Maple: Scalable Multi-Dimensional Range Search over Encrypted Cloud Data with Tree-based Index

Boyang Wang*
Dept. of Computer Science
Utah State University
Logan, UT, 84322
xd.bywang@gmail.com

Yantian Hou
Dept. of Computer Science
Utah State University
Logan, UT, 84322
houyantian@gmail.com

Ming Li
Dept. of Computer Science
Utah State University
Logan, UT, 84322
ming.li@usu.edu

Haitao Wang
Dept. of Computer Science
Utah State University
Logan, UT, 84322
haitao.wang@usu.edu

Hui Li
State Key Laboratory of ISN
Xidian University
Xi'an, Shaanxi, 710071, China
lihui@mail.xidian.edu.cn

ABSTRACT

Cloud computing promises users massive scale outsourced data storage services with much lower costs than traditional methods. However, privacy concerns compel sensitive data to be stored on the cloud server in an encrypted form. This posts a great challenge for effectively utilizing cloud data, such as executing common SQL queries. A variety of searchable encryption techniques have been proposed to solve this issue; yet efficiency and scalability are still the two main obstacles for their adoptions in real-world datasets, which are multi-dimensional in general. In this paper, we propose a tree-based public-key Multi-Dimensional Range Searchable Encryption (MDRSE) to overcome the above limitations. Specifically, we first formally define the leakage function and security of a tree-based MDRSE. Then, by leveraging an existing predicate encryption in a novel way, our tree-based MDRSE efficiently indexes and searches over encrypted cloud data with multi-dimensional tree structures (i.e., R-trees). Moreover, our scheme is able to protect single-dimensional privacy while previous efficient solutions fail to achieve. Our scheme is selectively secure, and through extensive experimental evaluation on a large-scale real-world dataset, we show the efficiency and scalability of our scheme.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems; H.3 [**Information Search and Retrieval**]: Information Search and Retrieval

*Boyang Wang is also a member of State Key Laboratory of ISN, Xidian University.

ASIA CCS'14, June 4–6, 2014, Kyoto, Japan.

http://dx.doi.org/10.1145/2590296.2590305.

Keywords

Multiple dimension; Range search; Encrypted cloud data; Tree structures

1. INTRODUCTION

With the adoption of cloud computing, data owners can reap huge economic benefits by outsourcing their data to the cloud instead of storing their data locally. Due to the serious privacy concerns in the cloud, sensitive data, such as financial or medical records, should be encrypted before being outsourced to the cloud server [2]. As a result, the cloud server is not allowed to reveal the content of outsourced data unless it possesses proper decryption keys. Unfortunately, the encryption on outsourced data inevitably introduces new challenges to data utilization. Specifically, how to enable users to effectively search over encrypted data as they generally perform in the plaintext domain, is one of the most significant issues needed to be solved in the cloud.

To enable different search functions over encrypted data, many Searchable Encryption (SE) schemes have been proposed. However, most of the existing SE schemes [4, 7, 12, 14, 15, 19, 20, 24, 29, 30, 32, 33, 35] are only able to handle rather simple queries, such as keyword search or single-dimensional range queries, which are not able to well support *multi-dimensional range queries* over encrypted data. Considering real-world datasets are often multi-dimensional while common SQL queries (such as equality and comparison) can all be flexibly expressed as range queries [28], we believe designing an efficient Multi-Dimensional Range Searchable Encryption (MDRSE) will be an important step forward to make searchable encryption practical and scalable on large datasets in a real-world cloud setting.

Boneh and Waters [9] proposed a predicate encryption, named Hidden Vector Encryption, which can be utilized to support multi-dimensional range queries over encrypted data. Meanwhile, Shi et al. [28] designed an encryption scheme to handle multi-dimensional range queries as well. These two schemes are both public-key approaches. Unfortunately, the straightforward applications of these two schemes can only achieve linear search time with regard to the total number of data records. Since the number of data records in a real-world dataset is generally very large (e.g.

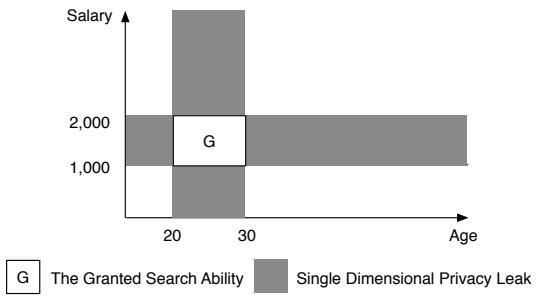

Figure 1: An example of privacy leak in LSED$^+$ [25].

in the order of millions or larger), the straightforward applications of these two schemes are clearly not practical.

Moving a step forward, Lu [25] designed a symmetric-key single dimensional range search scheme (named LSED) on encrypted data within *logarithmic* search time, and mentioned a direct extension of it (denoted as LSED$^+$) in the multi-dimension by achieving *sublinear* search time. However, this direct extension, which decomposes each multi-dimensional search token into independent ones in each single dimension, will result in significant privacy leak in every single dimension [25]. For instance, as described in Fig. 1, given the search token of a two-dimensional range query $Q = (age \in [20, 30]) \wedge (salary \in [1,000, 2,000])$ in LSED$^+$, the cloud server is able to independently search the corresponding results to a single dimensional range query $Q^x = (age \in [20, 30])$ on X-dimension or $Q^y = (salary \in [1,000, 2,000])$ on Y-dimension, which clearly reveals more private information to the cloud server than it is allowed to. Such type of leakage is referred to as *single-dimensional privacy* in this paper.

The recent work [34] proposed by Wang et al. also has privacy leakage in single dimensions for the same reason as LSED$^+$, although its efficiency is faster than linear search as well. Cash et al. [13] recently designed a sublinear SE scheme supporting conjunctive keyword search, which can protect privacy in single dimensions. However, how to extend this work to efficiently support multi-dimensional range queries is still unknown. Therefore, how to design an efficient and single-dimensionally private MDRSE remains open.

In this paper, we endeavor to solve the above problem by proposing Maple, a *tree-based* public-key MDRSE. By utilizing multi-dimensional tree structures (specifically, R-trees [17]) to index data records, we can achieve faster-than-linear search efficiency with respect to the number of data records[1]. Obviously, this goal is easy to achieve in the plaintext domain utilizing various multi-dimensional tree structures [5, 6, 17]. However, when data records and index are all encrypted, the main design challenge is how to preserve faster-than-linear search complexity while not sacrificing too much privacy (including single-dimensional privacy), which is not achieved by recent solutions [25, 34]. A high-level comparison among our scheme and previous solutions is presented in Table 1. Our main contributions are:

(1) We formally define the privacy leakage function, selective security and single-dimensional privacy of a tree-based public-key MDRSE in this paper. Particularly, we are among the first to distinguish *path pattern* from traditional access pattern and include path pattern in the privacy leakage function of a tree-based searchable encryption; we also introduce the necessary requirement of *isomorphic* tree structures in the security of a tree-based searchable encryption.

[1]R-trees do not guarantee a good *worst-case* performance [17,18], but they generally perform well with real-world data. Therefore, in the rest of this paper, when we mention R-trees achieve faster-than-linear search, we mean it is under an *empirical* sense.

	Faster than Linear Search?	Single-Dimensional Privacy
Shi et al. [28]	×	√
Boneh et al. [9]	×	√
LSED$^+$ [25]	√	×
Wang et al. [34]	√	×
Our Scheme	√	√

Table 1: Comparison among Different Solutions.

(2) By leveraging Hidden Vector Encryption [9] in a novel way, we can test geometric relations (including whether a point is inside a hyper-rectangle or two hyper-rectangles intersect) over encrypted data. By doing this, we can design a tree-based public-key MDRSE, which is able to efficiently support multi-dimensional range queries. Since our scheme is able to exactly follow the search algorithm of an R-tree in the plaintext domain while all the nodes of the tree are encrypted, our scheme can achieve the same search complexity (i.e., faster-than-linear search with regard to the number of data records) as a traditional R-tree in the plaintext domain. Moreover, the security analyses show that our scheme is selectively secure and single-dimensionally private.

(3) We observe that, in order to achieve single-dimensional privacy during the design of a tree-based MDRSE, we should be very careful about not only what type of tree structure is leveraged, but also which kind of predicate encryption is used. That is why the recent work in [34] is also based on R-trees as ours, but it cannot achieve single-dimensional privacy. On the other hand, some tree structures, such as kd-trees [5] and range trees [6], are faster-than-linear search as well. However, these trees cannot achieve single-dimensional privacy due to the nature of their tree structures (explainations about this observation are presented in Sec. 5).

2. PROBLEM DEFINITION

System Model. As shown in Fig. 2, the system model of our scheme includes two entities: a data owner and the cloud server. The data owner wants to outsource its database (i.e., a sets of data records) to the cloud server in order to reduce the storage cost on its local device. For example, $(25, 170, 2000)$ is one of the data records described in Fig. 2. Besides outsourcing storage, the data owner also wants to use its own data correctly. Specifically, the data owner should be able to retrieve the correct results from the outsourced data records by submitting a range query (e.g., $Q = [30, 40] \wedge [160, 185] \wedge [3000, 7000]$).

As in previous works [13, 15, 19, 20], the cloud server is assumed to be an *honest-but-curious* party. It means the cloud server can provide reliable data storage (will ensure the integrity of data [3]) and search services (will return the correct search results by following the protocols), but it is curious about the content of data records that are outsourced by the data owner.

Definition of MDRSE. Since we focus on handle multi-dimensional data in this paper, we first present some basic definitions in the multi-dimension, including lattices, points, and hyper-rectangles [28]:

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 $\mathbf{X}$ in $\mathbb{L}_\Delta$ is defined as $\mathbf{X} = (x_1, ..., x_w)$, where $x_i \in [T_i]$, $\forall i \in [1, w]$.*

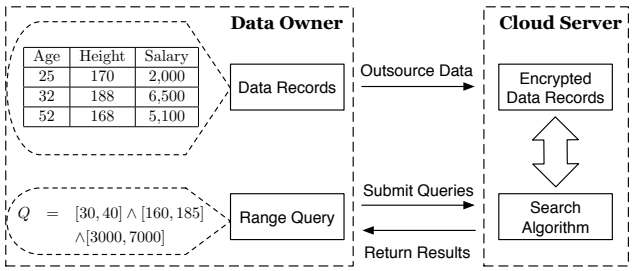

Figure 2: The system model includes a data owner and the cloud server.

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,* $\mathbf{R}_i \subseteq [1, T_i]$*,* $\forall i \in [1, w]$*.*

It is easy to see that, in the aforementioned system model, a data record is essentially a point and a range query is actually a hyper-rectangle. Next, we introduce the definition of public-key MDRSE. Since we intend to improve search efficiency, we introduce a multi-dimensional tree structure Γ in the following definition to index data.

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

- $(\mathtt{PK}, \mathtt{SK}) \leftarrow \mathtt{GenKey}(1^\lambda, \Delta)$*: 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 public key* PK *and a secret key* SK*.*
- $\Gamma \leftarrow \mathtt{BuildTree}(\mathbf{D})$*: is a deterministic algorithm run by the data owner to build a multi-dimensional tree structure 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 structure* Γ*.*
- $(\Gamma^*, \mathbf{C}) \leftarrow \mathtt{Enc}(\mathtt{PK}, \Gamma, \mathbf{M})$*: is a probabilistic algorithm run by the data owner to encrypt a multi-dimensional tree structure and messages. It takes as input a public key* PK*, multi-dimensional tree structure* Γ *and* n *messages* $\mathbf{M} = \{M_1, ..., M_n\}$*, where message* $M_i \in \mathcal{M}$ *is for each data record* D_i *and* $\mathcal{M}$ *is the message space, and outputs an encrypted multidimensional tree structure* Γ^* *and* n *ciphertexts* $\mathbf{C} = \{C_1, ..., C_n\}$*.*
- $\mathtt{TK}_Q \leftarrow \mathtt{GenToken}(\mathtt{SK}, Q)$*: 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* $\mathtt{TK}_Q$*.*
- $\mathbf{I}_Q \leftarrow \mathtt{Search}(\Gamma^*, \mathbf{C}, \mathtt{TK}_Q)$*: is a deterministic algorithm run by the server to search over an encrypted multi-dimensional tree structure and ciphertexts. It takes as input an encrypted multi-dimensional data structure* Γ^**,* n *ciphertexts* $\mathbf{C} = \{C_1, ..., C_n\}$ *and a search token* $\mathtt{TK}_Q$*, and outputs a set* $\mathbf{I}_Q$ *of identifiers, where* I_i *is the identifier (e.g., a memory location in the cloud server) of data record* D_i*, and* $I_i \in \mathbf{I}_Q$ *if the corresponding data record* $D_i \in Q$*.*

Correctness. *We say that the above tree-based public-key MDRSE scheme is correct if for all* $\lambda \in \mathbb{N}$*, all* $(\mathtt{PK}, \mathtt{SK})$ *output by* $\mathtt{GenKey}(1^\lambda, \Delta)$*, all* $D_i \in \mathbb{L}_\Delta$*, all* Γ *output by* $\mathtt{Buildtree}(\mathbf{D})$*, for all* $M_i \in \mathcal{M}$*, all* $(\Gamma^*, \mathbf{C})$ *output by* $\mathtt{Enc}(\mathtt{PK}, \Gamma, \mathbf{M})$*, all* $Q \subseteq \mathbb{L}_\Delta$*, all* $\mathtt{TK}_Q$ *output by* $\mathtt{GenToken}(\mathtt{SK}, Q)$*, for any* $i \in [1, n]$

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

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

In the above scheme, the tree structures of Γ and its encrypted version Γ^* are *isomorphic* (denoted as $\Gamma \simeq \Gamma^*$). The main difference is that all the nodes (including non-leaf nodes and leaf nodes) in Γ^* are encrypted compared to the nodes in Γ. An example of Γ and Γ^* is shown in Fig. 3.

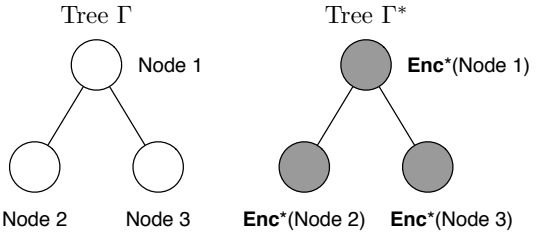

Figure 3: An example of Γ **and** Γ^***, where** $\Gamma \simeq \Gamma^*$ **and** $\mathtt{Enc}^*$ **is assumed as a probabilistic encryption algorithm performed on each node.**

As pointed out by [10], since the main objective of a searchable encryption is to test whether a data record D_i is satisfying a query Q (i.e., $D_i \in Q$ in this paper) without revealing D_i, the message space $\mathcal{M}$ can be set to a singleton, such as $\mathcal{M} = \{\mathtt{Flag}\}$. Then, in $\mathtt{Search}(\Gamma^*, \mathbf{C}, \mathtt{TK}_Q)$, the algorithm will reveal a string Flag for each ciphertext C_i if data record $D_i \in Q$, and return the identifier I_i of this data record D_i. A larger message space of $\mathcal{M}$ can be used if more information need to be unlocked when $D_i \in Q$ [9].

Moreover, as emphasized in [10], there is no need for a searchable encryption to explicitly provide an extra algorithm for decrypting the content of search results (e.g., data records) returned by the cloud server. It is because the data owner can always independently encrypt and decrypt each data record by leveraging a standard public-key (or symmetric-key) encryption.

3. SECURITY DEFINITION

Informally, the security objective of a searchable encryption is to reveal as less information as possible to the honest-but-curious cloud server while successfully searching the data records for a given query. As the descriptions of other searchable encryption schemes, before we introduce the security definition of our tree-based MRDSE, let us first define what kinds of information will be leaked to the cloud server.

Privacy Leakage. First, since the data owner stores all the data records in the cloud server and submits queries to the cloud server for search, the basic size information of data records and queries, such as the total number of data records (n), the number of dimensions (w), the size of each dimension ($|T_i|$), and the number of queries submitted (t), are inevitably leaked to the cloud server.

Definition 2. **(Size Pattern).** *Given* n *data records* $\mathbf{D} = \{D_1, ..., D_n\}$*, the size pattern of* $\mathbf{D}$ *induced by a number of* t *query* $\mathbf{Q} = \{Q_1, ..., Q_t\}$ *is a tuple* $\alpha(\mathbf{D}, \mathbf{Q}) = (n, w, |T_1|, ..., |T_w|, t)$*.*

In addition, as previous searchable encryption schemes [15, 19, 20], our tree-based MDRSE scheme will not be able

to preserve *access pattern* or *search pattern*. Specifically, revealing access pattern means the cloud server can learn the identifiers (e.g., memory locations in the server) of data records that satisfied for each query; leaking search pattern indicates the cloud server can distinguish whether a same data record is returned for two different queries. The definitions of access pattern and search pattern are presented as below by following [15, 19, 20].

Definition 3. **(Access Pattern).** *Given n data records $\mathbf{D} = \{D_1, ..., D_n\}$, the access pattern of $\mathbf{D}$ induced by a number of t query $\mathbf{Q} = \{Q_1, ..., Q_t\}$, is a tuple $\beta(\mathbf{D}, \mathbf{Q}) = (\mathbf{I}_{Q_1}, ..., \mathbf{I}_{Q_t})$, where $\mathbf{I}_{Q_i}$ is the set of identifiers of data records that returned by query Q_i, for $1 \leq i \leq t$.*

Definition 4. **(Search Pattern).** *Given n data records $\mathbf{D} = \{D_1, ..., D_n\}$, the search pattern of $\mathbf{D}$ induced by a number of t query $\mathbf{Q} = \{Q_1, ..., Q_t\}$, is an $n \times t$ binary matrix $\gamma(\mathbf{D}, \mathbf{Q})$ such that for $1 \leq i \leq n$, and $1 \leq j \leq t$, the element in the i-th row and j-th column is 1, if an identifier I_i is returned by a query Q_j.*

Theoretically speaking, access pattern and search pattern can be protected by using Oblivious RAM [13, 31], but the efficiency of Oblivious RAM is still a huge concern for real applications. How to protect access pattern and search pattern for multi-dimensional range queries is out of the scope of this paper.

Besides access pattern and search pattern mentioned in previous works, another privacy leakage we would like to particularly capture for a tree-based searchable encryption is *path pattern*. Specifically, revealing path pattern means the cloud server learns the paths from the root node to several leaf nodes in a tree structure for a given query (i.e., the identifiers of all the non-leaf nodes and all the leaf nodes that a query reaches while searching in a tree structure).

Definition 5. **(Path Pattern).** *Given n data records $\mathbf{D} = \{D_1, ..., D_n\}$ and the corresponding tree structure $\Gamma = \texttt{BuildTree}(\mathbf{D})$, the path pattern of $(\mathbf{D}, \Gamma)$ induced by a number of t query $\mathbf{Q} = \{Q_1, ..., Q_t\}$, is a tuple $\delta(\mathbf{D}, \Gamma, \mathbf{Q}) = (\mathbf{P}_{Q_1}, ..., \mathbf{P}_{Q_t})$, where $\mathbf{P}_{Q_i}$ is the set of identifiers of nodes in Γ that reached by query Q_i, for $1 \leq i \leq t$.*

Path pattern not only exists in the multi-dimensional tree structures we utilized in this paper, but also lives in general tree structures in the single dimension, such as binary trees. Essentially, we can also think path pattern as "access pattern" revealed in a tree structure for a given query.

One of the main motivations for us to capture this new pattern in a searchable encryption while the access pattern (defined in Def. 3) have already been included in the privacy leakage is that, given the same access pattern for two queries, the path pattern of these two queries in our later design may not be the same. A simple example of this case is illustrated in Fig. 4, where the search decision at each non-leaf node in tree Γ is made by testing whether a query intersects with the range represented that non-leaf node. Here tree Γ can be imaged as a simple case of an R-tree with only one dimension (further introduction of R-trees will be presented in Sec. 4).

To the best of our knowledge, our work is among the first to distinguish path pattern from the traditional access pattern and capture path pattern in the privacy leakage of a tree-based searchable encryption. Note that the recent solution [19] for keyword search based on the keyword red-black (KRB) tree does not need to particularly capture path pattern in its privacy leakage. It is because if the access pattern

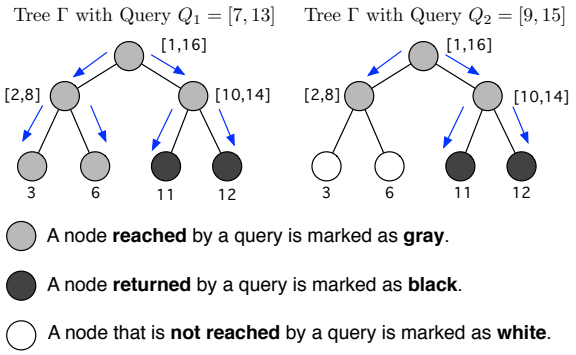

Figure 4: An example of query Q_1 and Q_2 on tree Γ, where Q_1 and Q_2 return the same node (the access pattern defined by Def. 3 are the same) but these two queries do not have the same search paths in the tree (the path pattern are different).

are the same in KRB-trees, the path pattern are also the same [19]. Further details of KRB-trees can found in [19].

Another leakage of our scheme is that, it will also reveal the values of each query to the cloud server (i.e., the cloud server learns which particular range the data owner is searching for a given query), which is referred to as *query privacy*. The query privacy induced by a number of t query $\mathbf{Q} = \{Q_1, ..., Q_t\}$ can be defined as a tuple $\eta(\mathbf{Q}) = (\mathbf{V}_{Q_1}, ..., \mathbf{V}_{Q_t})$, where $\mathbf{V}_{Q_i}$ is the set of values revealed by query Q_i. We will discuss the essential reason about why our scheme fails to protect query privacy, and how to *mitigate* the leakage of query privacy in practice in Sec. 5.

Now we define a *leakage function* [19, 20] to capture all the information leakage we mentioned above in a tree-based public-key MDRSE scheme.

Definition 6. **(Leakage Function).** *Given n data records $\mathbf{D} = \{D_1, ..., D_n\}$, the corresponding tree structure $\Gamma = \texttt{BuildTree}(\mathbf{D})$, the leakage function of $(\mathbf{D}, \Gamma)$ induced by a number of t query $\mathbf{Q} = \{Q_1, ..., Q_t\}$, is a sequence*

$$\mathcal{L}(\mathbf{D}, \Gamma, \mathbf{Q}) = \{\alpha(\mathbf{D}, \mathbf{Q}), \beta(\mathbf{D}, \mathbf{Q}), \gamma(\mathbf{D}, \mathbf{Q}), \delta(\mathbf{D}, \Gamma, \mathbf{Q}), \eta(\mathbf{Q})\},$$

comprised of size pattern, access pattern, search pattern, path pattern, and content of queries.

Selective Security. By understanding the preceding leakage function, we discuss the selective security of a tree-based public-key MDRSE scheme. Informally speaking, *selective security* means by submitting two sets $\mathbf{D}_0$, $\mathbf{D}_1$ of data records with the same length and isomorphic tree structure (i.e., $\Gamma_0 \simeq \Gamma_1$), a computationally bounded adversary, who has obtained search tokens for t queries $\mathbf{Q} = \{Q_1, ..., Q_t\}$ (*the selection of each query is restricted by $\mathbf{D}_0$, $\mathbf{D}_1$ and leakage function $\mathcal{L}$*), is not able to distinguish the two sets of data records.

Definition 7. **(Selective Security).** *Let $\Pi = (\texttt{GenKey}, \texttt{BuildTree}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ be a tree-based public-key MDRSE scheme over lattice $\mathbb{L}_\Delta$, $\lambda \in \mathbb{N}$ be the security parameter, the selective security game between an adversary and the challenger in Π is described as below*

- **Init:** *The adversary submits two sets $\mathbf{D}_0$ and $\mathbf{D}_1$ of data records 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 $1 \leq i \leq n$, $\Gamma_0 = \texttt{BuildTree}(\mathbf{D}_0)$ and $\Gamma_1 = \texttt{BuildTree}(\mathbf{D}_1)$.*

- **Setup:** *The challenger runs* $\texttt{GenKey}(1^\lambda, \Delta)$ *to generate a public key* PK *and a secret key* SK*. It gives* PK *to the adversary, and keeps* SK *private.*

- **Phase 1:** *The adversary adaptively requests search tokens for a number of* t' *range queries* $\mathbf{Q}' = \{Q_1, ..., Q_{t'}\}$ *with the following two restrictions:*

  1. *For* $1 \leq j \leq t'$, $\mathcal{L}(\mathbf{D}_0, \Gamma_0, Q_j) = \mathcal{L}(\mathbf{D}_1, \Gamma_1, Q_j)$;
  2. *For* $1 \leq j \leq t'$ *and for* $1 \leq i \leq n$, *either* $(D_{0,i} \in Q_j) \wedge (D_{1,i} \in Q_j)$, *or* $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$,

  The challenger responses each query Q_j, *for* $1 \leq j \leq t'$, *by running* $\texttt{GenToken}(\texttt{SK}, Q_j)$.

- **Challenge:** *The adversary submits two message sets* $\mathbf{M}_0 = \{M_{0,1}, ..., M_{0,n}\}$ *and* $\mathbf{M}_1 = \{M_{1,1}, ..., M_{1,n}\}$, *where for* $1 \leq i \leq n$, $M_{0,i}$ *and* $M_{1,i}$ *have the equal length and are subjected to the following restrictions:*

  1. *If there exist some* $j \in [1, t']$ *such that* $(D_{0,i} \in Q_j) \wedge (D_{1,i} \in Q_j)$, *then* $M_{0,i} = M_{1,i}$;
  2. *Otherwise, if for every* $j \in [1, t']$, $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$, *then* $M_{0,i} \neq M_{1,i}$.

  Then, the challenger flips a coin $b \in \{0, 1\}$, *and returns* $(\Gamma_b^*, \mathbf{C}_b) \leftarrow \texttt{Enc}(\texttt{PK}, \Gamma_b, \mathbf{M}_b)$ *to the adversary.*

- **Phase 2:** *The adversary continues to adaptively request search tokens for a number of* t *range queries* $\mathbf{Q}_t = (Q_{t'+1}, ..., Q_{t'+t})$, *which are still subjected to the same restrictions in Phase 1 and Challenge.*

- **Guess:** *The adversary takes a guess* b' *of* b.

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

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

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

For the ease of understanding, we can also say that, for $\mathbf{D}_0$ and $\mathbf{D}_1$, as long as the information leakage induced by $\mathbf{Q}$ are the same under leakage function $\mathcal{L}$, $\mathbf{D}_0$ and $\mathbf{D}_1$ are computationally indistinguishable to the adversary.

Note that the requirement of the *isomorphic* tree structure for the two sets of data records in the above security game is necessary, which is similar as the requirement of the *same length* for two messages in the security game of a traditional encryption (e.g., AES). Otherwise, two sets of data records can be easily distinguished via two different tree structures.

Single-Dimensional Privacy. As we mentioned in previous sections, protecting *single-dimensional privacy* is one of the major objectives in this paper. Informally, single-dimensional privacy indicates that, given a search token generated by a multi-dimensional query Q, the cloud server should not be able to obtain the exact search results for any single-dimensional query of Q. In the rest of this paper, we use Q^k to describe the single-dimensional query of Q in the k-th dimension. For instance, given a multi-dimensional query $Q = ([20, 30] \wedge [600, 800])$, the single-dimensional queries of Q in the first and second dimension are presented as $Q^1 = ([20, 30])$ and $Q^2 = ([600, 800])$, respectively. It is easy to see that, if $D \in Q$, then $D \in Q^k$ is true, for every $k \in [1, w]$. However, on the other hand, given $D \in Q^k$, for some $k \in [1, w]$, we cannot sure whether $D \in Q$ is true.

In order to rigorously capture single-dimensional privacy in a tree-based public-key MDRSE scheme, we would like to define a single-dimensional privacy game as follows:

Definition 8. **(Single-Dimensional Privacy).** *Let* $\Pi = (\texttt{GenKey}, \texttt{BuildTree}, \texttt{Enc}, \texttt{GenToken}, \texttt{Search})$ *be a tree-based public-key MDRSE scheme over lattice* $\mathbb{L}_\Delta$, $\lambda \in \mathbb{N}$ *be the security parameter, the single-dimensional privacy game between an adversary and the challenger in* Π *is described as below*

- **Init:** *is same as* **Init** *in* Def. 7.

- **Setup:** *is same as* **Setup** *in* Def. 7.

- **Phase 1:** *The adversary adaptively requests search tokens for a number of* t' *range queries* $\mathbf{Q}' = \{Q_1, ..., Q_{t'}\}$ *with the following three restrictions:*

  1. *For* $1 \leq j \leq t'$, $\mathcal{L}(\mathbf{D}_0, \Gamma_0, Q_j) = \mathcal{L}(\mathbf{D}_1, \Gamma_1, Q_j)$;
  2. *For* $1 \leq j \leq t'$ *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. *If* $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$, *for some* $i \in [1, n]$ *and some* $j \in [1, t']$, *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 challenger responses each query Q_j, *for* $1 \leq j \leq t'$, *by running* $\texttt{GenToken}(\texttt{SK}, Q_j)$.

- **Challenge:** *is same as* **Challenge** *in* Def. 7.

- **Phase 2:** *is same as* **Phase 2** *in* Def. 7.

- **Guess:** *The adversary takes a guess* b' *of* b.

The advantage of an adversary $\mathcal{A}$ *in the above single-dimensional privacy game is defined as* $\mathbf{Adv}^{SDP}_{\Pi,\mathcal{A}}(1^\lambda, \Delta)$. *We say that a tree-based public-key MDRSE scheme* Π *is single-dimensionally private, if for all polynomial time adversaries have at most negligible advantage in the above game:*

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

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

Note that if we compare the two games in Def. 7 and Def. 8, the main difference is that Def. 8 has an additional **third** restriction in Phase 1. It is easy to see that if a tree-based public-key MDRSE scheme is selectively secure defined in Def. 7, then it is also single-dimensionally private presented in Def. 8.

4. PRELIMINARIES

Hidden Vector Encryption. Hidden Vector Encryption (HVE) [9], which is a type of predicate encryption proposed by Boneh and Waters, can be utilized to search whether a value is inside a range (e.g., whether $x \in [a, b]$) without revealing this value. Specifically, the encryptor encrypts a pair (x, x), and then the predicate tests $(x \geq a) \wedge (x \leq b)$ over the ciphertext of x [9]. Besides, this advanced search is single-dimensionally private (i.e., the predicate will not reveal whether $x \geq a$ or $x \leq b$ independently). Details of security analyses and how to initialize this advanced search based on HVE can be found in [9].

According to [9, 28], the above method can be further extended to test whether a point is inside a hyper-rectangle

(PK, SK)←Setup (1^λ, Δ): Given a security parameter λ and $\Delta = (T_1, \cdots, T_w)$, output a public key PK and a secret key SK.

C ←Enc (PK, D, M): Given a public key PK, a point D, a message M, where $M \in \mathcal{M}$ and $\mathcal{M} = \{\texttt{Flag}\}$, output a ciphertext C.

$\texttt{TK}_Q$ ←GenToken (SK, Q): Given a secret key SK and a hyper-rectangle Q, output a token $\texttt{TK}_Q$.

$\{M, \perp\}$ ←Query ($\texttt{TK}_Q$, C): Given a token $\texttt{TK}_Q$ and a ciphertext C, output M, if $D \in Q$ (the decryption of C is $M = \texttt{Flag}$), and output $\perp$ otherwise.

Correctness: PPE is correct, if for all $D \in \mathbb{L}_\Delta$, all $Q \subseteq \mathbb{L}_\Delta$, all $M \in \mathcal{M}$, all (PK, SK)←Setup (1^λ, Δ), all C ←Enc (PK, D, M), all $\texttt{TK}_Q$ ←GenToken (SK, Q),

- $\texttt{Query}(\texttt{TK}_Q, C) = M$, if $D \in Q$;
- $\Pr[\texttt{Query}(\texttt{TK}_Q, C) = \perp] \geq 1 - \texttt{negl}(\lambda)$, if $D \notin Q$;

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

Figure 5: A Brief Description of PPE.

(e.g., $(x, y) \in [a, b] \wedge [c, d]$) without revealing the point. More concretely, the encryptor can encrypt a tuple (x, x, y, y), and then the predicate tests $(x \geq a) \wedge (x \leq b) \wedge (y \geq c) \wedge (y \leq d)$ over encrypted data. We simply denote this extended approach, which is essentially based on HVE as well, as Point Predicate Encryption (PPE) in this paper. A brief description of PPE is presented in Fig. 5. The security of PPE can be easily conducted based on the security of HVE [9, 28]. We will leverage PPE as a building block in our design to encrypt every node in a tree structure.

R-trees. The R-tree, proposed by Guttman [17], is a type of multi-dimensional data structure, which can be used in many applications, such as range search. The basic idea of an R-tree is to group nearby objects (e.g., points or hyper-rectangles) and represent them with a *bounding box* in the next higher level of the tree. Each leaf node in the tree represents a point (i.e., a data record), and each non-leaf node represents a bounding box (i.e., a hyper-rectangle). An example of an R-tree with four points in the two-dimension are described in Fig. 6.

Clearly, in an R-tree, if a range query does not intersect a bounding box, it also cannot intersect with any objects inside this box. Based on this principle, the search algorithm can efficiently filter most of the unmatched points out of the results while traversing an R-tree, which essentially improves the search efficiency of range queries. A detailed description of the search algorithm in an R-tree is illustrated in Algorithm 1. We can observe that **whether a point lies in a hyper-rectangle** (line 2) and **whether two hyper-rectangles intersect** (line 4) are the two basic factors to make decisions in the search algorithm of an R-tree.

Algorithm 1: SearchR-Tree(v,Q)

Input: The root (or a node) v of an R-tree, and a range query (i.e., a hyper-rectangle) Q.
Output: All points at leaves below v that lie in Q.

```
1 if v is a leaf then
2     Report the point stored at v if it lies in Q.
3 else if v is a non-leaf then
4     if The hyper-rectangle stored at v intersects Q then
5         SearchR-Tree(ChildNode(v), Q).
```

Note that there are other similar multi-dimensional tree structures, such as kd-trees [5], range trees [6] and etc., which can also provide a better search efficiency than linear search with regard to the number of data records. Besides the consideration of improving efficiency, another major reason for us to leverage R-trees is that the designs with R-trees have the potential to achieve single-dimensional privacy. While other multi-dimensional tree structures, like kd-trees or range trees, cannot satisfy single-dimensional privacy no matter what kind of encryption primitive is used on each node. Further discussions about this issue will be presented in the next section.

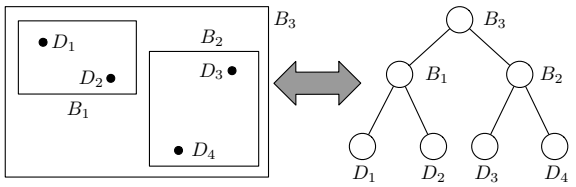

Figure 6: An example of an R-tree with four points.

5. TREE-BASED PUBLIC-KEY MDRSE

In this section, we will first show how to design a basic solution, which is based on PPE without any tree structure but only achieves linear search time regarding to the number of data records. Then, taking this basic solution as a stepping stone, we will describe how to build Maple, a tree-based public-key MDRSE scheme, which is able to not only improve the search efficiency (i.e., faster-than-linear complexity with regard to the number of data records) but also protect the privacy of data records.

Basic Solution. By directly using PPE without any tree structure, we can build a basic solution of public-key MDRSE. Specifically, the data owner can encrypt every data record with PPE, and outsource all the encrypted data to the cloud server. Then, the cloud server can perform search on all the encrypted data records **one by one** based on a token submitted by the data owner. Clearly, the search complexity of this basic solution is **linearly** increasing with the number of data records stored in the cloud server. Considering the databases outsourced to the cloud generally contain a huge number of records, this basic solution is clearly not practical. In fact, this basic solution is essentially the direct implementation of HVE [9] for multi-dimensional range queries. Although it is not efficient, this basic solution can still be included in the formal definition of a public-key MDRSE presented in Sec. 2 by setting tree structure Γ as null.

Our Design: Overview and Challenge. In order to improve the search efficiency, we would like to exploit multi-dimensional tree structures (specifically, R-trees) to index all the data records and handle range queries by exactly following the search algorithm of the tree structures. Obviously, it is easy to implement with traditional techniques if the data records are in plaintext. However, it is not simple to exactly follow the search algorithm of the R-tree while all the nodes in the tree are encrypted. More importantly, we need to protect single-dimensional privacy as well, which previous efficient solutions [25, 34] fail to achieve.

Specifically, we can still directly leverage PPE to encrypt every leaf node (i.e., a data record) in an R-tree in order to decide whether a point lies in a hyper-rectangle (line 2 in Algorithm 1) when a range query is submitted; besides, we also need to find a method to encrypt every non-leaf node, so that we can verify whether two hyper-rectangles intersect (line 4 in Algorithm 1) in the ciphertext domain given a range query. Interestingly, by continuing to use PPE in a novel way, we can find a solution to decide whether two hyper-rectangles intersect over encrypted data. More importantly, we can still protect single-dimensional privacy.

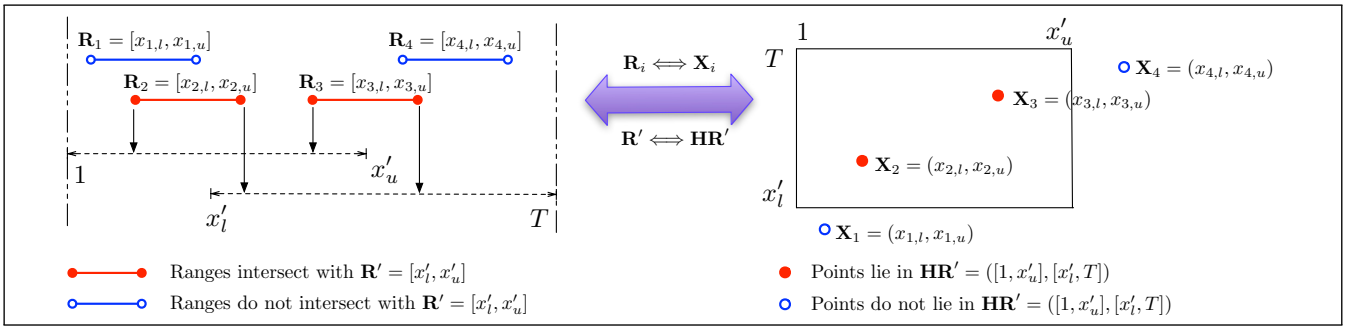

Figure 7: Examples of whether two ranges intersect, and the equivalent relation between a 2-dimensional point and a rectangle.

Using predicate encryption to test other types of geometric relations over encrypted data, such as whether a point is on a line, can be found in [36].

Hyper-Rectangle Intersection. We now explain how to use PPE to verify whether two hyper-rectangles intersect in the ciphertext domain without revealing single-dimensional privacy. For the ease of description, we first start with testing whether two ranges intersect in a single dimension, and then we extend this method into the multi-dimension for verifying the intersection of two hyper-rectangles. The essential idea of our method here is to first transform the geometric relation of two ranges into an **equivalent** geometric relation between a 2-dimensional point and a rectangle, and then encrypt this 2-dimensional point with PPE.

Specifically, given two ranges $\mathbf{R}$ and $\mathbf{R}'$ in a single dimension, we treat the first range $\mathbf{R} = [x_l, x_u]$ as a 2-dimensional point as $\mathbf{X} = (x_l, x_u)$, and transform the second range $\mathbf{R}' = [x'_l, x'_u]$ into a rectangle as $\mathbf{HR}' = ([1, x'_u], [x'_l, T])$. Then, we can decide whether the two original ranges $\mathbf{R}$, $\mathbf{R}'$ intersect by checking whether the point $\mathbf{X}$ lies in the rectangle $\mathbf{HR}'$. If the point is indeed in the rectangle, then the two ranges intersect with each other. Otherwise, they do not intersect. The reason is that these two geometric relations are equivalent, which is proved in the following lemma. Some examples of the intersections of two ranges and the equivalent geometric relation between a point and a rectangle are described in Fig. 7.

Lemma 1. *For two ranges* $\mathbf{R} = [x_l, x_u]$ *and* $\mathbf{R}' = [x'_l, x'_u]$, *we have a point* $\mathbf{X} = (x_1, x_2) = (x_l, x_u)$ *based on range* $\mathbf{R}$ *and a rectangle* $\mathbf{HR}' = (\mathbf{R}'_1, \mathbf{R}'_2)$ *based on range* $\mathbf{R}'$, *where* $\mathbf{R}'_1 = [1, x'_u]$ *and* $\mathbf{R}'_2 = [x'_l, T]$. *If* $\mathbf{X} \in \mathbf{HR}'$, *then* $\mathbf{R} \cap \mathbf{R}' \neq \varnothing$; *otherwise,* $\mathbf{X} \notin \mathbf{HR}'$, *then* $\mathbf{R} \cap \mathbf{R}' = \varnothing$.

The correctness of this lemma can be explained as below

$$\mathbf{R} \cap \mathbf{R}' \neq \varnothing \Leftrightarrow \begin{cases} x_l \leq x'_u \\ x_u \geq x'_l \end{cases} \Leftrightarrow \begin{cases} x_l \in [1, x'_u] \\ x_u \in [x'_l, T] \end{cases} \Leftrightarrow \mathbf{X} \in \mathbf{HR}'.$$

Based on this equivalent geometric relation, we can actually encrypt a special message (e.g., `Flag`) with this 2-dimensional point (transformed from $\mathbf{R}$) using PPE, then the decryption result of the ciphertext with a token generated by a rectangle (transformed from $\mathbf{R}'$) will be the special message `Flag`, if the two ranges intersect (i.e., $\mathbf{R} \cap \mathbf{R}' \neq \varnothing$). In this way, we can decide the intersection of two ranges over ciphertexts.

Why Do We Transfer It to An Equivalent Geometric Relation? Readers may wonder why do we have to transfer the geometric relation of two ranges to an equivalent relation between a 2-dimensional point and a rectangle before encryption. How about we directly verify whether the two ranges (i.e., $\mathbf{R} = [x_l, x_u]$ and $\mathbf{R}' = [x'_l, x'_u]$) intersect by checking whether $x_l \in [1, x'_u]$ and $x_u \in [x'_l, T]$ respectively? Is this simpler?

In fact, we have no doubt that the method mentioned in the previous paragraph is simpler than ours, and it is indeed able to correctly verify whether the two ranges intersect. However, the problem about this method is that if we verify the two values (i.e., x_l and x_u) separately, it will inevitably compromise single-dimensional privacy as the same as the limitation introduced in LSED$^+$ [25] and the example we explained in Fig. 1.

Now, by extending our method in Lemma 1, we can decide whether the intersection of two hyper-rectangles is null in the multi-dimension.

Lemma 2. *For two hyper-rectangles* $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$ *and* $\mathbf{HR}' = (\mathbf{R}'_1, ..., \mathbf{R}'_w)$, *where* $\mathbf{R}_i = [x_{i,l}, x_{i,u}]$ *and* $\mathbf{R}'_i = [x'_{i,l}, x'_{i,u}]$, *for* $i \in [1, w]$, *we have a point* $\mathbf{X} = (x_1, ..., x_{2w}) = (x_{1,l}, x_{1,u}, ..., x_{w,l}, x_{w,u})$ *and a hyper-rectangle* $\mathbf{HR}'' = (\mathbf{R}''_1, ..., \mathbf{R}''_{2w})$, *where* $\mathbf{R}''_{2i-1} = [1, x'_{i,u}]$ *and* $\mathbf{R}''_{2i} = [x'_{i,l}, T]$, *for* $1 \leq i \leq w$. *If* $\mathbf{X} \in \mathbf{HR}''$, *then* $\mathbf{HR} \cap \mathbf{HR}' \neq \varnothing$; *otherwise,* $\mathbf{X} \notin \mathbf{HR}''$, *then* $\mathbf{HR} \cap \mathbf{HR}' = \varnothing$.

The correctness of this lemma can be proved using the similar approach in Lemma 1. Similarly, we can encrypt a special message `Flag` with this $2w$-dimensional point (transformed from $\mathbf{HR}$) using PPE, and the result of the decryption of the ciphertext with a token generated by a $2w$-dimensional hyper-rectangle (transformed from $\mathbf{HR}'$) is `Flag`, if $\mathbf{HR} \cap \mathbf{HR}' \neq \varnothing$.

Maple: Scheme Details. With the approach we explained above, we now introduce Maple, a tree-based public-key MDRSE scheme by following the definition presented in Sec. 2. Details of Maple are illustrated in Fig. 8. Specifically, the data owner first generates a public key and a secret key in `GenKey`, and builds an R-tree Γ based on a set $\mathbf{D}$ of data records in `BuildTree`. In `Enc`, the data owner encrypts every node in Γ, and outputs an encrypted R-tree Γ^* and a set of ciphertexts, where Γ^* has the isomorphic tree structure as Γ. Each leaf node (i.e., a point) is directly encrypted with PPE; while each non-leaf node (i.e., a hyper-rectangle) is first transformed into a $2w$-dimensional point, and then encrypted with PPE.

The data owner is able to generate a search token based on a range query in `GenToken`, and submit it to the cloud server to operate `Search`, which starts from the root node of the encrypted R-tree Γ^*. As presented in Fig. 8, at a non-leaf node, if the decryption of a ciphertext is `Flag` (means the hyper-rectangle represented this non-leaf node intersects with the hyper-rectangle represented the range query), then the cloud server continues to search all the child nodes of this non-leaf node; otherwise, the cloud server stops searching its

$\texttt{GenKey}(1^\lambda, \Delta)$: Given a security parameter λ and $\Delta = (T_1, ..., T_w)$, the data owner defines 2Δ as $2\Delta = (T_1, T_1, ..., T_w, T_w)$, and computes

$$(\texttt{PK}_1, \texttt{SK}_1) \leftarrow \texttt{PPE.Setup}(1^\lambda, \Delta),$$
$$(\texttt{PK}_2, \texttt{SK}_2) \leftarrow \texttt{PPE.Setup}(1^\lambda, 2\Delta);$$

and outputs a public key $\mathbf{PK} = \{\texttt{PK}_1, \texttt{PK}_2\}$ and a secret key $\mathbf{SK} = \{\texttt{SK}_1, \texttt{SK}_2\}$.

$\texttt{BuildTree}(\mathbf{D})$: Given a set of data records $\mathbf{D} = (D_1, ..., D_w)$, where $D_i = (d_{i,1}, ..., d_{i,w})$, for $1 \le i \le w$, the data owner builds an R-tree Γ.

$\texttt{Enc}(\mathbf{PK}, \Gamma, \mathbf{M})$: Given a public key $\mathbf{PK}$, an R-tree Γ and a set of messages $\mathbf{M} = (M_1, ..., M_n)$, the data owner encrypts each node in R-tree Γ as follows:

- If it is a non-leaf node, let $\mathbf{HR}$ be the hyper-rectangle that represents this node, transform $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$, where $\mathbf{R}_i = [x_{i,l}, x_{i,u}]$, into a point $\mathbf{X} = (x_1, ..., x_{2w}) = (x_{1,l}, x_{1,u}, ..., x_{w,l}, x_{w,u})$, and compute

$$C \leftarrow \texttt{PPE.Enc}(\texttt{PK}_2, \mathbf{X}, \texttt{Flag}),$$

- If it is a leaf node, it can be represented as a data record $D_i = (d_{i,1}, ..., d_{i,w})$. Let $\mathbf{X}$ be the point that represents this node, set $\mathbf{X} = (x_1, ..., x_w) = (d_{i,1}, ..., d_{i,w})$, and compute

$$C_i \leftarrow \texttt{PPE.Enc}(\texttt{PK}_1, \mathbf{X}, M_i),$$

where $M_i \in \mathcal{M}$ and $\mathcal{M} = \{\texttt{Flag}\}$.

Eventually, the data owner outputs an encrypted R-tree Γ^* and a set of ciphertext $\mathbf{C} = (C_1, ..., C_n)$.

$\texttt{GenToken}(\mathbf{SK}, Q)$: Given a secret key $\mathbf{SK}$ and a range query $Q = [x_{1,l}, x_{1,u}] \wedge \cdots \wedge [x_{w,l}, x_{w,u}]$, the data owner represents Q as a hyper-rectangle $\mathbf{HR} = (\mathbf{R}_1, ..., \mathbf{R}_w)$, where $\mathbf{R}_i = [x_{i,l}, x_{i,u}]$ and $1 \le i \le w$; generates another hyper-rectangle $\mathbf{HR}' = (\mathbf{R}'_1, ..., \mathbf{R}'_{2w})$, where $\mathbf{R}'_{2i-1} = [1, x_{i,u}]$ and $\mathbf{R}'_{2i} = [x_{i,l}, T_i]$, for $1 \le i \le w$; computes

$$\texttt{TK}_1 \leftarrow \texttt{PPE.GenToken}(\texttt{SK}_1, \mathbf{HR}),$$
$$\texttt{TK}_2 \leftarrow \texttt{PPE.GenToken}(\texttt{SK}_2, \mathbf{HR}');$$

and outputs $\mathbf{TK}_Q = \{\texttt{TK}_1, \texttt{TK}_2\}$.

$\texttt{Search}(\Gamma^*, \mathbf{C}, \mathbf{TK}_Q)$: Given an encrypted R-tree Γ^*, a set of ciphertexts $\mathbf{C} = \{C_1, ..., C_n\}$, a search token $\mathbf{TK}_Q$, the cloud server searches from the root node of Γ^* and performs as follows:

- If it is a non-leaf node represented with a ciphertext C, then if

$$\texttt{Flag} = \texttt{PPE.Query}(\texttt{TK}_2, C);$$

continue to search its child nodes; otherwise stop searching its child nodes.

- If it is a leaf node represented with a ciphertext C_i, then if

$$M_i = \texttt{PPE.Query}(\texttt{TK}_1, C_i);$$

where $M_i \in \mathcal{M}$ and $\mathcal{M} = \{\texttt{Flag}\}$, indicate $D_i \in Q$ and return I_i; otherwise, do not return I_i.

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

Figure 8: Details of Maple.

child nodes. At a leaf node, if the decryption of a ciphertext is $M_i \in \mathcal{M}$ (indicates the point represented this leaf node lies in the hyper-rectangle represented the range query), the cloud server returns the identifier (e.g., memory location) of the corresponding data record stored on this leaf node. Eventually, the cloud server returns all the identifiers of the data records that satisfy the submitted range query.

Correctness. Since every node in an encrypted R-tree Γ^* is essentially encrypted with an instance of PPE, the correctness of Maple can be easily explained based on the correctness of PPE. Details are presented in Appendix.

Efficiency. Compared to the search algorithm (described in Sec. 4) of an R-tree in the plaintext domain, the search algorithm in Maple **exactly** follows the same search rules in an R-tree not only at every non-leaf node but also at every leaf node, which enables Maple to achieve the same search complexity as an R-tree with regard to the number of data records. Therefore, our scheme is able to achieve a better efficiency (faster-than-linear complexity regarding to the number of data records) compared to the basic solution.

According to the analyses in [9, 28], an instance of PPE on each node in the tree introduces $O(wT)$ encryption time, ciphertext size and public key size, and $O(w)$ token size and search time, where w denotes the number of dimensions and T denotes the size of each dimension.

Security Analysis. We now discuss the security of Maple.

Theorem 1. *(Selective Security). Maple is selectively secure, if HVE is selectively secure.*

PROOF. In our scheme, the encryption of an R-tree Γ is essentially to encrypt all the nodes in Γ one by one. Every node can be *directly or indirectly* encrypted with an instance of PPE, which is essentially an instance of HVE. Briefly speaking, since HVE is selectively secure, Maple – the multiple instances of HVE – is also selectively secure, which is based on the claims about the security of multiple encryption from the textbook [21]. Detailed analyses of Maple by following the selective security game defined in Sec. 2 are presented in Appendix. □

Theorem 2. *(Single-Dimensional Privacy). Maple is single-dimensionally private, if it is selectively secure.*

PROOF. According to Def. 7 and Def. 8, if Maple is selectively secure under Def. 7, then it is single-dimensionally private under Def 8. See details in Appendix. □

Note that in Maple, the search token $\mathbf{TK}_Q$ contains two sub-tokens $\{\texttt{TK}_1, \texttt{TK}_2\}$, which can be separately used to test different geometric relations during range search in an R-tree (described in Fig. 8). Since each sub-token is generated based on all the w dimensions using PPE, the separate use of these sub-tokens does not compromise single-dimensional privacy of Maple.

About Query Privacy. As we defined in the leakage function $\mathcal{L}$, our scheme does not protect query privacy. It is because the HVE [9] we essentially leveraged in Maple is a public-key approach. While public-key approaches inherently fail to protect query privacy as pointed out by [27].

Based on the current design, one practical approach for us to *mitigate* query privacy leakage in real applications is to add redundancy for each range query. Specifically, for each range query, the data owner can slightly and randomly enlarge the range query to a redundant one. Then, it will first retrieve redundant encrypted results from the cloud server, decrypt them, and calculate the exact results on local devices. In this way, the cloud server will not directly learn the actual range query that the data owner is searching for. As a necessary trade-off, the data owner needs to spend additional overheads to retrieve the redundant encrypted results from the cloud server, and consume extra overheads to calculate the exact results on its local devices as well.

Design with Other Trees. With the same designing methodology (i.e., leveraging HVE to verify geometric relations), we can also build tree-based MDRSE schemes with

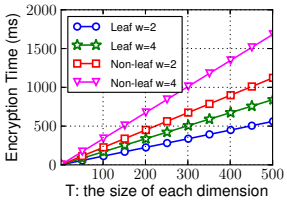

Figure 9: Impact of T on encryption time (millisecond) per node.

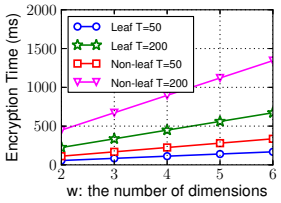

Figure 10: Impact of w on encryption time (millisecond) per node.

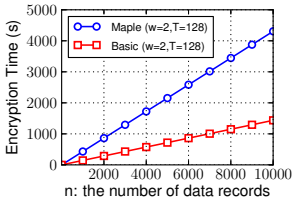

Figure 11: Impact of n on the total encryption time (second).

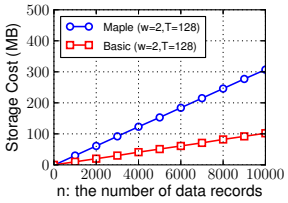

Figure 12: Impact of n on the total storage cost (MB).

different trees, such as kd-trees [5] and range trees [6]. Because the search algorithms of these trees also depend on the geometric relations of hyper-rectangles. However, due to the different nature of tree structures, the design with these two tree structures cannot achieve single-dimensional privacy, even through the use of HVE itself can protect the privacy in single dimensions.

The reason is that, in these trees, the search decision (i.e., the test of the geometric relation) from one node to its child nodes only depends on a single dimension. For instance, the search algorithm of a range tree is to first search all the results in X-dimension, and then based on these results, it will continue to search in Y-dimension. As a result, while searching from one node to its child nodes in X-dimension, the search algorithm does not need to consider the predicate of a multi-dimensional range query in other dimensions. In this case, the cloud server can stop the algorithm right after the search in X-dimension, which easily compromise the single-dimensional privacy in X-dimension. The privacy of Y-dimension can be compromised as well by searching all the data records only in Y-dimension (by skipping X-dimension). The privacy of each particular single dimension in kd-trees can be compromised in a similar way by skipping irrelevant dimensions. Due to the space limitation, we omit the further details in this paper.

Informally, with these trees and HVE, we could still achieve selective security, but only with *tighter* restrictions:

- *For* $1 \leq j \leq t'$, $\mathcal{L}(\mathbf{D}_0, \Gamma_0, Q_j) = \mathcal{L}(\mathbf{D}_1, \Gamma_1, Q_j)$;
- *For* $1 \leq j \leq t'$, *for* $1 \leq i \leq n$ *and for* $1 \leq k \leq w$, *either* $(D_{0,i} \in Q_j^k) \wedge (D_{1,i} \in Q_j^k)$, *or* $(D_{0,i} \notin Q_j^k) \wedge (D_{1,i} \notin Q_j^k)$.

If we compare the restrictions of the single-dimensional privacy game presented in Def. 8, it is easy to see that the above restrictions fail to protect single-dimensional privacy.

R-trees do not have such type of limitations, because the search decision of R-trees from one node to its child nodes depends on all the dimensions. It is worth to notice that, the recent work in [34] is also based on R-trees, however, it is not able to protect single-dimensional privacy. The reason is that the encryption primitive they used at each node requires to divide each search token of a multi-dimensional query into the independent ones in single dimensions just like the limitation introduced in [25]. Therefore, according to our observation, in order to design a faster-than-linear and single-dimensionally private tree-based MDRSE scheme, we need to be careful about not only what type of tree structures will be leveraged, but also what kind of encryption primitive at each node will be used.

6. PERFORMANCE

In this section, we evaluate the performance of our scheme in experiments based on a large-scale real-world dataset. All the experiments are tested in Ubuntu with Intel Core i5 3.30 GHz Processor and 4 GB Memory. We leverage Pairing-Based Library (PBC[2]) to simulate the cost of cryptographic operations. More specifically, the cost of a pairing operation, which is the dominating operation in the search of Maple, is around 6.33 milliseconds (tested on super-singular curve $y^2 = x^3 + x$). We use a part of a large-scale real world data set with millions of records (U.S. census data 1990[3]) to evaluate the search performance of our scheme and the basic solution. In the following experiments, the number of dimensions is denoted as w; the number of data records is denoted as n; and we assume the size of each dimension T.

Encryption Overhead. As shown in Fig. 9 and 10, the encryption time of a non-leaf node and a leaf node are linearly increasing with the number of dimensions and the size of each dimension respectively. For Maple, the total encryption time for all the data records is less efficient than the one in the basic solution as illustrated in Fig. 11. It is because Maple needs to spend additional time to encrypt all the non-leaf nodes in the tree compared to the basic solution. In addition, Maple also requires additional overhead in terms of the storage cost compared to the basic solution. These extra overheads in encryption time and storage cost are necessary trade-offs for improving search efficiency. It is because the essential idea of using tree structures (even in the plaintext domain) is to reduce search time by spending more cost in terms of indexing and storage.

Token Generation Time. In Fig. 13, we describe and compare the generation time of a search token in Maple and the basic solution. Maple requires more time in this process because Maple needs to essentially generate two sub-tokens, one for verifying whether one point lies in a hyper-rectangle and one for deciding whether two hyper-rectangles intersect. While in the basic solution, it only needs to compute one token to test whether a point lies in a hyper-rectangle.

Search Efficiency. We can see from Fig. 14 that the search time per node in Maple is linearly increasing with the number of dimensions. Meanwhile, we can also observe that the search time for each leaf node and the one for each non-leaf node in our scheme are different. The reason is that for each non-leaf node, Maple verifies the geometric relation (whether two hyper-rectangle intersect) with a PPE instance of $2w$ dimensions; while for each leaf node, deciding the geometric relation (whether a point lies in a hyper-rectangle) is performed with a PPE instance of w dimensions.

Next, we compare the efficiency of Maple and the basic solution with 20 random range queries for each case. In order to simulate the search efficiency of Maple, we run the search code of R-trees in the plaintext domain, but add additional time of computing cryptographic operations for testing the geometric relation at each node during the search process.

[2] crypto.stanford.edu/pbc/

[3] http://archive.ics.uci.edu/ml/datasets.html

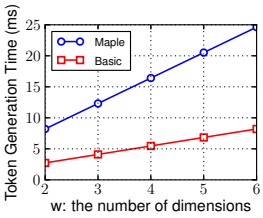

Figure 13: Impact of w on generation time (millisecond) per token.

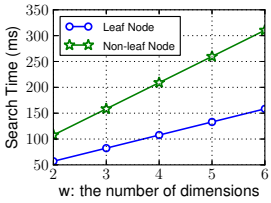

Figure 14: Impact of w on search time (millisecond) per node.

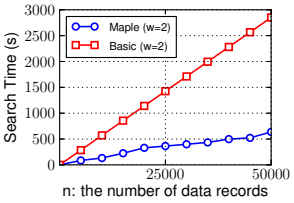

Figure 15: Impact of n on search time (second) with $w = 2$.

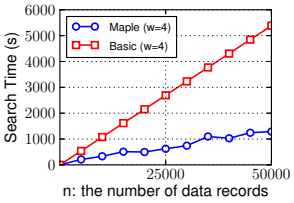

Figure 16: Impact of n on search time (second) with $w = 4$.

As presented in Fig. 15 and 16, the total search time of the basic solution is linearly increasing with the number of data records while the search time can be significantly reduced with the use of Maple.

In Table 2, we compare the search performance of Maple and the basic solution under different scales of data records. We can observe that Maple has a better scalability for large datasets. Specifically, when $n = 100,000$ and $w = 2$, the basic solution requires an average of $5,700$ seconds on each range query while Maple only costs about 928 seconds for operating each range query. Since the dominating cryptographic operations during search is pairing operations [9], we can reduce the computation time for each pairing operation by taking the advantage of special hardware (such as Elliptic Semiconductor CLP-17[4]), which can essentially further improve the search efficiency of Maple in practice.

Table 2: Average Search Time (second) when $w = 2$.

The number of data records	Basic	Maple
1,000	57.02	18.16
10,000	570.18	133.33
100,000	5,700.80	928.42

7. RELATED WORK

Searchable Encryption. SE schemes can be divided into two categories, symmetric-key-based and public-key-based. The first practical symmetric-key keyword search for encrypted text documents was proposed by Song et al. [29], which has *linear* complexity with regard to the document length. Since then, the quest for higher efficiency and better functionality has never stopped. Various schemes have been proposed to improve the efficiency of keyword search by creating an encrypted searchable index such like an inverted index [14,15,33,35]. However, the above works can only support single keyword search functionalities. Recently Cao et al. [11] proposed a multi-keyword ranked search scheme over encrypted cloud data. Kamara et al. focused on enabling privacy-preserving dynamic updates for SSE [20]. However, the search complexity is still *linear* with the number of documents. Moreover, the above works mostly focus on document search and do not apply very well to multi-dimensional range search considered in this paper.

On the other hand, the first public-key keyword searchable encryption was proposed by Boneh et al. [7]. To support more complex queries, conjunctive keyword search solutions over encrypted data have been proposed [16], where the search needs to touch every data record. Bellare et al. [4] proposed an efficient deterministic searchable public-key encryption scheme. Logarithmic search complexity can be achieved with this scheme; however, it only allows equality test. Li et al. [24] proposed a public-key scheme by introducing an additional trusted proxy that keeps a secret proxy key. Unfortunately, it lacks support for efficient arbitrary range queries. Recently, Lai et al. proposed a public-key scheme to support expressive keyword search [23].

Predicate Encryption. Predicate Encryption (or Functional Encryption) is a promising approach to achieve more flexible search functionalities [8, 26]. In general, given a ciphertext encrypted under an attribute A and a predicate function $f()$, the encryption enables a user who holds the corresponding token to $f()$ to test whether $f(A) = 1$ without decrypting ciphertexts.

The primitives designed in [9, 28] are both predicate encryption schemes. According to our discussion in Sec. 1, the straightforward applications of these two schemes can only achieve linear complexity. Katz et al. [22] proposed another public-key predicate encryption scheme that supports inner products. Theoretically, equality, subset, conjunctive, disjunctive predicates can all be expressed in the form of inner-products. However, this scheme incurs an exponential blowup for handling range queries. A good summary about the security definitions of Functional Encryption (FE) and some recent positive results about FE can be found in [1].

8. CONCLUSION

In this paper, we design a tree-based public-key MDRSE for supporting multi-dimensional range queries on encrypted cloud data. We formally define the leakage function and security game of a tree-based public-key MDRSE. By using non-trivial combination of R-trees and HVE, our proposed scheme is able to improve search efficiency and protect single-dimensional privacy simultaneously, while previous solutions fail to achieve. Through rigorous analyses and experiments with real-world datasets, we demonstrate the security and efficiency of our scheme. Our future works include 1) enhancing query privacy while keeping all the good properties in the current design; 2) studying other types of tree structures to improve efficiency and privacy as well.

9. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for providing useful suggestions and pointing out the insecurity of our preliminary version based on kd-trees. 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.

10. REFERENCES

[1] S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai. Function Private Functional Encryption and Property Preserving Encryption: New Definitions and Positive Resuslts. https://eprint.iacr.org/2013/744.

[4] http://www.elliptictech.com

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, Apirl 2010.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable Data Possession at Untrusted Stores. In *Proc. of ACM CCS'07*, pages 598–610, 2007.

[4] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and Efficiently Searchable Encryption. In *Proc. of CRYPTO'07*, pages 535–552, 2007.

[5] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] J. L. Bentley. Decomposable Searching Problems. *Information Processing Letters*, 8(5):201–244, 1979.

[7] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In *Proc. of EUROCRYP'04*, pages 506–522, 2004.

[8] D. Boneh, A. Sahai, and B. Waters. Functional Encryption: A New Vision for Public Key Cryptography. *Communications of the ACM*, 55(11):56–64, 2012.

[9] D. Boneh and B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Proc. of TCC'07*, pages 535–554, 2007.

[10] X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Proc. of CRYPTO'06*, pages 290–307, 2006.

[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data. In *Proc. of IEEE INFOCOM'11*, pages 829–837, 2011.

[12] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. In *Proc. of NDSS'14*, 2014.

[13] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries . In *Proc. of CRYPTO'13*, pages 353–373, 2013.

[14] Y.-C. Chang and M. Mitzenmacher. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *Proc. of ACNS'05*, pages 442–455, 2005.

[15] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proc. of ACM CCS'06*, pages 79–88, 2006.

[16] P. Golle, J. Staddon, and B. Waters. Secure Conjunctive Keyword Search over Encrypted Data. In *Proc. of ACNS'04*, pages 31–45, 2004.

[17] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proc. of ACM SIGMOD'84*, pages 47–57, 1984.

[18] S. Hwang, K. Kwon, S. Cha, and B. Lee. Performance Evaluation of Main-Memory R-tree Variants. In *Advances in Spatial and Temporal Databases*, 2003.

[19] S. Kamara and C. Papamanthou. Parallel and Dynamic Searchable Symmetric Encryption. In *Proc. of FC*, pages 258–274, 2013.

[20] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic Searchable Symmetric Encryption. In *Proc. of ACM CCS'12*, pages 965–976, 2012.

[21] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC Press, 2007.

[22] J. Katz, A. Sahai, and B. Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Proc. of EUROCRYPT'08*, pages 146–162, 2008.

[23] J. Lai, X. Zhou, R. H. Deng, Y. Li, and K. Chen. Expressive Search on Encrypted Data. In *Proc. of ACM ASIACCS'13*, pages 243–251, 2013.

[24] M. Li, S. Yu, N. Cao, and W. Lou. Authorized Private Keyword Search over Encrypted Data in Cloud Computing. In *Proc. of IEEE ICDCS'11*, pages 383–392, 2011.

[25] Y. Lu. Privacy-Preserving Logarithmic-time Search on Encrypted Data in Cloud. In *Proc. of NDSS'12*, 2012.

[26] T. Okamoto and K. Takashima. Hierarchical Predicate Encryption for Inner-Products. In *Proc. of ASIACRYPT'09*, pages 214–231, 2009.

[27] E. Shen, E. Shi, and B. Waters. Predicate Privacy in Encryption Systems. In *Proc. of TCC'09*, pages 457–473, 2009.

[28] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig. Multi-Dimensional Range Query over Encrypted Data. In *Proc. of IEEE S&P'07*, pages 350–364, 2007.

[29] D. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *Proc. of IEEE S&P'00*, pages 44–55, 2000.

[30] E. Stefanov, C. Papamanthou, and E. Shi. Practical Dynamic Searchable Encryption with Small Leakage. In *Proc. of NDSS'14*, 2014.

[31] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In *Proc. of ACM CCS'13*, pages 299–310, 2013.

[32] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li. Privacy-Preserving Multi-keyword Text Search in the Cloud Supporting Similarity-based Ranking. In *Proc. of ACM AISACCS'13*, pages 71–82, 2013.

[33] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure Ranked Keyword Search over Encrypted Cloud Data. In *Proc. of ICDCS'10*, pages 253–262, 2010.

[34] P. Wang and C. V. Ravishankar. Secure and Efficient Range Queries on Outsourced Databases Using R-trees. In *Proc. of IEEE ICDE'13*, pages 314–325, 2013.

[35] Z. Yang, S. Zhong, and R. N. Wright. Privacy-Preserving Queries on Encrypted Data. In *Proc. of ESORICS'06*, pages 479–495, 2006.

[36] M. Zhang and T. Takagi. GeoEnc: geometric are based keys and policies in functional encrytion systems. In *Proc. of ACISP'11*, pages 241–258, 2011.

APPENDIX

Proof of Correctness. *We say Maple is correct if for all* $\lambda \in \mathbb{N}$*, all* $(\mathbf{PK}, \mathbf{SK})$ *output by* $\mathtt{GenKey}(1^\lambda, \Delta)$*, all* $D_i \in \mathbb{L}_\Delta$*, all* Γ *output by* $\mathtt{BuildTree}(\mathbf{D})$*, for all* $M_i \in \mathcal{M}$*, all* $(\Gamma^*, \mathbf{C})$ *output by* $\mathtt{Enc}(\mathbf{PK}, \Gamma, \mathbf{M})$*, all* $Q \subseteq \mathbb{L}_\Delta$*, all* $\mathbf{TK}_Q$ *output by* $\mathtt{GenToken}(\mathbf{SK}, Q)$*, for any* $i \in [1, n]$*:*

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

where $\texttt{negl}(\lambda)$ *is a negligible function in* λ. It is because if $D_i \in Q$,

$$\begin{aligned} & \Pr[\texttt{Search}(\Gamma^*, \mathbf{C}, \mathbf{TK}_Q) = \mathbf{I}_Q, \text{ where } I_i \in \mathbf{I}_Q] \\ = \ & \Pr[\texttt{PPE.Query}(\texttt{TK}_1, C_i) = M_i, \text{ where } M_i \in \mathcal{M}] = 1. \end{aligned}$$

If $D_i \notin Q$,

$$\begin{aligned} & \Pr[\texttt{Search}(\Gamma^*, \mathbf{C}, \mathbf{TK}_Q) = \mathbf{I}_Q, \text{ where } I_i \notin \mathbf{I}_Q] \\ = \ & \Pr[\texttt{PPE.Query}(\texttt{TK}_1, C_i) = \perp] \geq 1 - \texttt{negl}(\lambda). \end{aligned}$$

Proof of Theorem 1. In **Init** of the selective security game of Maple, it is equivalent for the adversary to submit two sets of points $\{\mathbf{D}_0, \mathbf{B}_0\}$ and $\{\mathbf{D}_1, \mathbf{B}_1\}$ with $\Gamma_0 \simeq \Gamma_1$, $\Gamma_0 = \texttt{BuildTree}(\mathbf{D}_0)$ and $\Gamma_1 = \texttt{BuildTree}(\mathbf{D}_1)$, where

$$\begin{aligned} \{\mathbf{D}_0, \mathbf{B}_0\} &= \{D_{0,1}, ..., D_{0,n}, B_{0,1}, ..., B_{0,m}\}, \\ \{\mathbf{D}_1, \mathbf{B}_1\} &= \{D_{1,1}, ..., D_{1,n}, B_{1,1}, ..., B_{1,m}\}. \end{aligned}$$

Here, each D denotes a leaf node, each B denotes a non-leaf node, n denotes the number of leaf nodes and m represents the number of non-leaf nodes. Since the two tree structures are isomorphic, without loss of generality, there should exist an isomorphic function f between these two sets, such that for $1 \leq i \leq n$ and $1 \leq k \leq m$, $f(D_{0,i}) = D_{1,i}$ and $f(B_{0,j}) = B_{1,j}$.

In the **Setup**, the challenger runs $\texttt{GenKey}(1^\lambda, \Delta)$ as

$$\begin{aligned} (\texttt{PK}_1, \texttt{SK}_1) &\leftarrow \texttt{PPE.Setup}(1^\lambda, \Delta), \\ (\texttt{PK}_2, \texttt{SK}_2) &\leftarrow \texttt{PPE.Setup}(1^\lambda, 2\Delta); \end{aligned}$$

to generate a public key $\mathbf{PK} = \{\texttt{PK}_1, \texttt{PK}_2\}$ and secret key $\mathbf{SK} = \{\texttt{SK}_1, \texttt{SK}_2\}$. It gives $\mathbf{PK}$ to the adversary, and keeps $\mathbf{SK}$ private.

In the **Phase 1**, The adversary adaptively issues search tokens for a number of t' range queries $\mathbf{Q}' = \{Q_1, ..., Q_{t'}\}$ with the following two restrictions:

1. For $1 \leq j \leq t'$, $\mathcal{L}(\mathbf{D}_0, \Gamma_0, Q_j) = \mathcal{L}(\mathbf{D}_1, \Gamma_1, Q_j)$;
2. For $1 \leq j \leq t'$ and for $1 \leq i \leq n$, either $(D_{0,i} \in Q_j) \wedge (D_{1,i} \in Q_j)$, or $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$.

According to the definition of our leakage function, the first restriction indicates that each query Q_j has the same path pattern on Γ_0 and Γ_1. Specifically, it means at each non-leaf node $B_{0,k}$, each query Q_j makes the same search decision as the one at non-leaf node $B_{1,k}$ (e.g., if query Q_j continues to search the child nodes of node $B_{0,k}$, it will also continue to search the child nodes of node $B_{1,k}$). Since the search decision at each non-leaf node in an R-tree is made by verifying the intersection of two hyper-rectangles without any information on any single-dimension (described in Algorithm 1), the same path pattern based on leakage function $\mathcal{L}$ further indicates

- For $1 \leq j \leq t'$ and for $1 \leq k \leq m$, either $(B_{0,k} \in Q_j) \wedge (B_{1,k} \in Q_j)$, or $(B_{0,k} \notin Q_j) \wedge (B_{1,k} \notin Q_j)$.

For each query $Q_j \in \mathbf{Q}$, its search token $\mathbf{TK}_{Q_j} = (\texttt{TK}_{j,1}, \texttt{TK}_{j,2})$ is computed as

$$\begin{aligned} \texttt{TK}_{j,1} &\leftarrow \texttt{PPE.GenToken}(\texttt{SK}_1, \mathbf{HR}_j), \\ \texttt{TK}_{j,2} &\leftarrow \texttt{PPE.GenToken}(\texttt{SK}_2, \mathbf{HR}'_j); \end{aligned}$$

where $\mathbf{HR}_j$ and $\mathbf{HR}'_j$ are represented via Q_j by following the rules in $\texttt{GenToken}(\mathbf{SK}, Q_j)$.

In **Challenge**, The adversary submits two message sets $\mathbf{M}_0 = \{M_{0,1}, ..., M_{0,n}\}$ and $\mathbf{M}_1 = \{M_{1,1}, ..., M_{1,n}\}$, where for all $1 \leq i \leq n$, $M_{0,i}$ and $M_{1,i}$ have the equal length and are subjected to the following restrictions:

1. If there exist some $j \in [1, t']$ such that $(D_{0,i} \in Q_j) \wedge (D_{1,i} \in Q_j)$, then $M_{0,i} = M_{1,i}$;
2. Otherwise, if for every $j \in [1, t']$, $(D_{0,i} \notin Q_j) \wedge (D_{1,i} \notin Q_j)$, then $M_{0,i} \neq M_{1,i}$.

Then, the challenger flips a coin $b \in \{0, 1\}$, and returns

$$\begin{aligned} & \texttt{PPE.Enc}(\texttt{PK}_1, D_{b,i}, M_{b,i}), \text{ for } 1 \leq i \leq n, \\ & \texttt{PPE.Enc}(\texttt{PK}_2, B_{b,k}, \texttt{Flag}), \text{ for } 1 \leq k \leq m. \end{aligned}$$

Then, in **Phase 2**, the adversary continues to submit a number of t queries subjected to the same restrictions in **Phase 1** and **Challenge**. Finally, the adversary guesses the value of b in **Guess**.

Based on the security of HVE in [10], given two points X_0 and X_1, PPE is selectively secure under the following restriction:

- For $1 \leq j \leq t'$, either $(X_0 \in Q_j) \wedge (X_1 \in Q_j)$, or $(X_0 \notin Q_j) \wedge (X_1 \notin Q_j)$,

which means the advantage of an adversary $\mathcal{A}$ to distinguish this two points is negligible (i.e., $\mathbf{Adv}^{SS}_{\texttt{PPE},\mathcal{A}}(1^\lambda, \Delta) \leq \texttt{negl}(\lambda)$).

As we explained in the security game of Maple, for each pair of $D_{0,i}$ and $D_{1,i}$ (the second restriction in **Phase 1**) or each pair of $B_{0,k}$ and $B_{1,k}$ (the first restriction in **Phase 1**), it satisfies the restriction of the security game of PPE (essentially HVE [9]), then the advantage of an adversary to guess each $D_{b,i}$ or each $B_{b,k}$ is negligible. Since if the adversary can distinguish any pair of ($D_{0,i}$, $D_{1,i}$) or ($B_{0,k}$, $B_{1,k}$), it will be able to distinguish the two sets $\mathbf{D}_0$ and $\mathbf{D}_1$. Therefore, the total advantage of an adversary to guess the value of b in the security game of Maple can be calculated as

$$\begin{aligned} \mathbf{Adv}^{SS}_{\texttt{Maple},\mathcal{A}}(1^\lambda, \Delta) &\leq 1 - (1 - \mathbf{Adv}^{SS}_{\texttt{PPE},\mathcal{A}}(1^\lambda, \Delta))^{n+m} \\ &= 1 - (1 - \texttt{negl}(\lambda))^{n+m} \\ &= 1 - (1 - \texttt{negl}'(\lambda)) \\ &= \texttt{negl}'(\lambda), \end{aligned}$$

where $\texttt{negl}(\lambda)$ and $\texttt{negl}'(\lambda)$ are negligible functions in λ[5]. Therefore, Maple is selectively secure if HVE is selectively secure.

Proof of Theorem 2. Since we already have that for any two points $X_{0,i}$ and $X_{1,i}$ under the restriction of

- For $1 \leq j \leq t'$, either $(X_0 \in Q_j) \wedge (X_1 \in Q_j)$, or $(X_0 \notin Q_j) \wedge (X_1 \notin Q_j)$,

in the selective security game, the advantage of an adversary to distinguish these two points is negligible. Then, with an additional restriction in single-dimensional privacy game:

- And if $(X_{0,i} \notin Q_j) \wedge (X_{1,i} \notin Q_j)$, for some $i \in [1, n]$ and some $j \in [1, t']$, there exists some $k \in [1, w]$, such that $(X_{0,i} \in Q^k_j) \wedge (X_{1,i} \notin Q^k_j)$ or $(X_{0,i} \notin Q^k_j) \wedge (X_{1,i} \in Q^k_j)$,

it does not increase the advantage of the adversary. Therefore, we have

$$\mathbf{Adv}^{SDP}_{\texttt{Maple},\mathcal{A}}(1^\lambda, \Delta) = \mathbf{Adv}^{SS}_{\texttt{Maple},\mathcal{A}}(1^\lambda, \Delta) \leq \texttt{negl}'(\lambda),$$

where $\texttt{negl}'(\lambda)$ is a negligible function in λ. So, Maple is single-dimensionally private if it is selectively secure.

[5]According to the properties of negligible functions presented in Chapter 3 in the textbook [21], for a negligible function $\texttt{negl}(\lambda)$, it is easy to have $(1 - \texttt{negl}(\lambda))^n = 1 - \texttt{negl}(\lambda) \cdot F = 1 - \texttt{negl}'(\lambda)$, where F is a positive polynomial and $\texttt{negl}'(\lambda)$ is still a negligible function in λ.