,i} \notin Q_1)$;
  3. *And if* $(D_{j,i} \notin Q_0) \wedge (D_{j,i} \notin Q_1)$, *for some* $i \in [1, n]$, *there exists some* $k \in [1, w]$, *such that* $(D_{j,i} \in Q_0^k) \wedge (D_{j,i} \notin Q_1^k)$ *or* $(D_{j,i} \notin Q_0^k) \wedge (D_{j,i} \in Q_1^k)$.

The advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}_{\Pi,\mathcal{A}}^{SSDQP}(1^\lambda, \Delta) = |\Pr[b' = b] - 1/2|$.

Correspondingly, we can also define the selective single-dimensional data security game, which has an additional ***third*** restriction for responding token queries compared to Definition 3:

– *Token Query: the description is the same as in Definition 3 with :*
  1. $\mathcal{L}(\mathbf{D}_0, \Gamma_0, Q_j) = \mathcal{L}(\mathbf{D}_1, \Gamma_1, Q_j)$;
  2. *And for* $1 \leq i \leq n$, *either* $(D_{0,i} \in Q_j) \wedge (D_{1,i} \in Q_j)$, *or* $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$;
  3. *And if* $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$, *for some* $i \in [1, n]$, *there exists some* $k \in [1, w]$, *such that* $(D_{0,i} \in Q_j^k) \wedge (D_{1,i} \notin Q_j^k)$ *or* $(D_{0,i} \notin Q_j^k) \wedge (D_{1,i} \in Q_j^k)$.

The advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}_{\Pi,\mathcal{A}}^{SSDDP}(1^\lambda, \Delta) = |\Pr[b' = b] - 1/2|$.

We say that scheme Π is selectively single-dimensional secure if the advantages of any polynomial time adversary in both of the two preceding selective single-dimensional security games are at most negligible:

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{SSDQP}(1^\lambda, \Delta) \leq \mathtt{negl}(\lambda), \quad \mathbf{Adv}_{\Pi,\mathcal{A}}^{SSDDP}(1^\lambda, \Delta) \leq \mathtt{negl}(\lambda).$$

If we compare the additional third restriction with the second one in each corresponding game, we can observe that it is actually a *redundant* one considering the existence of the second restriction. It indicates that an adversary will not obtain additional advantages compared to previous security games in Definitions 2 and 3. Therefore, we have

Lemma 1. *If scheme Π is selectively query secure in Definition 2 and selectively data secure in Definition 3, it is also selectively single-dimensional secure.*

Note that some scheme [18] could also achieve selectively query and data secure, but with *stronger* restrictions (*e.g., for* $1 \le i \le n$ and $1 \le k \le w$, *either* $(D_{0,i} \in Q_j^k) \wedge (D_{1,i} \in Q_j^k)$, *or* $(D_{0,i} \notin Q_j^k) \wedge (D_{1,i} \notin Q_j^k)$), which inherently prevent it from achieving single-dimensional privacy [28]. That is why we emphasized with "in Definition 2" and "in Definition 3" in the above lemma.

6 Tree-Based Symmetric-Key MDRSE

Overview. The essential idea of our design is to utilize predicate encryption (more specifically, SSW [22]) to verify geometric relations, including whether a point is inside a hyper-rectangle and whether two hyper-rectangles intersect. As a result, a data owner can encrypt the nodes (i.e., points or hyper-rectangles) in an R-tree, then the cloud server can still operate the search algorithm of an R-tree correctly and privately in the ciphertext domain.

6.1 Geometric Relations on Encrypted Data

A ppoint Is Inside a Hyper-rectangle. Previous work [18] has proved that by using SSW, a primitive named *Range Predicate Encryption* can be built to verify whether a value d is inside a single dimension range $\mathbf{R}$, where the output will be 1 iff $d \in \mathbf{R}$. Specifically,

- $\texttt{RPE.Setup}(1^\lambda, T)$: *Given security parameter* λ *and* T, *output secret key* $\texttt{SK}$ *by running* $\texttt{SSW.Setup}(1^\lambda, T)$.
- $\texttt{RPE.Enc}(\texttt{SK}, d)$: *Given* $\texttt{SK}$ *and a value* d, *where* $d \in [1, T]$, *output ciphertext* C *by running* $\texttt{SSW.Enc}(\texttt{SK}, \boldsymbol{u})$, *where* $\boldsymbol{u} = (u_1, ..., u_T)$ *and*

$$u_i = 1, \text{ if } i = d; \qquad u_i = 0, \text{ otherwise.}$$

- $\texttt{RPE.GenToken}(\texttt{SK}, R)$: *Given* $\texttt{SK}$ *and a range* $\mathbf{R} = [x_l, x_r]$, *where* $\mathbf{R} \subseteq [1, T]$, *output token* $\texttt{TK}$ *by running* $\texttt{SSW.GenToken}(\texttt{SK}, \boldsymbol{v})$, *where* $\boldsymbol{v} = (v_1, ..., v_T)$ *and*

$$v_i = 0, \text{ if } i \in [x_l, x_r]; \qquad v_i = 1, \text{ otherwise.}$$

- $\texttt{RPE.Query}(\texttt{TK}, C)$: *Given* $\texttt{TK}$ *and* C, *output* 1 *or* 0 *by running* $\texttt{SSW.Query}(\texttt{TK}, C)$, *where output* 1 *iff* $\boldsymbol{u} \circ \boldsymbol{v} = 0$ *and output* 0 *otherwise.*

Based on this range predicate encryption, we can extend it into the multi-dimension in this paper and construct a *Point Predicate Encryption* to verify whether a point D is inside a hyper-rectangle $\mathbf{HR}$, where the output will be 1 iff $D \in \mathbf{HR}$. The correctness of this extension from the single dimension into the multi-dimension follows a simple geometric fact that *if a point is inside a hyper-rectangle, then the value of this point in* ***every*** *dimension will be inside the range of the corresponding single dimension, and vise versa:*

$$D \in \mathbf{HR} \quad \Leftrightarrow \quad \{d_k \in \mathbf{R}_k\}, \text{ for every } k \in [1, w],$$

where $D = (d_1, ..., d_w)$ *and* $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$. The details of this point predicate encryption[2] are presented as follows with an example in Fig. 4:

- $\texttt{PPE.Setup}(1^\lambda, \Delta)$: *Given security parameter* λ *and* $\Delta = \{T_1, ..., T_w\}$, *output secret key* SK *by running* $\texttt{SSW.Setup}(1^\lambda, wT)$.
- $\texttt{PPE.Enc}(\texttt{SK}, D)$: *Given* SK *and a point* $D = (d_1, ..., d_w)$, *where* $D \in \mathbb{L}_\Delta$, *output ciphertext* C *by running* $\texttt{SSW.Enc}(\texttt{SK}, \boldsymbol{u})$, *where* $\boldsymbol{u} = (u_1, ..., u_{wT})$ *and for* $1 \leq k \leq w$,

$$\begin{cases} u_i = 1, & \text{if } i = d_k + (k-1)T; \\ u_i = 0, & \text{otherwise.} \end{cases}$$

- $\texttt{PPE.GenToken}(\texttt{SK}, \mathbf{HR})$: *Given* SK *and a hyper-rectangle* $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$, *where* $\mathbf{HR} \subseteq \mathbb{L}_\Delta$ *and* $\mathbf{R}_k = [x_{k,l}, x_{k,r}]$, for $1 \leq k \leq w$, *output token* TK *by running* $\texttt{SSW.GenToken}(\texttt{SK}, \boldsymbol{v})$, *where* $\boldsymbol{v} = (v_1, ..., v_{wT})$ *and for* $1 \leq k \leq w$,

$$\begin{cases} v_i = 0, & \text{if } i \in [x_{k,l} + (k-1)T, x_{k,r} + (k-1)T]; \\ v_i = 1, & \text{otherwise.} \end{cases}$$

- $\texttt{PPE.Query}(\texttt{TK}, C)$: *Given* TK *and* C, *output* 1 *or* 0 *by running* $\texttt{SSW.Query}(\texttt{TK}, C)$, *where output* 1 *iff* $\boldsymbol{u} \circ \boldsymbol{v} = 0$ *and output* 0 *otherwise.*

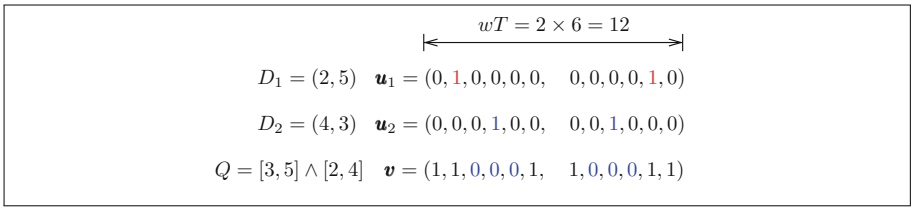

Fig. 4. An example of point predicate encryption with $w = 2$ and $T = 6$, where $D_1 \notin Q$ due to $\boldsymbol{v} \circ \boldsymbol{u}_1 \neq 0$; $D_2 \in Q$ due to $\boldsymbol{v} \circ \boldsymbol{u}_2 = 0$.

Two Hyper-rectangles Intersect. We can also verify whether two hyper-rectangles intersect on encrypted data by using SSW. Similar as the preceding geometric relation, we can first start with the simplest case (*i.e., the case in the single dimension*) by deciding whether two ranges intersect. Specifically, we can build a *Range Intersection Predicate Encryption*, where the output is 1 iff two ranges intersect (i.e., $\mathbf{R} \cap \mathbf{R}' \neq \varnothing$). The correctness in testing the intersection of two ranges is based on the following equivalent geometric relation:

$$\mathbf{R} \cap \mathbf{R}' \neq \varnothing \Leftrightarrow \begin{cases} x_l \in [1, x'_r]; \\ x_r \in [x'_l, T] \end{cases} \Leftrightarrow \begin{cases} x_l \in [1, x'_r]; \\ (x_r + T) \in [x'_l + T, 2T] \end{cases}$$

where $\mathbf{R} = [x_l, x_r]$ and $\mathbf{R}' = [x'_l, x'_r]$. The description in the *third column* above is trivial from the one in the middle column, but it will help readers follow the details of the following algorithms more easily. The details of the range intersection predicate encryption are as below:

[2] For the ease of description, we assume each dimension has the same size (i.e., $T_k = T$, for every $k \in [1, w]$) in the following algorithms..

- $\texttt{RIPE.Setup}(1^\lambda, T)$: *Given security parameter* λ *and* T*, output secret key* $\texttt{SK}$ *by running* $\texttt{SSW.Setup}(1^\lambda, 2T)$.
- $\texttt{RIPE.Enc}(\texttt{SK}, \mathbf{R})$: *Given* $\texttt{SK}$ *and a range* $\mathbf{R} = [x_l, x_r]$*, where* $\mathbf{R} \subseteq [1, T]$*, output ciphertext* C *by running* $\texttt{SSW.Enc}(\texttt{SK}, \boldsymbol{u})$*, where* $\boldsymbol{u} = (u_1, ..., u_{2T})$ *and*

$$\begin{cases} u_i = 1, & \text{if } i = x_l \text{ or } i = x_r + T; \\ u_i = 0, & \text{otherwise.} \end{cases}$$

- $\texttt{RIPE.GenToken}(\texttt{SK}, \mathbf{R}')$: *Given* $\texttt{SK}$ *and a range* $\mathbf{R}' = [x'_l, x'_r]$*, where* $\mathbf{R}' \subseteq [1, T]$*, output token* $\texttt{TK}$ *by running* $\texttt{SSW.GenToken}(\texttt{SK}, \boldsymbol{v})$*, where* $\boldsymbol{v} = (v_1, ..., v_{2T})$ *and*

$$\begin{cases} v_i = 0, & \text{if } i \in [1, x'_r] \text{ or } i \in [x'_l + T, 2T] \\ v_i = 1, & \text{otherwise.} \end{cases}$$

- $\texttt{RIPE.Query}(\texttt{TK}, C)$: *Given* $\texttt{TK}$ *and* C*, output* 1 *or* 0 *by running* $\texttt{SSW.Query}(\texttt{TK}, C)$*, where output* 1 *iff* $\boldsymbol{u} \circ \boldsymbol{v} = 0$ *and output* 0 *otherwise.*

With this range intersection predicate encryption, we can further extend it into the multi-dimension case to design a *Hyper-rectangle Intersection Predicate Encryption*, which can test whether two hyper-rectangles intersect on encrypted data. The correctness of this extension also follows a simple geometric fact that *if a hyper-rectangle intersects with another hyper-rectangle, then the range of the first hyper-rectangle in* ***every*** *dimension intersects the corresponding range of the second hyper-rectangle, and visa versa:*

$$\mathbf{HR} \cap \mathbf{HR}' = \varnothing \Leftrightarrow \{\mathbf{R}_k \cap \mathbf{R}'_k = \varnothing\} \Leftrightarrow \begin{cases} x_{k,l} \in [1, x'_{k,r}], \\ x_{k,r} \in [x'_{k,l}, T] \end{cases}$$

for every $k \in [1, n]$*, where* $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_n)$*,* $\mathbf{HR}' = (\mathbf{R}'_1, ..., \mathbf{R}'_n)$*,* $\mathbf{R}_k = [x_{k,l}, x_{k,r}]$ *and* $\mathbf{R}'_k = [x'_{k,l}, x'_{k,r}]$. The details of this hyper-rectangle intersection predicate encryption are presented as below:

- $\texttt{HIPE.Setup}(1^\lambda, \Delta)$: *Given security parameter* λ *and* $\Delta = (T_1, ..., T_w)$*, output secret key* $\texttt{SK}$ *by running* $\texttt{SSW.Setup}(1^\lambda, 2wT)$.
- $\texttt{HIPE.Enc}(\texttt{SK}, \mathbf{HR})$: *Given* $\texttt{SK}$ *and a hyper-rectangle* $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$*, where* $\mathbf{HR} \subseteq \mathbb{L}_\Delta$ *and* $\mathbf{R}_k = [x_{k,l}, x_{k,r}]$*, for* $1 \le k \le w$*, output ciphertext* C *by running* $\texttt{SSW.Enc}(\texttt{SK}, \boldsymbol{u})$*, where* $\boldsymbol{u} = (u_1, ..., u_{2wT})$ *and for* $1 \le k \le w$*,*

$$\begin{cases} u_i = 1, & \text{if } i = x_{k,l} + (2k-2)T \text{ or } i = x_{k,r} + (2k-1)T; \\ u_i = 0, & \text{otherwise.} \end{cases}$$

- $\texttt{HIPE.GenToken}(\texttt{SK}, \mathbf{HR}')$: *Given* $\texttt{SK}$ *and a hyper-rectangle* $\mathbf{HR}' = (\mathbf{R}'_1, ..., \mathbf{R}'_w)$*, where* $\mathbf{HR}' \subseteq \mathbb{L}_\Delta$ *and* $\mathbf{R}'_k = [x'_{k,l}, x'_{k,r}]$, for $1 \le k \le w$*, output token* $\texttt{TK}$ *by running* $\texttt{SSW.GenToken}(\texttt{SK}, \boldsymbol{v})$*, where* $\boldsymbol{v} = (v_1, ..., v_{2wT})$ *and for* $1 \le k \le w$*,*

$$\begin{cases} v_i = 0, & \text{if } i \in [1 + (2k-2)T, \quad x'_{k,r} + (2k-2)T] \\ & \text{or } i \in [x'_l + (2k-1)T, \quad 2kT]; \\ v_i = 1, & \text{otherwise.} \end{cases}$$

- $\texttt{HIPE.Query}(\texttt{TK}, C)$: *Given* $\texttt{TK}$ *and* C*, output* 1 *or* 0 *by running* $\texttt{SSW.Query}(\texttt{TK}, C)$*, where output* 1 *iff* $\boldsymbol{u} \circ \boldsymbol{v} = 0$ *and output* 0 *otherwise.*

Since these predicate encryptions presented above are the extensions of SSW, the security of them can be easily proved based on the security of SSW.

6.2 Elm: Full Scheme

With R-trees and the preceding predicate encryptions (extended from SSW), we build Elm, a tree-based symmetric-key MDRSE. Basically, our scheme follows the definition we described in Sect. 4. A data owner will first generate a secret key in `GenKey` and build an R-tree based on its data records in `BuildTree`. Then, the data owner encrypts all the nodes in the R-tree in `Enc` and outsources the encrypted tree to the cloud. Specifically, each leaf node is encrypted with point predicate encryption and each non-leaf node is encrypted with hyper-rectangle intersection predicate encryption.

Given a multi-dimensional range query, the data owner is able to compute a search token in `GenToken`. Each search token contains two sub-tokens: one (i.e., $\mathtt{TK}_{leaf}$) is for testing whether a leaf node is inside the multi-dimensional range query, and another one (i.e., $\mathtt{TK}_{nleaf}$) is for checking whether a non-leaf node intersects with the multi-dimensional range query. Finally, the cloud server returns the identifiers of all the matched results to the data owner by running `Search`. The details of Elm are presented as follows.

- $\mathtt{GenKey}(1^\lambda, \Delta)$: Given a security parameter λ and $\Delta = (T_1, ..., T_w)$, the data owner computes a secret key $\mathbf{SK} = \{\mathtt{SK}_{leaf}, \mathtt{SK}_{nleaf}\}$, where

$$\mathtt{SK}_{leaf} \leftarrow \mathtt{PPE.Setup}(1^\lambda, \Delta), \quad \mathtt{SK}_{nleaf} \leftarrow \mathtt{HIPE.Setup}(1^\lambda, \Delta).$$

- $\mathtt{BuildTree}(\mathbf{D})$: Given a set of data records $\mathbf{D} = (D_1, ..., D_n)$, where $D_i = (d_{i,1}, ..., d_{i,w})$, for $1 \leq i \leq w$, the data owners builds an R-tree $\Gamma = \{D_1, ..., D_n, N_1, ..., N_m, \mathbf{P}\}$, where D_i is a leaf node, for $1 \leq i \leq n$, N_j is a non-leaf node, for $1 \leq j \leq m$, and $\mathbf{P}$ is a set of pointers covering all the parent-child relations in tree Γ.
- $\mathtt{Enc}(\mathbf{SK}, \Gamma)$: Given $\mathbf{SK}$ and Γ, the data owner encrypts every leaf node and every non-leaf node respectively:

$$C_i \leftarrow \mathtt{PPE.Enc}(\mathtt{SK}_{leaf}, D_i), \ \textit{for } 1 \leq i \leq n;$$
$$E_j \leftarrow \mathtt{HIPE.Enc}(\mathtt{SK}_{nleaf}, N_j), \ \textit{for } 1 \leq j \leq m.$$

  Then, the data owner generates and outsources the encrypted R-tree $\Gamma^* = \{C_1, ..., C_n, E_1, ..., E_m, \mathbf{P}\}$, to the cloud server, where C_i is an encrypted leaf node, for $1 \leq i \leq n$, E_j is an encrypted non-leaf node, for $1 \leq j \leq m$.
- $\mathtt{GenToken}(\mathbf{SK}, Q)$: Given $\mathbf{SK}$ and a multi-dimensional range query Q, the data owner computes a token $\mathbf{TK} = \{\mathtt{TK}_{leaf}, \mathtt{TK}_{nleaf}\}$, where

$$\mathtt{TK}_{leaf} \leftarrow \mathtt{PPE.GenToken}(\mathtt{SK}_{leaf}, Q), \quad \mathtt{TK}_{nleaf} \leftarrow \mathtt{HIPE.GenToken}(\mathtt{SK}_{nleaf}, Q).$$

- $\mathtt{Search}(\Gamma^*, \mathbf{TK})$: Given Γ^* and $\mathbf{TK} = (\mathtt{TK}_{leaf}, \mathtt{TK}_{nleaf})$, the cloud server searches as follows by starting from the root node of tree Γ^*:

  – If it is non-leaf node E_j, $\mathtt{Flag}_{nleaf} = \mathtt{HIPE.Query}(\mathtt{TK}_{nleaf}, E_j)$. If $\mathtt{Flag}_{nleaf} = 1$, continues to search the child nodes of this non-leaf node based on $\mathbf{P}$; otherwise, stops searching the child nodes.
  – If it is leaf node C_i, $\mathtt{Flag}_{leaf} = \mathtt{PPE.Query}(\mathtt{TK}_{leaf}, C_i)$. If $\mathtt{Flag}_{leaf} = 1$, returns the identifier I_i of this leaf node; otherwise, does not return the identifier.

  Finally, the cloud server returns a set $\mathbf{I}$ of identifiers, where $I_i \in \mathbf{I}$, if $D_i \in Q$.

Correctness. Since the search process in an encrypted R-tree is actually several search paths from the root node to several matched leaf nodes, the correctness of it depends on the correctness at each node in these paths. Informally, because the cloud server is able to correctly test the geometric relation at each node based on the correctness of SSW, which is the building block, the entire search process in Elm is correct. Due to the space limitation, detailed explanations about the correctness of Elm are presented in our technical report [29].

Efficiency. Since the search algorithm in Elm exactly follows the original search algorithm of an R-tree in the plaintext domain, the complexity of the search algorithm in Elm is *faster-than-linear* regarding to the number of data records n. Based on the complexity of SSW in [22], our design introduces $O(wT)$ overhead in *secret key size*, $O(wT)$ overhead in *encryption time, ciphertext size and search time* at each node, and $O(wT)$ overhead in *token size and token generation time* for each given query, where w is the number of dimensions and T is the size of each dimension.

Security Analysis. We now analyze the security of Elm, including query privacy, data privacy and single-dimensional privacy.

Theorem 1 *(Selective Query Privacy). Elm is selectively query secure, if SSW is selectively query secure.*

Proof. From the high-level, the proof of this theorem can be analyzed with two aspects. First, because Elm is essentially a scheme with multiple instances of SSW (i.e., one instance per node in the tree) and SSW is a probabilistic encryption with selective query security, therefore, Elm is selectively query secure (according to the claim in Chap. 3 in [16] that *any probabilistic symmetric-key encryption scheme that is secure under chosen-plaintext attacks automatically implies the multiple encryption of it is secure under chosen-plaintext attacks*).

Second, the inherent leakage of path pattern in R-trees, which we used in Elm, do not reveal additional information based on what we defined in Definition 2 (according to recent observation [28], compared to R-trees, the inherent leakage of path pattern in other similar trees, such as kd-trees and range trees, inevitably reveal additional information, particularly in single dimensions, which will fail to satisfy the restrictions in Definition 2). Due to the space limitation, the detailed proof of this selective query privacy of Elm following Definition 2 can be found in [29].

Theorem 2 *(Selective Data Privacy). Elm is selectively data secure, if SSW is selectively data secure.*

Proof. The selective data privacy of Elm can be proved in a similar way as in Theorem 1. See details in our technical report [29].

Theorem 3 *(Single-Dimensional Privacy). Elm is selectively single-dimensional secure, if Elm is selectively query secure in Theorem 1 and selectively data secure in Theorem 3.*

Proof. Based on Lemma 1 in Sect. 5.

6.3 Dynamic Data

As we mentioned at the beginning of this paper, another benefit from enhancing query privacy compared to [28] is that, our scheme is able to support *dynamic data* in an encrypted R-tree without revealing updated data records. More specifically, in order to update a data record (either add a new one or delete an existing one) in an R-tree, the data owner needs to first submit an *update query* (based on the content of this updated data record) to locate which part of the tree should be updated accordingly. Without protecting query privacy in [28], the cloud server will directly learn the updated data record through this update query, which is clearly not secure for dynamic data. While with our scheme, we can still securely decide which part of the tree should be updated without revealing the updated data record. Basically, we can achieve this objective by still leveraging an extension of SSW similarly as the preceding use of point predicate encryption in the design of Elm. Due to space limitations, how to particularly support dynamic operations, including insert, delete and modify, over encrypted data are presented in our technical report [29].

Unfortunately, so far, *secure update* in Elm can only work with some simple cases, where assuming one update operation only introduces one node update in the tree. The reason is that the cases with updates at several nodes in an R-tree could be more complicated and challenging on encrypted data (more details about update algorithms of R-trees in the plaintext domain can be found in [19]). For example, in some cases, one update may need to "split" one bounding box (a non-leaf node) into two new ones while the splitting process requires evaluation and comparison of distances among data/nodes. Considering our scheme cannot compute or compare distance on encrypted data, Elm cannot directly support this splitting process for complicated update. Of course, this type of computation and comparison of distances on encrypted data can be evaluated with additional use of other cryptographic approaches, such as Asymmetric Scalar-product Preserving Encryption used in k-nearest neighbor search [32] or Secure Two-Party Computation with two non-colluding servers [9]. However, the naive combination of these methods with Elm will make the entire scheme more complicated and cumbersome. More importantly, computation and comparison of distances, especially in a tree, will reveal much more additional privacy to the public cloud compared to the current privacy leakage defined in the leakage function, which needs to be rigorously defined and studied in the future.

7 Performance

In this section, we evaluate the performance of Elm, especially the search performance. We use Pairing-Based Cryptography (PBC) Library to simulate the cost on cryptographic operations in the following experiments. We test them in Ubuntu 12.04 with Intel Core i5 3.30 GHz Processor and 2 GB Memory.

We first evaluate the search time at a leaf node or a non-leaf node in Figs. 5 and 6. As we discussed before, the complexity at each node is $O(wT)$. Clearly, the search time over encrypted data at a leaf node or a non-leaf node is linearly

increasing with the number of dimensions w or the size of each dimension T. According to the details of SSW [22], the dominating cryptographic operations at each node in the evaluation are pairing operations. The average time of evaluating one pairing operation (tested on super-singular curve $y^2 = x^3 + x$ with preprocessing in PBC) in our experiments is around 2.28 milliseconds.

Next, we demonstrate the search efficiency of our scheme is indeed faster-than-linear regarding to the number of data records n. To do this, we have a basic scheme, which only encrypts every data record with point predicate encryption in Sect. 6 (imaging an incomplete version of Elm without any non-leaf nodes), and compare its search efficiency with Elm. Since there is no non-leaf nodes in the basic scheme to index data, the search algorithm of this basic scheme has to check every encrypted data record one-by-one (i.e., linear complexity). To simulate the performance of Elm, we run the search code of an R-tree in the plaintext domain with multiple random queries, but we sleep the search process at each node in the tree for a certain time, which is equivalent to the computation time for evaluating the corresponding geometric relation on encrypted data.

The comparison of this basic scheme and Elm is tested based on a part of a real-world dataset (U.S. census 1990 [2]) and is presented in Fig. 7 and Table 2. We can see from the table and Fig. 7 that Elm is much faster than the basic one for handing multi-dimensional range queries. Specifically, when $n = 100,000$, the basic solution requires $46,056$ s while Elm only needs $4,236$ s in average to operate search on encrypted data.

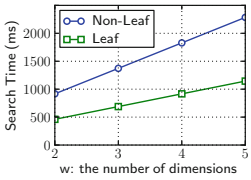

Fig. 5. Impact of w on search time (millisecond) at each node with $T = 50$.

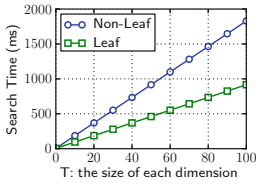

Fig. 6. Impact of T on search time (millisecond) at each node with $w = 2$.

We can also see that, in order to protect users' privacy, the performance of Elm on encrypted data is around 2×10^5 times slower than the one in plaintext.

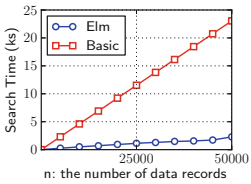

Fig. 7. Impact of n on search time (kilosecond) with $w = 2$ and $T = 50$.

Since Elm has the same complexity (regarding to the number of data records n) as an R-tree in the plaintext domain, this performance gap is mainly introduced by the $O(wT)$ pairing operations on each node. One of our future work is to study how to minimize this gap via lightweight primitives without relying on pairing operations while still preserving a same or similar level of privacy.

Table 2. Average Search Time when $w = 2$ and $T = 50$.

n	Basic (second)	Elm (second)	R-tree in Plaintext (millisecond)
1,000	461	71	0.37
10,000	4,606	515	2.34
100,000	46,056	4,236	18.42

8 Conclusion and Future Work

We design a tree-based symmetric-key MDRSE in this paper to achieve faster-than-linear search, data privacy, query privacy and single-dimensional privacy for multi-dimensional range queries on encrypted data. We demonstrate the security and efficiency of the proposed scheme through rigorous analyses and experiments. For our future work, we will focus on achieving secure *fully dynamic data* operations in a tree-based multi-dimensional searchable encryption.

Acknowledgement. We would like to thank the reviewers for providing many useful comments. This work was supported in part by the US National Science Foundation under grant CNS-1218085, NSF of China 61272457, National Project 2012ZX03002003-002, 863 Project 2012AA013102, 111 Project B08038, IRT 1078, FRF K50511010001 and NSF of China 61170251.

References

1. http://www.wuala.com/
2. http://archive.ics.uci.edu/ml/datasets.html

3. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
4. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
6. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
7. Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M.C., Steiner, M.: Dynamic searchable encryption in very-large databases: data structures and implementation. In: Proceedings of NDSS 2014 (2014)
8. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013)
9. Chun, H., Elmehdwi, Y., Li, F., Bhattacharya, P., Jiang, W.: Outsourceable two-party privacy-preserving biometric authentication. In: Proceedings of ACM ASIACCS 2014 (2014)
10. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of ACM CCS 2006 (2006)
11. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM **43**(3), 431–473 (1996)
12. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
13. Guttman, A.: R-Trees: a dynamic index structure for spatial searching. In: Proceedings of ACM SIGMOD 1984 (1984)
14. Kamara, Seny, Papamanthou, Charalampos: Parallel and dynamic searchable symmetric encryption. In: Sadeghi, Ahmad-Reza (ed.) FC 2013. LNCS, vol. 7859, pp. 258–274. Springer, Heidelberg (2013)
15. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of ACM CCS 2012, pp. 965–976 (2012)
16. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, Boca Raton (2007)
17. Lai, J., Zhou, X., Deng, R.H., Li, Y., Chen, K.: Expressive search on encrypted data. In: Proceedings of ACM ASIACCS 2013, pp. 243–251 (2013)
18. Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In: Proceedings of NDSS 2012 (2012)
19. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A.N., Theodoridis, Y.: R-Trees: Theory and Applications. Advanced Information and Knowledge Processing. Springer, London (2006)
20. Pappas, V., Krell, F., Vo, B., Kolesnikov, V., Malkin, T., Choi, S.G., George, W., Keromytis, A., Bellovin, S.: Blind seer: a searchable private DBMS. In: Proceedings of IEEE S&P 2014 (2014)
21. Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. IEEE Internet Comput. **16**(1), 69–73 (2012)

22. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
23. Shi, E., Bethencourt, J., Chan, T.H.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Proceedings of IEEE S&P 2007, pp. 350–364 (2007)
24. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of IEEE S&P 2000 (2000)
25. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path ORAM: an extremely simple oblivious RAM protocol. In: Proceedings of ACM CCS 2013 (2013)
26. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: Proceedings of NDSS 2014 (2014)
27. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of ACM AISACCS 2013 (2013)
28. Wang, B., Hou, Y., Li, M., Wang, H., Li, H.: Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In: Proceedings of ACM ASIACCS 2014 (2014)
29. Wang, B., Hou, Y., Li, M., Wang, H., Li, H., Li, F.: Tree-based multi-dimensional range search on encrypted data with enhanced privacy. Technical report, Utah State University (2014). http://digital.cs.usu.edu/~mingli/tech/elm14.pdf
30. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: Proceedings of ICDCS 2010 (2010)
31. Wang, P., Ravishankar, C.V.: Secure and efficient range queries on outsourced databases using R-trees. In: Proceedings of IEEE ICDE 2013 (2013)
32. Wong, W.K., Cheung, D.W., Kao, B., Mamoulis, N.: Secure kNN compuation on encrypted databases. In: Proceedings of SIGMOD 2009 (2009)
33. Wong, W.K., Kao, B., Cheung, D.W., Li, R., Yiu, S.M.: Secure query processing with data interoperability in a cloud database environment. In: Proceedings of ACM SIGMOD 2014 (2014)