This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

Achieving Secure and Dynamic Range Queries Over Encrypted Cloud Data

Wei Yang†, *Member, IEEE,* Yangyang Geng†, Lu Li‡†,
Xike Xie†, *Member, IEEE,* and Liusheng Huang†, *Member, IEEE*

Abstract—Cloud computing is motivating data owners to outsource their databases to the cloud. However, for privacy concerns, the sensitive data has to be encrypted before outsourcing, which inevitably posts a challenging task for effective data utilization. Existing work either focuses on keyword searches, or suffers from inadequate security guarantees or inefficiency. In this paper, we concentrate on multi-dimensional range queries over dynamic encrypted cloud data. We first propose a tree-based private range query scheme over dynamic encrypted cloud data (TRQED), which supports faster-than-linear range queries and protects single-dimensional privacy. Then, we discuss the defects of TRQED in terms of privacy-preservation. We modify the framework of the system by adopting a two-server model and put forward a safer range query scheme, called TRQED$^+$. By newly designed secure node query (SNQ) and secure point query (SPQ), we propose the perturbation-based oblivious R-tree traversal (ORT) operation to preserve both path pattern and stronger single-dimensional privacy. Finally, we conduct comprehensive experiments on real-world datasets and perform comparisons with existing works to evaluate the performance of the proposed schemes. Experimental results show that our TRQED and TRQED$^+$ surpass the state-of-the-art methods in privacy-preservation level and efficiency.

Index Terms—Range query, cloud data, data dynamics, privacy protection, R-tree, path pattern.

1 Introduction

With the prevalence of cloud computing, database outsourcing [1] has become a popular service nowadays. In the cloud computing paradigm, a data owner prefers to remotely store their data into the cloud so as to benefit from outsourcing heavy storage and management tasks to the cloud server. This motivates more and more individuals and enterprises to outsource their local databases. Data outsourcing, however, raises serious privacy concerns. Sensitive data, such as emails and medical records, have to be encrypted to protect the privacy before outsourcing [2], [3]. Unfortunately, this introduces new challenges to data utilization. How to enable effective queries over encrypted cloud data while preserving the privacy is one of the most important to be solved.

One trivial approach is to download and decrypt the data locally, which is clearly impractical and contradicts with our motivation. Traditional searchable encryption (SE) schemes [2], [4], [5] have been proposed to support specific types of queries, such as keyword searching [5], [6]. However, considering the equality and inequality conditions specified by the SQL, based on which multi-dimensional range queries are evaluated, traditional SE schemes would not be adequate.

To support multi-dimensional range queries over encrypted data, Boneh *et al.* designed a predicate encryption scheme [7] by using Hidden Vector Encryption (HVE) [8]. The scheme, however, incurs high computation due to heavy reliance on public-key cryptography, and cannot prevent the cloud server from identifying whether two encrypted queries are from the same query, which we refer to as *query privacy*. To improve the efficiency, researchers proposed a symmetric scheme LSED$^+$ [9] and an $\widehat{R}$-tree scheme [10] by decomposing a multi-dimensional range query into multiple single-dimensional queries. However, both schemes suffer from *single-dimensional privacy* leakage, meaning the server would learn the relationship between every single-dimensional query and its corresponding query results. To solve this problem, Wang *et al.* [11] proposed a scheme Maple based on HVE. However, a public-key scheme, Maple incurs heavy computation cost, and fails to preserve the query privacy. Moreover, Maple cannot support dynamic operations, *i.e.*, *insertion*, *deletion* and *modification*. In summary, all aforementioned multi-dimensional range query solutions either suffer from inadequate privacy guarantees, *i.e.*, query privacy and single-dimensional privacy or fail in efficiency. Moreover, most works achieving faster-than-linear search do not support dynamic operations. Thus, the problem of private range query over dynamic encrypted cloud data remains open to date.

Our work focuses on addressing the problem of multi-dimensional private range queries over dynamic encrypted cloud data. Specifically, our design goals are as follows.

- **Multi-dimensional range query.** The design should enable multi-dimensional range queries over encrypted data, and return the correct results.
- **Data dynamics support.** The design supports data update including insertion, deletion and modification.
- **Privacy-preservation.** The design should prevent the cloud server from learning additional sensitive information from database, index and query. Especially, the privacy requirements include *Data privacy*, *Index privacy*, *Query privacy*, *Single-dimensional privacy* and *Path pattern privacy*.
- **High efficiency.** The above goals should be achieved with low system overhead and high efficiency.

In our previous work [12], we firstly proposed a Tree-based

- † *School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China, and also with the Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China. E-mail: {qubit, lshuang, xkxie}@ustc.edu.cn; geng325@mail.ustc.edu.cn.*
- ‡ *School of Information Engineering, Yancheng Teachers University, Yancheng 224007, China. E-mail: lil@yctu.edu.cn*

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

private Range Query scheme over dynamic Encrypted cloud Data (TRQED), which enables efficient range queries without revealing sensitive information to the cloud server. In TRQED, we proposed the perturbation-based *inner product comparison* (IPC) and extended dimensions to convert the same query to varied query vectors and obfuscate the cloud server's view, which can preserve query privacy and single-dimensional privacy. To reduce the system overhead, we also designed the lightweight *point intersection predicate encryption* (PIPE) and the *range intersection predicate encryption* (RIPE), thus transforming the query processing into an efficient traversal of a tree by checking the relations between the given range query and tree nodes. Our contributions in the previous work [12] are summarized as follows:

- We present TRQED, the first private range query scheme which achieves faster-than-linear search and supports data dynamics while preserving the query privacy and single-dimensional privacy simultaneously.
- We use the perturbation-based IPC and extended dimensions to preserve the query privacy and single-dimensional privacy, and propose two building blocks PIPE and RIPE to enable efficient range query.
- We theoretically prove that our scheme is secure against semi-honest adversaries under known background model.

In TRQED, however, we did not cover path pattern privacy [11] which can leak out useful information to the cloud server. Besides, TRQED cannot fully preserve single-dimensional privacy against powerful statistical attacks. Based on the TRQED scheme, we redesign the system framework and make a trade-off between efficiency and security to further enhance privacy protection level. To preserve path pattern and single-dimensional privacy, we perform a range query by using two cloud servers to traverse R-tree collaboratively. Moreover, we exploit *secure scalar product* (SSP) protocol and develop *oblivious R-tree traversal* (ORT), which mainly consists of *secure node query* (SNQ) and *secure point query* (SPQ), to prevent the two cloud servers from inferring each other's sensitive information.

Compared with the preliminary conference version [12] of this paper, we have modified the architecture of system model in TRQED to support stronger privacy guarantees. To preserve path pattern and stronger single-dimensional privacy, we have adopted a two-server model and designed oblivious R-tree traversal operation. The enhanced version of TRQED is called TRQED$^+$. In summary, the paper substantially extends our previous work as follows:

- We modify the framework of the system by adopting two-server model and put forward TRQED$^+$, a security-enhanced version of TRQED. Compared with TRQED, TRQED$^+$ additionally provides stronger single-dimensional privacy preservation and path pattern preservation.
- We propose the perturbation-based oblivious R-tree traversal operation, which includes SNQ and SPQ. Oblivious traversal allows us to preserve path pattern privacy during R-tree traversal. Moreover, we achieve stronger single-dimensional privacy preservation by exploiting SSP protocol in SNQ and SPQ.
- To evaluate the performance of TRQED and TRQED$^+$, we conduct extensive experiments on real-world datasets. Evaluation shows that TRQED is more efficient than existing works, and for our safer TRQED$^+$, the overhead on computation and communication are affordable.

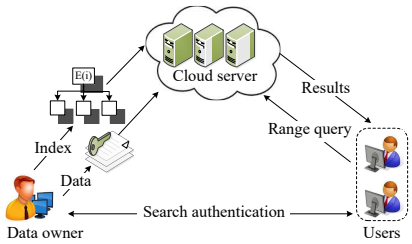


Fig. 1: Architecture of private range query over encrypted cloud data

We organize this paper in the following manner. In Section 2, we outline the system model, threat models, design goals, and primary notations. Section 3 depicts the basic design of IPE, which includes PIPE and RIPE. Then, we detailedly present the TRQED and TRQED$^+$ in Sections 4 and 5, respectively. We show the simulation results and introduce the related works on searchable encryption and range queries in Sections 6 and 7. Finally, we conclude this paper in Section 8.

2 Problem Formulation

2.1 System Model

The system model we consider in this paper consists of three entities: a *data owner*, a *cloud server* and multiple authorized *users*, as depicted in Fig. 1. The data owner wants to outsource its database (*e.g.*, a large set of data records) to the cloud in order to reduce the management cost. To protect the privacy, the sensitive data is encrypted before outsourcing. For queries over encrypted data, the data owner builds a searchable index with a multi-dimensional index, and encrypts the index. Afterwards, the owner outsources the encrypted database and index to the cloud server. Users have mutual authentication capability with the data owner, and can search the encrypted cloud data by submitting an encrypted range query to the server. Once receiving the search token, the cloud server searches the tree-based index and returns the set of matched data records.

Herein, we employ R-tree [13] to build the index. R-tree is a type of multi-dimensional data structure, which represents each node is with a Minimum Bounding Rectangle (MBR), and nearby objects are grouped in the same layer. In our work, each data record is essentially an element of points set of leaf node. We consider each node of R-tree has the form (R, p), where R represents an MBR. A leaf-node points to a set of (encrypted) multi-dimensional points that its MBR covers. Closely located leaf-nodes are further grouped into an upper-level node. The process is repeated recursively until the root node is derived. Thus, an index node stores a set of pointers for its children. In particular, the pointers of a leaf node are for the multi-dimensional points, and the pointers of a non-leaf node are for its child nodes.

2.2 Threat Model

We consider a *semi-honest* (also known as *honest-but-curious* [5], [10], [11], [14], [15]) cloud server in the threat model, which is the foundation of designing secure protocols against malicious adversaries [16]. That is, the server follows the protocol honestly, but it is curious to record all intermediate results and try to deduce useful information about the data and queries. We assume the authorized users are trusted by the data owner and do not collude with the server.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

In terms of the level of privacy protection, there are two kinds of threat models, namely *Known ciphertext model* and *Known background model* with respect to the assumption of the cloud servers ability. Wang *et al.* [17] adopt the former model to provide security for privacy-preserving range queries over encrypted cloud data. Herein, we adopt the latter, namely *Known background model*, which is also considered in some existing works [5], [6]. In this model, the cloud server is supposed to not only know the encrypted dataset and index as in the former two models, but also possess more knowledge such as the distribution of queries. As an instance of possible attacks, the server could launch frequency analysis attacks combined with the background information to deduce useful information.

2.3 Privacy Requirements

Before proposing a secure query scheme, we need to clearly understand the definition of various privacy requirements, which are related to different data stakeholders [18].

- **Data privacy** and **Index privacy**. The data owner holds the original database, and builds a tree-based index to organize data. Both data and index involve sensitive information of the owner. Therefore, a secure scheme should ensure that the cloud server cannot infer exact information from encrypted data and index.
- **Query privacy**. The authorized users submit encrypted search token to the cloud server, and the latter executes query task. However, the search token and search results can not reveal the specific query range of the authorized user.
- **Single-dimensional privacy** and **Path pattern**. During the search process, the cloud server can not obtain single-dimensional matching results and search paths from the intermediate results.

2.4 Notations

We use the following primary notations and definitions in the rest of the paper.

- $\mathcal{D}$: the set of n data records $\mathcal{D} = \{D_1, \cdots, D_n\}$.
- $\mathcal{D}^*$: the encrypted data set $\mathcal{D}^* = \{D_1^*, \cdots, D_n^*\}$.
- $\mathcal{A}$: the set of w numerical attributes, denoted as $\mathcal{A} = \{A_1, \cdots, A_w\}$.
- D_i: a data record, denoted as a point $D_i = \{d_1, \cdots, d_w\}$, where d_j is a value in the j-th dimension.
- $\mathcal{I}$: the R-tree index of $\mathcal{D}$. $\mathcal{I} = \{D_1, \cdots, D_n; N_1, \cdots, N_m\}$, where N_j is a node, and D_i is a data point associated with a leaf node.
- $\mathcal{I}^*$: the encrypted index. $\mathcal{I}^* = \{C_1, \cdots, C_n; E_1, \cdots, E_m\}$, where E_j is an encrypted node, and C_i is an encrypted data point.
- Q: a range query over any subset of w' attributes, $w' \leq w$, denoted as an $\mathsf{MBR} = (R_1, \cdots, R_w)$, where R_i is denoted as a range $\{(x_{i,l}, x_{i,r})\}$ in the i-th dimension.
- ID_Q: the query result returned as a set of data record identifiers.

Each numeric attribute A_i represents a dimension. Note that $\mathcal{D}$ and $\mathcal{I}$ are encrypted separately, and the encrypted data point C_i is the encrypted index for data record D_i, and stores the location of the encrypted record D_i^*.

3 Intersection Predicate Encryption

The range queries over R-tree can be implemented by checking the relations that whether a point (*i.e.*, a data record C_i) is inside an MBR (*i.e.*, a range query Q) or whether an MBR E_i intersects with the given query region Q. In this section, we first present a method called *inner product comparison* (IPC) to convert a range query into inner product computation in the ciphertext domain, and design two predicate encryptions to determine the geometric relationships through extended usage of SSW [19] and secure kNN (k-nearest neighbor) computation [20].

3.1 Inner Product Comparison

3.1.1 Comparison Predicate

In the predicate query model [9], the users send predicate functions p as queries to the cloud server. The server searches the tuples of which the attribute values d satisfies $p(d) = true$ and then returns the query results. Herein, for a one-dimensional range query $[x_l, x_r]$ and a point d, we propose a function $p : \mathbb{R}^+ \rightarrow \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers as:

$$p(d) = (d - x_l)(d - x_r). \tag{1}$$

There are three types of relationships between point d and range $[x_l, x_r]$: $d < x_l$, $d > x_r$ and $x_l \leq d \leq x_r$. For the former two types, $(d - x_l)(d - x_r) > 0$, and for the third type, $(d - x_l)(d - x_r) \leq 0$. Therefore, the predicate function regards $p(d) \leq 0$ as true iff d is in the closed range $[x_l, x_r]$ and false otherwise.

To preserve privacy, we add perturbations δ to the predicate $p(d)$ as:

$$\widetilde{p}(d) = (d - x_l)(d - x_r)(d + \delta), \tag{2}$$

where δ is a κ bit positive number (κ is a given security parameter). δ is much larger than $|d|$ so that $d + \delta$ must be a positive number. Therefore, $\widetilde{p}(d) \leq 0$ iff $x_l \leq d \leq x_r$. As such, the statistical information of transformed queries are different from that of plain ones, and the perturbation-based predicate can find the correct tuples of which $d \in [x_l, x_r]$.

3.1.2 Vector Extraction

We transform the predicate $\widetilde{p}(d)$ into query and value vectors. According to the expansion form of Eq. (2), we define the query vector $\boldsymbol{p}$ and value vector $\boldsymbol{v}$ as:

$$\boldsymbol{p} = \begin{pmatrix} 1 \\ -x_l - x_r + \delta \\ x_l x_r - x_l\delta - x_r\delta \\ x_l x_r \delta \end{pmatrix}, \quad \boldsymbol{v} = (d^3, d^2, d, 1)^T$$

As such, the inner product is $\langle \boldsymbol{p}, \boldsymbol{v} \rangle = \widetilde{p}(d)$, and $\langle \boldsymbol{p}, \boldsymbol{v} \rangle \leq 0$ is true iff $d \in [x_l, x_r]$.

3.1.3 Vector Encryption

We employ the matrix-based encryption to encrypt the vectors, which enables the cloud server to process the encrypted queries without revealing sensitive information. The secret key for vector encryption is a four-dimensional invertible matrix. Alternatively, we can also adopt a ϕ-dimensional matrix where $\phi > 4$ as the key, and add dummy values into additional $(\phi - 4)$ dimensions which satisfy that $\sum_{i=5}^{\phi} p_i v_i = 0$. For simplicity, we herein adopt a four-dimensional invertible matrix M to encrypt $\boldsymbol{p}$. Correspondingly,

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

the encryption key for $\boldsymbol{v}$ is $\widetilde{M} = |det(M)|M^{-1}$, where $det(M)$ is the determinant of M as follows:

$$E_M(\boldsymbol{p}) = r_p M^T \boldsymbol{p} \tag{3}$$

$$E_{\widetilde{M}}(\boldsymbol{v}) = r_v |det(M)| M^{-1} \boldsymbol{v} \tag{4}$$

where r_p and r_v are randomly generated positive numbers.

In the ciphertext domain, the query processing is to find the tuples of which the encrypted value vector $E_{\widetilde{M}}(\boldsymbol{v})$ satisfies $\langle E_M(\boldsymbol{p}), E_{\widetilde{M}}(\boldsymbol{v})\rangle \leq 0$, which is computed as

$$\langle E_M(\boldsymbol{p}), E_{\widetilde{M}}(\boldsymbol{v})\rangle = r_p r_v |det(M)| \langle \boldsymbol{p}, \boldsymbol{v}\rangle. \tag{5}$$

As $r_p r_v |det(M)| > 0$, $\langle E_M(\boldsymbol{p}), E_{\widetilde{M}}(\boldsymbol{v})\rangle$ depends on the $\langle \boldsymbol{p}, \boldsymbol{v}\rangle$. Thus, $\langle E_M(\boldsymbol{p}), E_{\widetilde{M}}(\boldsymbol{v})\rangle \leq 0$ is true iff $d \in [x_l, x_r]$.

According to IPC, we can estimate the relationship between point d and range $[x_l, x_r]$ without decryption. Next, we design two secure and efficient building blocks to check whether a value d is inside a range MBR and whether two MBRs intersect.

3.2 Point Intersection Predicate Encryption (PIPE)

Based on IPC, we introduce a *Point Intersection Predicate Encryption* (PIPE) to check whether a value d is inside a range MBR. Inspired by the predicate encryptions in [9], [11], the 1-dimensional PIPE (1dPIPE) is detailed as follows:

- Setup(1^κ). On input a security parameter κ, output a secret key SK.
- Encrypt(SK, d). On input SK and a value d, output ciphertext $C = r_v \widetilde{M} \boldsymbol{v}$, where $\boldsymbol{v} = (d^3, d^2, d, 1)^T$ and r_v is a generated random positive number.
- GenToken(SK, R). On input SK and a range $R = [x_l, x_r]$, output search token $TK = r_p M^T \boldsymbol{p}$, where $\boldsymbol{p} = (1, -x_l - x_r + \delta, x_l x_r - x_l\delta - x_r\delta, x_l x_r \delta)^T$, and r_p is a random positive number.
- Query(TK, C). On input TK and C, output 1 iff $\langle TK, C\rangle \leq 0$ and 0 otherwise.

We now extend the 1dPIPE to the multi-dimensional case and construct the w-dimensional point predicate encryption (wdPIPE) to check whether a multi-dimensional point D is inside a query range. It is based on the geometric relationship that if a point is inside a query range, the value of this point in every dimension is inside the corresponding single-dimension range of the MBR, and vice versa:

$$D \in \mathsf{MBR} \Leftrightarrow \{d_i \in R_i\}, \forall i \in [1, w]. \tag{6}$$

where the point $D = (d_1, \cdots, d_w)$ and query range $\mathsf{MBR} = (R_1, \cdots, R_w)$. The details of wdPIPE are presented in Alg. 1. Note that the dimension w is extended to $2w$ dimensions in the inner product queries, of which scalar products of the added dimensions are below zero, to obfuscate the cloud server's view and preserve the single-dimensional privacy. It is worth emphasizing that Ω_P is a permutation function, which is generated randomly by the data owner, to obfuscate the order of $2w$ dimensions.

Correctness. The inner product for $i \leq w$ is

$$\langle \boldsymbol{tk}_i, \boldsymbol{c}_i\rangle = r_p r_v |det(M)| \langle \boldsymbol{p}_i, \boldsymbol{v}_i\rangle.$$

The inner product for dimension $i > w$ is

$$\langle \boldsymbol{tk}_i, \boldsymbol{c}_i\rangle = r_p r_v |det(M)| \sum r_{i,j} r_{i,j}^{'},$$

where $r_{i,j} > 0, r_{i,j}^{'} < 0$ for $1 \leq j \leq 4$, which guarantees the inner product below 0. Thus, if $D \in \mathsf{MBR}$, $\langle \boldsymbol{tk}_i, \boldsymbol{c}_i\rangle \leq 0$ $\forall i \in [1, 2w]$. The probability of returning the point is negligible if D is not inside the range.

Algorithm 1 wdPIPE

- Setup(1^κ). On input a security parameter κ, output a secret key $SK = \{M, \widetilde{M}, \Omega_P\}$.
- Encrypt(SK, D). On input SK and a point $D = (d_1, \cdots, d_w)$, output ciphertext $C = r_v(\Omega_P([\widetilde{M}\boldsymbol{v}_1, \cdots, \widetilde{M}\boldsymbol{v}_{2w}]))$, where r_v is a random positive number, and for $i \in [1, 2w]$,

$$\boldsymbol{v}_i = \begin{cases} (d_i^3,\ d_i^2,\ d_i,\ 1)^T, & i \leq w \\ (r_{i,1},\ r_{i,2},\ r_{i,3},\ r_{i,4})^T, & i > w \end{cases}$$

  where $r_{i,j}$ is a random positive number for $1 \leq j \leq 4$.
- GenToken(SK, MBR). On input SK and a range query $\mathsf{MBR} = (R_1, \cdots, R_w)$, where $R_k = [x_{k,l}, x_{k,r}]$, output search token $TK = r_p(\Omega_P([M^T\boldsymbol{p}_1, \cdots, M^T\boldsymbol{p}_{2w}]))$, where r_p is a random positive number, and for $i \in [1, 2w]$,

$$\boldsymbol{p}_i = \begin{cases} \begin{pmatrix} 1 \\ -x_{i,l} - x_{i,r} + \delta \\ x_{i,l}x_{i,r} - (x_{i,l} + x_{i,r})\delta \\ x_{i,l}x_{i,r}\delta \end{pmatrix}, & i \leq w \\ (r_{i,1}^{'},\ r_{i,2}^{'},\ r_{i,3}^{'},\ r_{i,4}^{'})^T, & i > w \end{cases}$$

  where $r_{i,j}^{'}$ is a random negative number for $1 \leq j \leq 4$.
- Query(TK, C). On input TK and C, output 1 iff $\langle \boldsymbol{tk}_i, \boldsymbol{c}_i\rangle \leq 0, \forall i \in [1, 2w]$, or output 0 otherwise.

Example. We show the correctness by an instance. For simplicity, we set $\Omega_P([1, 2, 3, 4]) = [4, 1, 2, 3]$ and the pair $\{M, \widetilde{M}\}$:

$$M = \begin{pmatrix} 1 & 4 & -1 & -1 \\ 1 & -2 & -1 & 1 \\ -3 & 3 & -4 & -2 \\ 0 & 1 & -1 & -1 \end{pmatrix}, \quad \widetilde{M} = \begin{pmatrix} 4 & 6 & -6 & 14 \\ 8 & -2 & 2 & -14 \\ -2 & -10 & -4 & 0 \\ 10 & 8 & 6 & -42 \end{pmatrix}$$

We have two points (*i.e.*, $D_1 = (1, 2)$ and $D_2 = (3, 4)$) and a range query (*i.e.*, $\mathsf{MBR} = ([2, 4], [1, 5])$). We encrypt D_1 and D_2 by using Encrypt(SK, D). Assume that $r_v = 9$ for D_1 and $r_v = 2$ for D_2. Then we can get C_1 and C_2:

$$C_1 = \begin{pmatrix} 522 & 162 & 522 & 468 \\ 162 & -54 & 414 & -72 \\ -828 & -144 & -576 & -990 \\ 234 & -162 & 738 & 414 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 184 & 316 & 684 & 324 \\ 312 & 380 & 948 & -52 \\ -176 & -312 & -608 & -372 \\ 320 & 636 & 1500 & -16 \end{pmatrix}$$

We encrypt the range query MBR by using GenToken(SK, MBR). Assume that $r_p = 3$ and $\delta = 77$, we can get a search token TK:

$$TK = \begin{pmatrix} 738 & 4302 & 4329 & 1587 \\ -1800 & -2652 & -3372 & -2112 \\ 1932 & 3384 & 4113 & 2823 \\ 1338 & 1086 & 1797 & 1569 \end{pmatrix}$$

By executing Query(TK, C), we can easily get the following result: Query$(TK, C_1) \rightarrow 0$, Query$(TK, C_2) \rightarrow 1$. Obviously,

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

the result is in line with the fact, *i.e.*, $D_1 \notin \mathsf{MBR}$ and $D_2 \in \mathsf{MBR}$.

3.3 Range Intersection Predicate Encryption (RIPE)

Based on IPC, we propose a new *Range Intersection Predicate Encryption* (RIPE) to check whether two MBRs intersect. We begin with the 1-dimensional case. For two ranges $R = [x_l, x_r]$ and $R^{'} = [x_l^{'}, x_r^{'}]$, we find that there exists a true proposition:

$$R \cap R^{'} \neq \varnothing \Leftrightarrow p = (x_l - x_r^{'})(x_r - x_l^{'}) \leq 0$$

Similarly, add the perturbation to the predicate, and we can obtain

$$\widehat{p} = (x_l - x_r^{'})(x_r - x_l^{'})(x_r + \delta) \tag{7}$$

where δ is also a positive random number. In the same way, we express the predicate as query and value vectors as follows:

$$\boldsymbol{p} = \begin{pmatrix} x_r + \delta \\ -x_r^2 - x_r\delta \\ -x_l x_r - x_l\delta \\ x_l x_r (x_r + \delta) \end{pmatrix}, \quad \boldsymbol{v} = (x_l^{'}x_r^{'}, x_r^{'}, x_l^{'}, 1)^T$$

In the same way, we design the 1dRIPE, and further extend it to the multi-dimensional case. As shown in Alg. 2, the wdRIPE is to check whether two multi-dimensional ranges intersect based on the fact that if two MBRs intersect, the range of each dimension will intersect, and vice versa:

$$\mathsf{MBR} \cap \mathsf{MBR}^{'} \neq \varnothing \Leftrightarrow p_i = (x_{i,l} - x_{i,r}^{'})(x_{i,r} - x_{i,l}^{'}) \leq 0$$

where $\mathsf{MBR} = (R_1, \cdots, R_w)$, $R_i = [x_{i,l}, x_{i,r}]$, $\mathsf{MBR}^{'} = (R_1^{'}, \cdots, R_w^{'})$, and $R_i^{'} = [x_{i,l}^{'}, x_{i,r}^{'}], \forall i \in [1, w]$.

Using IPC, the correctness proof of wdRIPE is similar to that of wdPIPE.

Example. We use the same $\{M, \widetilde{M}\}$ as in Subsection 3.2, and set $\Omega_R([1,2,3,4]) = [2,3,4,1]$. We have two MBRs (*i.e.*, $\mathsf{MBR}_1^{'} = ([1,4],[1,3])$ and $\mathsf{MBR}_2^{'} = ([5,6],[2,3])$) and a range query (*i.e.*, $\mathsf{MBR} = ([2,4],[1,5])$). We encrypt $\mathsf{MBR}_1^{'}$ and $\mathsf{MBR}_2^{'}$ by using $\mathsf{Encrypt}(SK, \mathsf{MBR}^{'})$. Assume that $r_v = 4$ for $\mathsf{MBR}_1^{'}$ and $r_v = 3$ for $\mathsf{MBR}_2^{'}$, then we can get C_1 and C_2:

$$C_1 = \begin{pmatrix} 152 & 368 & 216 & 192 \\ 24 & -272 & 40 & 48 \\ -160 & -128 & -360 & -208 \\ 72 & -928 & 64 & 144 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 132 & 402 & 252 & 420 \\ 96 & 6 & -168 & 672 \\ -150 & -474 & -336 & -420 \\ 162 & -108 & -336 & 1008 \end{pmatrix}$$

We encrypt the range query MBR by using $\mathsf{GenToken}(SK, \mathsf{MBR})$. Assume that $r_p = 5$ and $\delta = 55$, we can get a search token TK:

$$TK = \begin{pmatrix} -300 & 325 & 185 & 885 \\ 4800 & -855 & -2710 & 4130 \\ 900 & 1230 & 2125 & 885 \\ 2700 & 450 & 1175 & -2655 \end{pmatrix}$$

By executing $\mathsf{Query}(TK, C)$, we can easily get the following result: $\mathsf{Query}(TK, C_1) \rightarrow 1$, $\mathsf{Query}(TK, C_2) \rightarrow 0$. Obviously, the result is in line with the fact, *i.e.*, $\mathsf{MBR} \cap \mathsf{MBR}_1^{'} \neq \varnothing$ and $\mathsf{MBR} \cap \mathsf{MBR}_2^{'} = \varnothing$.

Algorithm 2 wdRIPE

- $\mathsf{Setup}(1^\kappa)$. On input a security parameter κ, output a secret key $SK = \{M, \widetilde{M}, \Omega_R\}$.
- $\mathsf{Encrypt}(SK, \mathsf{MBR}^{'})$. On input SK and an $\mathsf{MBR}^{'} = (R_1^{'}, \cdots, R_w^{'})$, where $R_k^{'} = [x_{k,l}^{'}, x_{k,r}^{'}]$, output ciphertext $C = r_v(\Omega_R([\widetilde{M}\boldsymbol{v}_1, \cdots, \widetilde{M}\boldsymbol{v}_{2w}]))$, where r_v is a generated random positive number, for $i \in [1, 2w]$,

$$\boldsymbol{v}_i = \begin{cases} (x_{i,l}^{'}x_{i,r}^{'}, \ x_{i,r}^{'}, \ x_{i,l}^{'}, \ 1)^T, & i \leq w \\ (r_{i,1}, \ r_{i,2}, \ r_{i,3}, \ r_{i,4})^T, & i > w \end{cases}$$

  where $r_{i,j}$ is a random positive number for $1 \leq j \leq 4$.
- $\mathsf{GenToken}(SK, \mathsf{MBR})$. On input SK and a range query $\mathsf{MBR} = (R_1, \cdots, R_w)$, where $R_k = [x_{k,l}, x_{k,r}]$, output search token $TK = r_p(\Omega_R([M^T\boldsymbol{p}_1, \cdots, M^T\boldsymbol{p}_{2w}]))$, where r_p is a random positive number, for $i \in [1, 2w]$,

$$\boldsymbol{p}_i = \begin{cases} \begin{pmatrix} x_{i,r} + \delta \\ -x_{i,r}^2 - x_{i,r}\delta \\ -x_{i,l}x_{i,r} - x_{i,l}\delta \\ x_{i,l}x_{i,r}(x_{i,r} + \delta) \end{pmatrix}, & i \leq w \\ (r_{i,1}^{'}, \ r_{i,2}^{'}, \ r_{i,3}^{'}, \ r_{i,4}^{'})^T, & i > w \end{cases}$$

  where $r_{i,j}^{'}$ is a random negative number for $1 \leq j \leq 4$.
- $\mathsf{Query}(TK, C)$. On input TK and C, output 1 iff $\langle \boldsymbol{tk}_i, \boldsymbol{c}_i \rangle \leq 0, \forall i \in [1, 2w]$, or output 0 otherwise.

4 Tree-based Private Range Query over Encrypted Cloud Data

Based on the building blocks PIPE and RIPE, we can present our TRQED formally.

4.1 TRQED Construction

To perform private range query, TRQED searches the R-tree index through the four polynomial algorithms (Setup, $\mathsf{EncIndex}$, $\mathsf{GenToken}$, Query), which are presented as follows.

- $\mathsf{Setup}(1^\kappa)$. On input a security parameter κ, the data owner generates a secret key $\boldsymbol{SK} = \{SK_{point}, SK_{node}, SK_{data}\}$, where

$$SK_{point} \leftarrow w\text{dPIPE.}\mathsf{Setup}(1^\kappa),$$
$$SK_{node} \leftarrow w\text{dRIPE.}\mathsf{Setup}(1^\kappa),$$
$$SK_{data} \leftarrow AES(1^\kappa).$$

- $\mathsf{EncIndex}(\boldsymbol{SK}, \mathcal{D})$. On input the $\boldsymbol{SK}$ and dataset $\mathcal{D}$, the data owner builds an R-tree $\mathcal{I} = \{D_1, \cdots, D_n, N_1, \cdots, N_m\}$, where $\{D_i\}_{i=1}^n$ is a data point associated with a leaf node, and $\{N_j\}_{j=1}^m$ is a node. Then data owner encrypts every node and data point separately as follows:

$$C_i \leftarrow w\text{dPIPE.}\mathsf{Encrypt}(SK_{point}, D_i),$$
$$E_j \leftarrow w\text{dRIPE.}\mathsf{Encrypt}(SK_{node}, N_j).$$

  where $\{C_i\}_{i=1}^n$ is an encrypted data point, and $\{E_j\}_{j=1}^m$ is an encrypted node. The data records are encrypted as follows:

$$D_i^* \leftarrow AES.encrypt(SK_{data}, D_i), \text{for } 1 \leq i \leq n.$$

  Afterwards, the data owner outsources the encrypted index $\mathcal{I}^* = \{C_1, \cdots, C_n, E_1, \cdots, E_m\}$ and dataset $\mathcal{D}^*$ to the cloud server.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

- GenToken($\boldsymbol{SK}$, Q). On input the $\boldsymbol{SK}$ and a range query Q, the owner generates a search token $\boldsymbol{TK} = \{TK_{point}, TK_{node}\}$, where

$$TK_{point} \leftarrow w\text{dPIPE.GenToken}(SK_{point}, Q),$$
$$TK_{node} \leftarrow w\text{dRIPE.GenToken}(SK_{node}, Q).$$

  Then $\boldsymbol{TK}$ is distributed to authorized users and submitted to the cloud.
- Query($\boldsymbol{TK}, \mathcal{I}^*$). On input the $\boldsymbol{TK}$ and $\mathcal{I}^*$, the cloud server searches tree $\mathcal{I}^*$ level-by-level as follows:
  - For a non-leaf node E_j, if wdRIPE.Query(TK_{node}, E_j) is true, it continues to search the child nodes of E_j; otherwise it stops searching this branch.
    For a leaf node $E_{j'}$, if wdRIPE.Query(TK_{node}, $E_{j'}$) is true, it continues to search the data points set associated with the node $E_{j'}$; otherwise it stops searching this branch.
  - For a data point C_i, if wdPIPE. Query(TK_{point}, C_i) is true, that indicates $D_i \in Q$, and it puts the identifier I_i of this point into the list ID_Q.

  Finally, the server returns a set ID_Q of identifiers of matched data records, and the user obtains the results.

Correctness. The correctness of TRQED is based on the correctness of wdPIPE and wdRIPE. Formally, for all $\kappa \in \mathbb{N}$, let $SK \xleftarrow{R}$ Setup(1^κ), $\mathcal{I}^* \xleftarrow{R}$ EncIndex($\boldsymbol{SK}, \mathcal{D}$), $TK \xleftarrow{R}$ GenToken($\boldsymbol{SK}, Q$). If $D_i \in Q$,

$$Pr[\text{Query}(\boldsymbol{TK}, \mathcal{I}^*) = ID_Q, I_i \in ID_Q] = 1.$$

If $D_i \notin Q$,

$$Pr[\text{Query}(\boldsymbol{TK}, \mathcal{I}^*) = ID_Q, I_i \in ID_Q] \leq negl(\kappa),$$

where $negl(\kappa)$ is a negligible function in κ. Thus, Query($\boldsymbol{TK}$, $\mathcal{I}^*$) returns I_i if and only if $D_i \in Q$.

Example. Fig. 2 depicts an example of R-tree and its encryption. The data owner builds an R-tree $\mathcal{I} = \{D_1, \cdots, D_8, N_1, \cdots, N_6\}$, and uses the secret key to encrypt it. For a range query Q submitted by the user, the data owner sends corresponding search token $\boldsymbol{TK} = \{TK_{point}, TK_{node}\}$ to the user. The user submits $\boldsymbol{TK}$ to the cloud, and the latter executes Query($\boldsymbol{TK}$, $\mathcal{I}^*$). We assume that the traversal process on the unencrypted R-tree $\mathcal{I}$ is $\{N_2\} \rightarrow \{N_5, N_6\} \rightarrow \{D_6, D_7\}$. Based on the correctness of wdPIPE and wdRIPE, the traversal process on $\mathcal{I}^*$ is $\{E_2\} \rightarrow \{E_5, E_6\} \rightarrow \{C_6, C_7\}$, *i.e.*, Query($\boldsymbol{TK}, \mathcal{I}^*$) returns $ID_Q = \{I_6, I_7\}$. Then, the cloud server sends $\{D_6^*, D_7^*\}$ to the user, and the latter uses SK_{data} to decrypt $\{D_6^*, D_7^*\}$ to obtain data records.

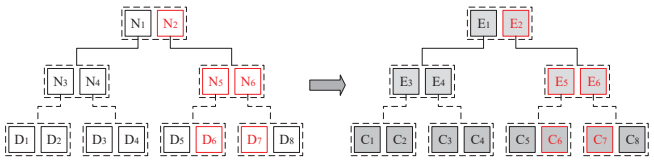


Fig. 2: An example of R-tree and its encryption

4.2 Data Update

TRQED can support basic data dynamics in range query processing. We assume that each time one record update is processed, and the fan-out of the R-tree is big enough so that TRQED does not split or merge a non-leaf node in the tree. Let us recall the data layout in R-tree [13]. R-tree organizes data in pages, *i.e.*, nodes correspond to disk pages. The maximum number of entries in each node (*i.e.*, *fan-out*) is approximately equal to $\frac{PAGE\_SIZE}{ENTRY\_SIZE}$. Currently, $PAGE\_SIZE$ is typically 4096 bytes. Therefore, the above assumption is reasonable.

4.2.1 Insertion

To insert a new record $\bar{D}$, we should find the leaf node of R-tree that contains this tuple. That is, the algorithm needs to traverse the index by testing whether a point (*i.e.*, $\bar{D}$) is inside an MBR of the node, which is based on equality query. Roughly speaking, equality query is a specific type of range query. For a new data record $\bar{D} = (\bar{d}_1, \cdots, \bar{d}_w)$, it can be expressed as a range $\overline{\text{MBR}} = (\bar{R}_1, \cdots, \bar{R}_w)$, where $\bar{R}_j = [\bar{d}_j, \bar{d}_j]$, for $1 \leq j \leq w$. Then the data owner computes an update token $\boldsymbol{TK}_\pi$ with GenToken($SK, \overline{\text{MBR}}$) of wdPIPE and wdRIPE. After receiving the token and $\bar{C}$ generated by wdPIPE.Encrypt($SK_{point}, \bar{D}$), the cloud server searches the R-tree until it finds the leaf node $\bar{E}$ that contains $\bar{D}$. Afterwards, the server just inserts this tuple into the points set of leaf node $\bar{E}$, and puts the encrypted record into $\mathcal{D}^*$.

4.2.2 Deletion

To delete a data record, the algorithm should search for its location in the R-tree index. Specifically, it first finds the leaf node that contains the record, and then checks whether a data point is equal to the record. Given the record $D^{'}$, the owner submits the deletion token $\boldsymbol{TK}_\pi$ similarly with the above $\overline{\text{MBR}}$. The token consists of $(\overline{TK}_{point}, \overline{TK}_{node})$, which enables the server to locate the record $\bar{D}$. Then the cloud server deletes $\bar{D}$, and deletes the corresponding encrypted record.

4.2.3 Modification

Data modification is considered as a combination of deletion of old value and insertion of new value [9]. The server just needs to perform one deletion and insertion to modify a record, and thus we omit the details here for brevity.

The UpdToken and Update are defined as follow:

- UpdToken($\boldsymbol{SK}, \pi, \overline{\text{MBR}}$). On input $\boldsymbol{SK}$, update option $\pi \in \{$*insertion, deletion, modification*$\}$, and a target record $\overline{\text{MBR}} = (\bar{R}_1, \cdots, \bar{R}_w)$, where $\bar{R}_j = [\bar{d}_j, \bar{d}_j]$, the owner outputs a token $\boldsymbol{TK}_\pi = (\overline{TK}_{point}, \overline{TK}_{node})$, where

$$\overline{TK}_{point} \leftarrow w\text{dPIPE.GenToken}(SK_{point}, \overline{\text{MBR}}),$$
$$\overline{TK}_{node} \leftarrow w\text{dRIPE.GenToken}(SK_{node}, \overline{\text{MBR}}).$$

- Update($\boldsymbol{TK}_\pi$, $\mathcal{I}^*$, $\bar{C}$). On input the token $\boldsymbol{TK}_\pi$, the index $\mathcal{I}^*$, and ciphertext of the record $\bar{C}$, where $\bar{C} = w$dPIPE.Encrypt($SK_{point}, \bar{D}$), the cloud server outputs a new $\mathcal{I}^*$ and updates the $\mathcal{D}^*$.
  - If $\pi =$*insertion*, call Query($\overline{TK}_{node}$, $\mathcal{I}^*$) and find the leaf node $\bar{E}$ where $\bar{C} \in \bar{E}$. Insert the data point $\bar{C}$ in the points set of leaf node $\bar{E}$, and add encrypted record into $\mathcal{D}^*$.
  - If $\pi =$*deletion*, call Query($\overline{\boldsymbol{TK}}, \mathcal{I}^*$) to find the data point $\bar{C}$. If yes, delete $\bar{C}$ from the R-tree, and delete encrypted record from $\mathcal{D}^*$.
  - If $\pi =$*modification*, first perform the *deletion* operation, and then perform *insertion* operation.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

4.3 Security Analysis

The security guarantees stated above ensure that nothing beyond the access pattern, search pattern and path pattern should be leaked. We first prove the intersection predicate encryption is indistinguishably secure. Then we adopt the simulation-based security model [16], [21], [22] to prove that TRQED is secure under known background model.

Lemma 1. The intersection predicate encryption schemes are semantically secure if the encrypted messages are indistinguishable.

Proof: We just have to prove that the probability for the probabilistic polynomial-time (PPT) adversary to break the encrypted messages of IPE schemes (*i.e.*, PIPE and RIPE) is negligible. Suppose the challenger runs Setup(1^κ) to generate an intersection encryption system SK, and a PPT adversary $\mathcal{A}$ submits two messages m_0 and m_1. The challenger randomly chooses $b \in \{0,1\}$, encrypts m_b with SK, and sends the ciphertext to $\mathcal{A}$. Then the adversary $\mathcal{A}$ takes a guess $b^{'}$ of b. Recall with the perturbation-based matrix encryption, a random number is used each time, thus transforming a plaintext into varied ciphertext with the same key. It is easy to conclude that $\mathcal{A}$ cannot guess b correctly with a probability higher than $1/2$. Thus the advantage in the security game is $\boldsymbol{Adv}_{IPE,\mathcal{A}}(\kappa) = |Pr[b = b^{'}] - \frac{1}{2}| < negl(\kappa)$. Moreover, considering the invertible matrix is random, it is difficult to be cracked, as there are an infinite number of key pairs. Thus we say that the IPE schemes are semantically secure. □

The indistinguishability of encrypted vectors and tokens is based on the indistinguishability of the pseudo-random number and perturbation-based matrix encryption, so that the index privacy is preserved. Based on Lemma 1, we analyze the security of TRQED.

Theorem 1. TRQED is secure against semi-honest adversaries under the known background model.

Before proving the Theorem 1, we introduce some notions used in [22] and adapt them for our proof.

- *History:* an interaction between the user and cloud server, determined by a dataset $\mathcal{D}$, a searchable index $\mathcal{I}$ and a set of queries $Q = (q_1, \cdots, q_\tau)$ submitted by users, denoted as the knowledge $H = (\mathcal{D}, \mathcal{I}, Q)$.
- *View:* the encrypted form of H under the secret key SK, denoted as $V(H)$, *i.e.*, the encrypted dataset $\mathcal{D}^*$, the secure index $\mathcal{I}^*$, and the search tokens $TK(Q)$. Note that the cloud server can only see the views.
- *Trace:* given a history H, the trace $Tr(H)$ is the information which can be learned by the cloud server. It contains the access pattern $\alpha(H)$, search pattern $\sigma(H)$, path pattern $\delta(H)$ and the returned identifiers $ID(Q)$. Let $ID(q)$ be the set of identifiers of the matched data records, and $\alpha(H) = \{ID(q_1), \cdots, ID(q_\tau)\}$. Let $P(q)$ be the set of identifiers of the matched nodes in $\mathcal{I}$, and $\delta(H) = \{P(q_1), \cdots, P(q_\tau)\}$. The search pattern $\sigma(H)$ is an $n \times \tau$ binary matrix where $\sigma(H)_{i,j}$ is 1 if I_i is returned by a query q_j, and 0 otherwise. Then we have $Tr(H) = \{ID(Q), \alpha(H), \sigma(H), \delta(H)\}$.

Under known background model, we assume the server obtains the $Tr(H)$, and a certain number of queries and its probability pairs (q_i, p_i). Informally, given two histories with the same trace, if the server with the distribution of queries cannot distinguish which view of them is generated by the simulator, he cannot learn additional knowledge beyond what we are willing to leak (*i.e.*, the trace), and thus our solution is secure.

Proof: Let $\mathcal{S}$ be a simulator that can simulate a view $V^{'}$ indistinguishable from the cloud server's view $V(H) = \{\mathcal{D}^*, \mathcal{I}^*, TK(Q)\}$. To achieve this, the $\mathcal{S}$ does the following:

- To generate $\mathcal{D}^{'}$, $\mathcal{S}$ selects a random $D_i^{'} \in \{0,1\}^{|D_i^*|}$, $D_i^* \in \mathcal{D}^*$, $1 \le i \le |\mathcal{D}^*|$, and outputs $\mathcal{D}^{'} = \{D_i^{'}, 1 \le i \le |\mathcal{D}^*|\}$.
- $\mathcal{S}$ randomly picks an invertible matrix $M_1^{'}, M_2^{'} \in \mathbb{R}^{4\times4}$. Set $SK^{'} = \{M_1^{'}, M_2^{'}\}$.
- To generate $I^{'}(\mathcal{D}^{'})$, $\mathcal{S}$ first generates a vector of $2w$ elements for each $v_i^{'} \in I^{'}(\mathcal{D}^{'})$, $1 \le i \le |\mathcal{I}^*|$ as the index, then does the following:
  1) For each element of $v_i^{'}$, $\mathcal{S}$ replaces it with a four-dimensional vector $(r_{i,1}, \cdots, r_{i,4})^T$, where $r_{i,k}$ is a random number for $1 \le i \le n, 1 \le k \le 4$.
  2) $\mathcal{S}$ encrypts each $v_i^{'}$ with the $M_2^{'}$, and obtains $I^{'}(\mathcal{D}^{'}) = \{Enc_{SK^{'}}(v_i^{'}), 1 \le i \le |\mathcal{I}^*|\}$.
- $\mathcal{S}$ constructs the query $Q^{'}$ and the search tokens $TK(Q^{'})$ as follows. For each $q_i^{'} \in Q^{'}$, $1 \le i \le \tau$,
  1) Generates a vector of $2w$ elements, denoted as $u_i^{'}$, and replace each element with a four-dimensional vector $(r_{i,1}^{'}, \cdots, r_{i,4}^{'})^T$, where $r_{i,k}^{'}$ is a random number for $1 \le i \le n, 1 \le k \le 4$. Output $Q^{'} = \{q_i^{'}, 1 \le i \le \tau\}$.
  2) Generate the search token for each $q_i^{'}$ by encrypting $u_i^{'}$ with $M_1^{'}$ from $SK^{'}$ for $1 \le i \le \tau$. Then $\mathcal{S}$ obtains $Enc_{SK^{'}}(Q^{'}) = \{Enc_{SK^{'}}(q_1^{'}), \cdots, Enc_{SK^{'}}(q_\tau^{'})\}$.
- $\mathcal{S}$ outputs the view $V^{'} = (\mathcal{D}^{'}, I^{'}(\mathcal{D}^{'}), TK(Q^{'}))$.

The correctness of such construction is easy to demonstrate by querying $TK(Q^{'})$ over $I^{'}(\mathcal{D}^{'})$. The index $I^{'}(\mathcal{D}^{'})$ and the token $TK(Q^{'})$ generate the same trace as the one that the cloud server has. We claim that no PPT adversary can distinguish the view $V^{'}$ from $V(H)$. Specifically, due to the semantic security of IPE and AES, no PPT adversary can distinguish the $\mathcal{D}^*$ from $\mathcal{D}^{'}$. Moreover, the PPT adversary with the query and probability pairs cannot distinguish which tokens are generated from the same query because of the indistinguishability of random perturbation-based comparison predicate, so that it cannot exploit the distributions of plain and encrypted queries to deduce information. Thus Theorem 1 holds. □

5 TRQED$^+$ WITH ENHANCED SECURITY

Our TRQED scheme achieves faster-than-linear search and supports dynamic operations while preserving the query and single-dimensional privacy, simultaneously. However, it should be pointed out that this scheme is weak in terms of preserving single-dimensional privacy. Although we have extended the number of dimensions from w to $2w$, the cloud server can still deduce which dimensions are for dummy data at a high probability by statistical analysis of the intersecting results of wdPIPE.Query and wdRIPE.Query. Besides, we ignore access pattern, search pattern and path pattern in order to ensure high efficiency. Unfortunately, through analyzing these informations, the cloud server may be able to deduce some useful information about the data and queries [11], [23]. So, in this section, we target at a safer design, called TRQED$^+$.

It is helpful to make use of Private Information Retrieval (PIR) [24] or Oblivious RAM (ORAM) Protocol [25] to address the issues of access pattern and search pattern. However, for practical

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

applications, both PIR and ORAM suffer from the effectiveness issue. In order to ensure effectiveness and practicality, we intentionally ignore access pattern and search pattern. Naturally, we focus on preserving path pattern and stronger single-dimensional privacy through effective design.

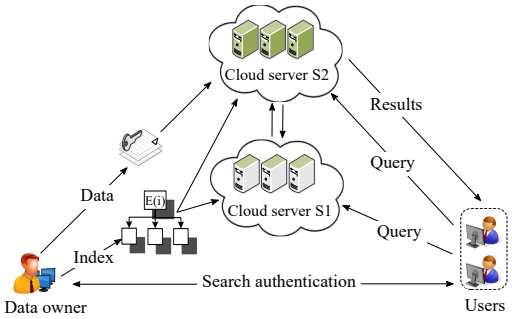


Fig. 3: Architecture of TRQED$^+$ with two cloud servers

5.1 System Model With Two Cloud Servers

In order to achieve our design goals, we add a cloud server to the original system model illustrated in Fig. 1. Therefore, the modified system model consists of a *data owner*, two *cloud servers* and multiple authorized *users*, as depicted in Fig. 3. The two servers are denoted as S_1 and S_2, respectively. We assume that S_1 and S_2 are *non-colluding* and *semi-honest*. Such an assumption is reasonable and has been widely used in related research fields [26], [27], [28]. The companies that provide cloud services, such as Google and Amazon, are legitimate and well-established. A collusion will damage their reputation.

Similarly, the data owner first builds index (*i.e.*, $\mathcal{I}$) and encrypts the index (*i.e.*, $\mathcal{I}^*$) and data (*i.e.*, $\mathcal{D}^*$). The data owner divides the encrypted index $\mathcal{I}^*$ into two parts, encrypted index nodes $\{E_1, \cdots, E_m\}$ and encrypted data points $\{C_1, \cdots, C_n\}$, and sends them to S_1 and S_2, respectively. Simultaneously, the data owner outsources the encrypted data set $\mathcal{D}^*$ to S_2. In the query phase, an authorized user sends the query range to the data owner and obtains the search token (*i.e.*, $\boldsymbol{TK} = \{TK_{point}, TK_{node}\}$). Similarly, the user submits TK_{point} and TK_{node} to S_1 and S_2, respectively. On receiving the search token, S_1 and S_2 cooperatively perform the query task.

5.2 Modified PIPE and RIPE

As mentioned above, extending dimension cannot fully preserve single-dimensional privacy. We will exploit a flexible matrix column random-permutation in our new design. Thus, we do not have to extend dimensions. Specifically, for wdPIPE and wdRIPE, we modify ciphertext C and search token TK as follows:

$$C = r_v(\widetilde{M}\boldsymbol{v}_1, \cdots, \widetilde{M}\boldsymbol{v}_w)$$
$$TK = r_p(M^T\boldsymbol{p}_1, \cdots, M^T\boldsymbol{p}_w)$$

We call the modified algorithms wdPIPE$^+$ and wdRIPE$^+$.

5.3 Oblivious R-tree Traversal

Inspired by the traversal path obfuscation in [29], we design an oblivious R-tree traversal (ORT) on the basis of the original scheme to preserving path pattern and single-dimension privacy. In our new design, cloud servers S_1 and S_2 cooperatively execute a query without sacrificing path pattern and single-dimension privacy.

5.3.1 Secure Scalar Product

Before describing ORT, we introduce a basic secure multi-party computation (SMC) protocol, called secure scalar product (SSP) protocol. In [30], Atallah *et al.* first put forward the scalar product protocol to solve the scalar product problem, defined as follows.

Definition 1. (**Scalar Product Problem**) Alice has a vector $\boldsymbol{x} = (x_1, \cdots, x_n)$ and Bob has a vector $\boldsymbol{y} = (y_1, \cdots, y_n)$. Alice (but not Bob) is to get the result of $u = \boldsymbol{x} \cdot \boldsymbol{y} + v$ where v is a random scalar known to Bob only.

For the usual scalar product computation, let $v = 0$ in Definition 1. SSP protocol is diffusely studied in data mining, information retrieval and other fields [31], [32]. Diverse approaches have been proposed in accordance with different settings. In this paper, we adopt the approach proposed in [32] to achieve the secure intersection operation.

5.3.2 Secure Intersection Operation

First, we present a secure intersection (SI) operation. We consider a ciphertext C and a range query TK which are defined in both wdPIPE and wdRIPE algorithm. C and TK are both $4 \times w$ matrices, so we define $C = (\boldsymbol{c}_1, \cdots, \boldsymbol{c}_w)$ and $TK = (\boldsymbol{tk}_1, \cdots, \boldsymbol{tk}_w)$, where $\boldsymbol{c}_i$ and $\boldsymbol{tk}_i$ are the column vectors. We assume that Alice has TK and Bob has C. They can execute secure intersection operation through obfuscating the two matrices as follows:

- **Phase 1**: At the beginning of this phase, Bob has ciphertext C, and Alice has search token TK. Then,
  - Bob obfuscates C by w random diagonal matrices $\{R_{1,1}, \cdots, R_{1,w}\}$ and a permutation function Ω_w.
  $$C' = \Omega_w((R_{1,1}\boldsymbol{c}_1, \cdots, R_{1,w}\boldsymbol{c}_w)) = (\boldsymbol{c}'_1, \cdots, \boldsymbol{c}'_w)$$
  - Alice obfuscates TK by w random diagonal matrices $\{R_{2,1}, \cdots, R_{2,w}\}$.
  $$TK' = (R_{2,1}\boldsymbol{tk}_1, \cdots, R_{2,w}\boldsymbol{tk}_w) = (\boldsymbol{tk}'_1, \cdots, \boldsymbol{tk}'_w)$$
  - Bob and Alice exchange C' and TK'.
- **Phase 2**: At the beginning of this phase, Bob has TK', and Alice has C'. Similarity,
  - Bob obfuscates TK' as follows:
  $$TK'' = \Omega_w((R_{1,1}^{-1}\boldsymbol{tk}'_1, \cdots, R_{1,w}^{-1}\boldsymbol{tk}'_w)) = (\boldsymbol{tk}''_1, \cdots, \boldsymbol{tk}''_w)$$
  - Alice obfuscates C' as follows:
  $$C'' = (R_{2,1}^{-1}\boldsymbol{c}'_1, \cdots, R_{2,w}^{-1}\boldsymbol{c}'_w) = (\boldsymbol{c}''_1, \cdots, \boldsymbol{c}''_w)$$

Now, Bob and Alice obtain obfuscated token TK'' and ciphertext C'', respectively. For $1 \le i \le w$, Bob and Alice can cooperatively calculate $\langle \boldsymbol{tk}''_i, \boldsymbol{c}''_i \rangle$ through above-mentioned SSP protocol. It's simple to prove that,

$$\langle \boldsymbol{tk}''_i, \boldsymbol{c}''_i \rangle \le 0, \forall i \in [1, w] \Leftrightarrow \langle \boldsymbol{tk}_i, \boldsymbol{c}_i \rangle \le 0, \forall i \in [1, w].$$

5.3.3 Secure Query Based R-tree

In our new scheme, data owner divides the encrypted index $\mathcal{I}^*$ into two parts, *i.e.*, $\{E_1, \cdots, E_m\}$ and $\{C_1, \cdots, C_n\}$, which are sent to S_1 and S_2 respectively. Moreover, authorized users submit TK_{point} to S_1 and TK_{node} to S_2. Naturally, the query process will also be divided into two parts, *i.e.*, *node query* and *point query*.

Note that tree traversal is executed according to the level iteratively. For level $h \in [1, H]$, S_1 has node set O and S_2

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

has range query token TK_{node}. In order to achieve a secure query between S_1 and S_2, we present a secure node query (SNQ) operation.

SNQ operation is described as follows:

1) S_1 first generates a random perturbation function Ω and adopts it to obfuscate the order of nodes in O. We denote perturbed O by $\mathbb{O}$. The purpose of adopting perturbation function is to obfuscate S_2's view.
2) For perturbed node set $\mathbb{O}$, S_1 (as Bob in SI) uses $\{R_{1,1}, \cdots, R_{1,w}\}$ and Ω_w to obfuscate each node of $\mathbb{O}$ according to **Phase 1** of SI, and the obfuscated result is denoted as $\mathbb{O}'$. Similarly, S_2 (as $Alice$ in SI) obfuscates TK_{node} and then gets TK'_{node}. Afterwards, S_1 and S_2 exchange $\mathbb{O}'$ and TK'_{node}.
3) Once receiving TK'_{node} from S_2, S_1 can immediately get TK''_{node} according to **Phase 2** of SI. For each node of $\mathbb{O}'$, S_2 first checks its flag. If the check result is true, S_2 continues to process it according to **Phase 2** and then gets the intersecting result by employing SSP protocol; otherwise, current node is impossible to intersect with the query range. After processing all node of $\mathbb{O}'$, S_2 gets the intersecting result set.
4) S_2 sends the intersecting result set to S_1. S_1 recovers the order of the results by exploiting the inverse perturbation function Ω^{-1} and then assigns the results to corresponding children.

The above SNQ operation is repeated until the leaf level. After processing leaf nodes, S_1 sends the intersecting results to S_2. According to the results, S_2 knows which data point sets intersect with the query range. Next, S_2 needs to query these point sets. Similarly, we present a secure point query (SPQ) operation by employing the above-mentioned SI operation.

SPQ operation is described as follows:

1) For point set $\mathbb{P}$, S_2 (as Bob in SI) uses $\{R_{2,1}, \cdots, R_{2,w}\}$ and Ω_w to obfuscate each point of $\mathbb{P}$ according to **Phase 1** of SI, and the obfuscated result is denoted as $\mathbb{P}'$. Similarly, S_1 (as $Alice$ in SI) obfuscates TK_{point} and gets TK'_{point}. Then S_2 and S_1 exchange $\mathbb{P}'$ and TK'_{point}.
2) Once receiving TK'_{point} from S_1, S_2 can immediately get TK''_{point} according to **Phase 2** of SI. For each point of $\mathbb{P}'$, S_1 processes it according to **Phase 2**. We denote obfuscated point set by $\mathbb{P}''$.
3) S_1 and S_2 use TK''_{point} to query every point of $\mathbb{P}''$ by employing SSP protocol, and S_2 gets the query result.

5.3.4 Flag Encryption

In our new scheme, S_2 needs to send the intersecting results of every level to S_1, so that the latter can assign the results to the corresponding child nodes. However, S_1 should not know the exact information of the results (*i.e.*, $flag$ equals 0 or 1). In other words, S_2 has to encrypt the intersecting results. We use *Paillier cryptosystem* [33] to ensure that S_1 cannot learn the intersecting results. Specifically, at the beginning of the query processing, S_2 generates a key pair $\{pk, sk\}$ and sends the public key pk to S_1. Before sending the intersecting results to S_1, S_2 uses pk to encrypt them. On the other hand, after assigning the encrypted results to child notes, S_1 multiples the flag of every child node by $\hat{0}$ (*encrypted form*) to obfuscate S_2's view. S_2 can check the encrypted flag by decrypting it with the secret key sk.

5.3.5 Oblivious R-tree Traversal Algorithm

Now, we can summarize the oblivious R-tree traversal (ORT) algorithm. The details of ORT are presented in Alg. 3.

Algorithm 3 ORT

- Setup(1^κ). On input a security parameter κ, S_2 generates a key pair $\{pk, sk\}$ for flag encryption. Moreover, S_1 and S_2 randomly generates $\{R_{1,1}, \cdots, R_{1,w}\}$ and $\{R_{2,1}, \cdots, R_{2,w}\}$, respectively, where $R_{1,i}$ and $R_{2,j}$ are diagonal matrices without zero diagonal elements.
- PaiE_{pk}, PaiD_{sk}. Encryption function and decryption function in Paillier cryptosystem.
- SNQ(O, TK_{node}). For $h \in [1, H]$, S_1 has node set O of level h and S_2 has query token TK_{node}. S_1 and S_2 execute above-mentioned secure node query operation. Note that the flags of nodes are initialized to $\mathsf{PaiE}_{pk}(1)$ when $h = 1$.
- SPQ($\mathbb{P}$, TK_{point}). S_2 has query token TK_{node} and S_1 has candidate point set $\mathbb{P}$. S_1 and S_2 execute above-mentioned secure point query operation.

5.4 TRQED$^+$ Construction

Adopting aforementioned oblivious R-tree traversal, we propose a tree-based private range query scheme over dynamic encrypted cloud data with enhanced security guarantees. As same as TRQED, TRQED$^+$ consists of four polynomial algorithms (Setup, EncIndex, GenToken, Query). We present them as follows.

- Setup(1^κ). The data owner generates a secret key $\boldsymbol{SK} = \{SK_{point}, SK_{node}, SK_{data}\}$.
- EncIndex($\boldsymbol{SK}, \mathcal{D}$). As similar as TRQED, the data owner builds an R-tree $\mathcal{I} = \{D_1, \cdots, D_n, N_1, \cdots, N_m\}$ and then encrypts the index and data as follows:

$$C_i \leftarrow w\text{dPIPE}^+.\mathsf{Encrypt}(SK_{point}, D_i),$$
$$E_j \leftarrow w\text{dRIPE}^+.\mathsf{Encrypt}(SK_{node}, N_j),$$
$$D_i^* \leftarrow AES.encrypt(SK_{data}, D_i).$$

  Afterwards, the data owner outsources $\{E_1, \cdots, E_m\}$ and $\{C_1, \cdots, C_n\}$ to the cloud servers S_1 and S_2, respectively. Simultaneously, the data owner outsources the encrypted dataset $\mathcal{D}^*$ to S_2.
- GenToken($\boldsymbol{SK}$, Q). The data owner generates a search token $\boldsymbol{TK} = \{TK_{point}, TK_{node}\}$, where

$$TK_{point} \leftarrow w\text{dPIPE}^+.\mathsf{GenToken}(SK_{point}, Q),$$
$$TK_{node} \leftarrow w\text{dRIPE}^+.\mathsf{GenToken}(SK_{node}, Q).$$

  Then $\boldsymbol{TK}$ is distributed to authorized users. Then users submit TK_{point} and TK_{node} to S_1 and S_2, respectively.
- Query($\boldsymbol{TK}$, $\mathcal{I}^*$). On input the $\boldsymbol{TK}$ and $\mathcal{I}^*$, the cloud servers S_1 and S_2 cooperatively perform a secure range search as follows:
  - By executing ORT.Setup(1^κ), S_2 generates a key pair $\{pk, sk\}$ and $\{R_{2,1}, \cdots, R_{2,w}\}$, and S_1 generates $\{R_{1,1}, \cdots, R_{1,w}\}$. Then, S_2 sends pk to S_1.
  - S_1 and S_2 cooperatively execute ORT.SNQ(O, TK_{node}) iteratively until the leaf level. On receiving the intersecting results of leaf nodes, S_1 uses the inverse perturbation function to recover the order of the results. Then S_1 multiplies them by ORT.PaiE(0) and then sends these results to S_2. By using ORT.PaiD($\cdot$) to decrypt these results, S_2 learns which point sets are candidates.
  - For candidate point set $\mathbb{P}$, S_1 and S_2 cooperatively execute ORT.SPQ($\mathbb{P}$, TK_{point}). According to the design of algorithm ORT.SPQ($\cdot$), S_2 will get the point query results.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

If a data point C_i belongs to the query range, S_2 puts the identifier I_i of this data point into the list ID_Q.

Finally, the cloud server S_2 returns a set ID_Q of identifiers of matched data records, and the user obtains the results.

5.5 An Example of Query

For a better understanding of TRQED$^+$, we describe the query process in detail through the example of Fig. 2. The same as TRQED, the data owner builds and encrypts R-tree in TRQED$^+$. The user submits range query Q, and receives corresponding search token $\boldsymbol{TK} = \{TK_{point}, TK_{node}\}$ from the data owner. However, the query process in TRQED$^+$ is quite different from that in TRQED. The data owner outsources $\{E_1, \cdots, E_6\}$ and $\{C_1, \cdots, C_8\}$ to S_1 and S_2, respectively. The user with query Q submits TK_{point} and TK_{node} to S_1 and S_2, respectively. Then, we describe the traversal process layer by layer.

- $\boldsymbol{h = 1}$ (*root, non-leaf layer*): S_1 initializes the flags of $\{E_1, E_2\}$ to $\{\hat{1}, \hat{1}\}$ (*encrypted form*). According to SNQ, S_1 can get obfuscated search token TK''_{node}, and S_2 can get obfuscated candidate nodes $\{E''_2, E''_1\}$. By employing SSP protocol, S_2 obtains the intersecting result set $\{1, 0\}$. After that, S_2 encrypts and sends the flag set $\{\hat{1}, \hat{0}\}$ to S_1, who then recovers the order of the flags and assigns them to corresponding child nodes.
- $\boldsymbol{h = 2}$ (*leaf layer*): S_1 multiples the flag of each node by $\hat{0}$ to obfuscate S_2's view, which is equivalent to the addition operation on the plaintext domain. Therefore, the flags of nodes N_3, N_4, N_5, N_6 are $\{\hat{0}, \hat{0}, \hat{1}, \hat{1}\}$. According to SNQ, S_1 obfuscates these nodes, and sends $\{N'_6(\hat{1}), N'_4(\hat{0}), N'_3(\hat{0}), N'_5(\hat{1})\}$. After that, S_2 can get candidate nodes $\{N''_6, N''_5\}$ by checking flags. By employing SSP protocol, S_2 obtains the intersecting result set $\{1, 1\}$. Then, S_2 encrypts and sends the flag set $\{\hat{1}, \hat{0}, \hat{0}, \hat{1}\}$ to S_1. Afterwards, S_1 recovers the order of the flags, and gets $\{\hat{0}, \hat{0}, \hat{1}, \hat{1}\}$.
- **Data points layer**: Similarly, for each encrypted flag, S_1 multiples it by $\hat{0}$ to obfuscate S_2's view, and then sends the flag set to S_2. By decrypting these flag, S_2 gets the candidate point set $\mathbb{P} = \{C_5, C_6, C_7, C_8\}$. S_1 and S_2 cooperatively execute ORT.SPQ($\mathbb{P}$, TK_{point}), and S_2 gets the results $\{0, 1, 1, 0\}$.

Finally, S_2 sends $\{D^*_6, D^*_7\}$ to the user, and the latter uses SK_{data} to decrypt $\{D^*_6, D^*_7\}$ to obtain data records.

5.6 Analysis

5.6.1 Correctness Analysis

As stated in SI (see Sec. 5.3.2), for a ciphertext C and a search token TK, SI can guarantee that,

$$\langle \boldsymbol{tk}''_i, \boldsymbol{c}''_i \rangle \leq 0, \forall i \in [1, w] \Leftrightarrow \langle \boldsymbol{tk}_i, \boldsymbol{c}_i \rangle \leq 0, \forall i \in [1, w].$$

Therefore, Our modifications do not affect the correctness of the original scheme TRQED. In other words, the correctness of TRQED$^+$ is based on the correctness of TRQED. And the correctness of TRQED has been proved in Sec. 4.1. Hence, the query results are correct.

5.6.2 Security Analysis

Compared with basic TRQED, TRQED$^+$ performs the *oblivious tree traversal* operation during the query processing. Inherited from TRQED scheme, it's easy to prove that the TRQED$^+$ scheme can protect data privacy, index privacy and query privacy under the known background model. Next, we show that ORT operation can protect path pattern and single-dimensional privacy.

Path pattern privacy. During searching each level of R-tree, we adopt SSP protocol in secure intersection operation so that S_1 is unable to know the intersecting results between nodes and query token. Although knowing intersecting results, S_2 cannot distinguish the nodes that are obfuscated by S_1. Specifically, before sending node set O of each level to S_2, S_1 uses perturbation function Ω to obfuscate the order of nodes in O. As a result, S_2 is unable to learn the true traversal path. Hence, neither S_1 nor S_2 can learn which nodes match the query token. For every query, $\{R_{1,1}, \cdots, R_{1,w}\}$ and $\{R_{2,1}, \cdots, R_{2,w}\}$ are independently generated by S_1 and S_2, respectively. And perturbation function Ω and Ω_w are always regenerated when they are used. Hence, an oblivious R-tree traversal algorithm can protect path pattern under known background model.

Single-dimensional privacy. For *node query* and *point query* during the query process, we adopt SNQ operation and SPQ operation, respectively. As we have analysed above, only S_2 is allowed to know the intersecting results, and S_2 cannot distinguish the nodes that are obfuscated. Moreover, S_1 adopts perturbation function Ω_w to obfuscate the order of all columns of each matrix. In other words, S_1 and S_2 cannot obtain the exact search results for any single-dimensional query. Similarly, In SPQ, for each data point, two cloud servers are not able to exactly know which dimensions match the range of the requested query. Therefore, ORT can fully prevent single-dimensional privacy leakage.

5.7 TRQED$^+$-based Data Update

Though TRQED$^+$ executes range query by two servers, it still supports basic dynamic operations in range query processing. As the same as TRQED, we assume that each time a record is updated, no splitting or merging is incurred. The main idea of TRQED$^+$-based data updating is similar to that of TRQED. Data modification is considered as a deletion followed by an insertion. The detailed description is as follows.

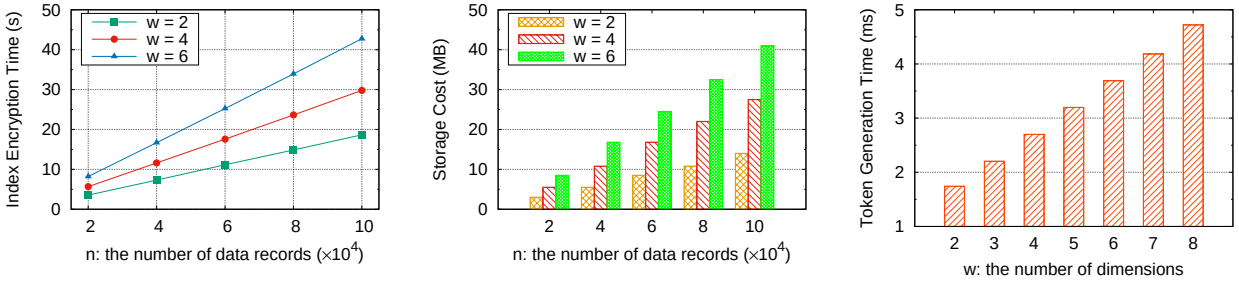
- Insertion. For a new data record $\bar{D}$ that needs to be inserted, we express it as a range $\overline{\text{MBR}}$. First, the data owner computes an update token $\boldsymbol{TK}_\pi = \overline{TK}_{node}$. Then the data owner sends $\boldsymbol{TK}_\pi$, $\bar{C}$ and the encrypted data record to the cloud server S_2. Once receiving them, S_2 searches the R-tree with the cooperation of cloud server S_1 until it finds the leaf node $\bar{E}$ that contains $\bar{D}$. S_2 puts the encrypted record into $\mathcal{D}^*$, and inserts $\bar{C}$ and identifier of encrypted record into the correct point set.
- Deletion. Given the record $\bar{D}$, the owner submits the deletion token $\boldsymbol{TK}_\pi = \{\overline{TK}_{point}, \overline{TK}_{node}\}$. The cloud server S_2 locates the record $\bar{D}$ with the cooperation of S_1. Then S_2 deletes $\bar{C}$, together with its corresponding encrypted record and identifier.

6 Evaluation

We evaluate the performance of our schemes on a real-world (REAL) dataset from the U.S. census bureau dataset [34]. The REAL dataset consists of $299,285$ records with more than 20 attributes. We chose a number of records (from $20,000$ to $100,000$) to construct our datasets with needed dimensionality, and built R-tree

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

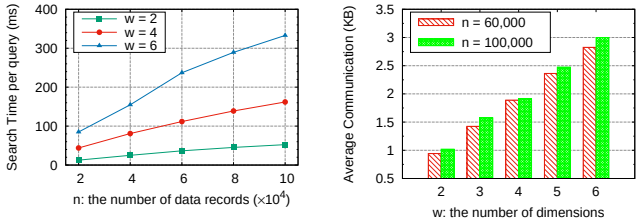
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering



(a) (b) (c)

Fig. 4: Setup and token generation cost: (a) R-tree index encryption time versus n with dimensions $w = 2, 4, 6$; (b) Total storage cost versus n with $w = 2, 4, 6$; (c) Token generation time versus w with $n = 100,000$.

indexes on each REAL dataset. (Note here that we do not limit the size of the dataset. Under the premise of ensuring the validity of the experiment, a lager dataset will get the similar evaluation result.) The data owner, user and cloud servers are all set on desktop computers with 3.20GHz CPU and 4GB RAM. We adopt GMP library and AES CTR mode for data encryption, and set the security parameter κ at 128 and bit length l of each attribute at 32. For performance evaluation, we focus on overhead and efficiency as the schemes are proved to have a high search accuracy, which we omit for space reasons. 100 queries are executed and their average costs are reported.



(a) (b)

Fig. 5: Search efficiency: (a) Search time versus n with $w = 2, 4, 6$; (b) Communication cost versus w with $n = 60,000, 100,000$.

6.1 Overhead and Efficiency of TRQED

We first discuss the performance of TRQED, which mainly includes overhead and efficiency. Compared to TRQED$^+$, TRQED fails to preserve path pattern and stronger single-dimensional privacy, but has higher search efficiency.

6.1.1 Setup

To set up the system, the data owner first encrypts all nodes of the R-tree index, of which each node requires about $\mathcal{O}(w)$ matrix-vector multiplications and bits. Here we are interested in the encryption overhead ignoring tree construction cost. As shown in Fig. 4(a) and Fig. 4(b), the costs of index encryption is increasing with the number of dimensions w and data records n, as the TRQED should encrypt each non-leaf node of R-tree with RIPE and each leaf node with PIPE. That is, TRQED trades off the setup time and storage cost for improvement of the search efficiency.

6.1.2 Token generation

We evaluate the experimental results of generating search tokens in Fig. 4(c). The time of token generation per query is linearly increasing with the number of dimensions w, as TRQED needs to spend additional time to generate tokens in the added dimensions.

6.1.3 Search

We can see from Fig. 5(a) the impact of w and n on the search time per query. The search time shows a logarithmic increase trend as n increases, as the cloud server searches the R-tree level-by-level honestly to check whether the node satisfies $\mathsf{Query}(TK, \mathcal{I}^*) = true$. If fixing n, the cost shows a linear trend with the dimensions w, as the amount of inner product computations at each node increases with w. This verifies search time is linear with the number of dimensions and the height of R-tree. In Fig. 5(b), it can be seen that the average communication cost is linear with w, n has little effect, as the communication cost is composed of the generated search tokens and matched records.

The efficiency of data update is similar to that of the search phase, as the update operation is essentially fulfilled through search. The detailed results are omitted due to space limitation. Roughly speaking, the *insertion* and *deletion* operations require about the same computation, while the *modification* needs about the double computation overhead.

6.2 TRQED v.s. Prior Art

We compare our TRQED with the existing protocols, *i.e.*, BonehW [7], LSED$^+$ [9], WangR [10], and Maple [11] in terms of privacy and data dynamics. As for efficiency, BonehW [7] fails to achieve faster-than-linear search and LSED$^+$ [9] achieves efficiency at the price of revealing ordering information and compromising privacy which makes it be impractical. Therefore, we focus our efficiency comparison on WangR [10] and Maple [11], which are practical and represent the state-of-the-art multi-dimensional range query schemes.

6.2.1 Privacy and Data Dynamics

We compare the privacy guarantees and data dynamics support of the four related schemes and our TRQED, as shown in Table 1. As for the query and single-dimensional privacy, LSED$^+$ and WangR preserve the query privacy and do not provide single-dimensional privacy. BonehW and Maple preserve the single-dimensional privacy and fail in query privacy. In contrast, our TRQED preserves both the query privacy and single-dimensional privacy. However, TRQED is weak in terms of preserving single-dimensional privacy. Although we have extended the dimension w to $2w$ dimensions, the cloud server can still deduce which dimensions are dummy data at high probability by statistical analysis of

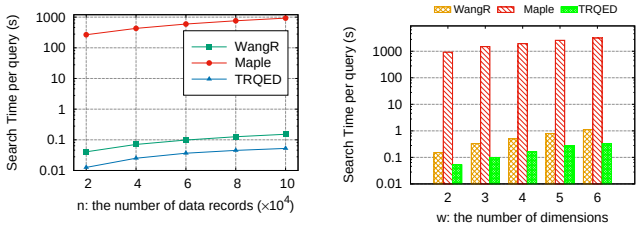
1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

intersecting results of wdPIPE.Query and wdRIPE.Query. As to data dynamics, only LSED$^+$ and our TRQED can provide data update while the other three schemes do not support.

6.2.2 Efficiency

The experimental results of search efficiency are depicted in Fig. 6(a) with n varying from $20,000$ to $100,000$. They all show an increasing trend as n increases. Fig. 6(b) illustrates the search time of the three schemes are all linearly increasing with w. By contrast, the time cost of TRQED is slightly smaller than that of WangR because of the smaller coefficient of the time complexity. Compared with Maple, TRQED and WangR are much faster than Maple, as Maple relies on public-key cryptography and the impact of domain limit T cannot be ignored. Thus, the proposed TRQED is more efficient.



(a) (b)

Fig. 6: Performance comparison: (a) Average search time of the schemes versus n with $w = 2$; (b) Average search time versus w with $n = 100,000$.

From the above comparison, we can conclude TRQED has a better performance than prior art. It has a comparatively desirable overhead.

6.3 Performance of TRQED$^+$

In terms of system architecture, TRQED$^+$ is very different from TRQED. To improve the security, TRQED$^+$ uses two non-collusion cloud servers. For comparison, we evaluate three algorithms, which are BASIC, TRQED and TRQED$^+$.

6.3.1 Enhanced security

Compared with TRQED, TRQED$^+$ provides stronger privacy guarantees, as shown in Table 1. TRQED can preserve single-dimensional privacy to a certain extent. However, the cloud server can use statistical analysis to guess which dimensions are dummy. By executing SNQ and SPQ operations, TRQED$^+$ can perform a search task while preserving single-dimensional privacy. Furthermore, TRQED fails in path pattern privacy, as with many existing arts. The oblivious R-tree traversal operation designed in TRQED$^+$ can assure that path pattern will not be revealed to cloud servers.

6.3.2 Search Efficiency

For evaluating the performance of TRQED$^+$, we add a BASIC solution to the comparison. Briefly, BASIC performs the SPQ operation to check all data points encrypted by wdPIPE$^+$ without any tree structure. As shown in Fig. 7, the search time of three algorithms are all increasing with data records n. Conspicuously, TRQED shows the best search performance. Compared with others, TRQED is about 200 times faster than TRQED$^+$ and 2000

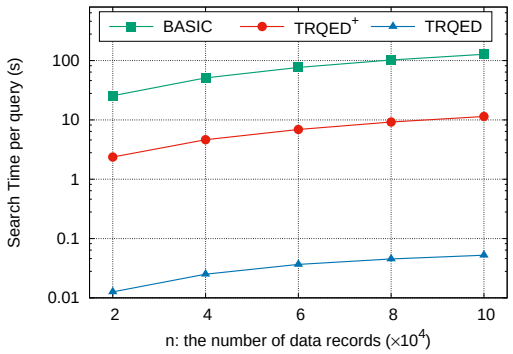


Fig. 7: Performance comparison: Average search time of the schemes versus n with $w = 2$

times faster than BASIC, as the latter two solutions use dual-server architecture with large communication time cost. Nevertheless, TRQED$^+$ and BASIC with more complex operations can better preserve single-dimensional privacy. Moreover, compared with BASIC, TRQED$^+$ can not only greatly improve the search efficiency by using R-tree structure, but also avoid path pattern being revealed during the traversal process.

7 Related Work

Our work is related to prior art in two aspects: searchable encryption and range query.

Searchable Encryption. The issue of secure query processing on encrypted cloud data has been studied in recent years. An ideal approach is using ORAM [25]. However, the efficiency of ORAM is a huge concern for real applications. Traditional Searchable Encryption (SE) schemes [2], [35] have been put forward to support simple types of queries, but directly deploying them would not be adequate for practical use. Wong *et al.* proposed a SCONEDB model [20], which achieved secure kNN query over encrypted vector databases. Li *et al.* [4] proposed an authorized private keyword search framework. Subsequently, researchers studied the privacy-preserving multi-keyword ranked search [5], fuzzy search [6], and verifiable search [36]. However, the researches only support keyword searches or single-dimensional range queries, which cannot be applied to the complex multi-dimensional range queries.

Range Query. Papadopoulos *et al.* [37] studied range queries in authenticated data structures while the privacy requirements are not considered. In [38], Kawamoto *et al.* proposed a private range query approach at the price of much more memorizing space. By using HVE, BonehW [7] designed a public-key query system to support multi-dimensional range queries over encrypted data. However, the scheme incurs high computation due to heavy reliance on public-key cryptography, and cannot protect query privacy. To improve query efficiency, [9] and [10] aimed to achieve faster-than-linear search time. However, as mentioned before, these two schemes would cause single-dimensional privacy leakage. To protect single-dimensional privacy, [11] proposed a scheme Maple based on HVE and R-trees. As a public-key scheme, it suffers from heavy computation and fails to provide query privacy. Afterward, the authors proposed a symmetric-key MDRSE [39], which achieves faster-than-linear search, query privacy and single-dimensional privacy simultaneously. However, it should know all the range domain sizes of the dimensions in

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

TABLE 1: Comparison among Different Solutions

	Faster-than-linear search	Query privacy	Single-dimensional privacy	Path pattern	Data dynamics support
BonehW [7]	×	×	√	−	×
LSED$^+$ [9]	√	√	×	×	√
WangR [10]	√	√	×	×	×
Maple [11]	√	×	√	×	×
TRQED	√	√	√*	×	√
TRQED$^+$	√	√	√	√	√

* Privacy guarantee is weak and vulnerable.

advance, and thus cannot support data update effectively. Furthermore, the client of the schemes who outsources database should be the one that searches, which is different from our private range query model.

8 Conclusion

In this paper, we have studied the problem of multi-dimensional private range queries over dynamic encrypted cloud data. Our proposed TRQED is the first private range query scheme that achieves faster-than-linear search and supports data dynamics while preserving the query privacy and single-dimensional privacy simultaneously. Furthermore, we also extended our design based on TRQED scheme. To achieving a higher privacy protection level, we have designed oblivious R-tree traversal and proposed TRQED$^+$, the security-enhanced version of TRQED. Perturbation-based ORT operation can prevent servers from learning path pattern during the search of R-tree. On the other hand, exploiting SSP protocol in SNQ and SPQ can achieve stronger single-dimensional privacy preservation.

However, we should point out that there are two main unsolved challenges, *i.e.*, splitting or merging of nodes, and access pattern on an encrypted database. Our schemes support data dynamics on condition that the updating data do not split or merge a non-leaf node in R-tree. Moreover, the cloud server could learn the access pattern during executing range queries. We hope these challenges will be overcome in future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61572456) and the Anhui Initiative in Quantum Information Technologies (No. AHY150300).

References

[1] H. Hacigumus, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proc. 18th Int. Conf. Data Eng.*, Feb. 2002, pp. 29–38.

[2] Dawn Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security and Privacy*, May 2000, pp. 44–55.

[3] M. Barhamgi, D. Benslimane, S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, M. Mrissa, and H. Taktak, "Secure and privacy-preserving execution model for data services," in *Advanced Information Systems Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 35–50.

[4] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *Proc. 31st Int. Conf. Distributed Computing Systems*, Jun. 2011, pp. 383–392.

[5] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837.

[6] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 2112–2120.

[7] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Fourth Conf. Theory Cryptograph*, 2007, pp. 535–554.

[8] T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products," in *Advances in Cryptology*. Berlin, Heidelberg: Springer, 2009, pp. 214–231.

[9] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud," in *Proc. NDSS*, 2012, pp. 1–17.

[10] P. Wang and C. V. Ravishankar, "Secure and efficient range queries on outsourced databases using rp-trees," in *Proc. IEEE 29th Int. Conf. Data Eng.*, Apr. 2013, pp. 314–325.

[11] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Maple: Scalable multi-dimensional range search over encrypted cloud data with tree-based index," in *Proc. 9th ACM Symp. Inf. Computer Commun. Secur.*, 2014, pp. 111–122.

[12] W. Yang, Y. Xu, Y. Nie, Y. Shen, and L. Huang, "Trqed: Secure and fast tree-based private range queries over encrypted cloud," in *Proc. DASFAA*, 2018, pp. 130–146.

[13] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.

[14] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast and scalable range query processing with strong privacy protection for cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2305–2318, Aug. 2016.

[15] M. Barhamgi, D. Benslimane, Y. Amghar, N. Cuppens-Boulahia, and F. Cuppens, "Privcomp: a privacy-aware data service composition system," in *Proceedings of the 16th International Conference on Extending Database Technology*. Citeseer, 2013, pp. 757–760.

[16] G. Oded, *Foundations of Cryptography: Volume 2, Basic Applications*. New York, NY, USA: Cambridge University Press, 2009.

[17] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, Jun. 2010, pp. 253–262.

[18] M. Barhamgi, A. K. Bandara, Y. Yu, K. Belhajjame, and B. Nuseibeh, "Protecting privacy in the cloud: Current practices, future directions," *IEEE Computer*, vol. 49, no. 2, pp. 68–72, Feb 2016.

[19] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. 6th Theory Cryptography Conf.*, 2009, pp. 457–473.

[20] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. 35th ACM SIGMOD*, 2009, pp. 139–152.

[21] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. 16th Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2010, pp. 577–594.

[22] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 79–88.

[23] S. Li and M. Wong, "Privacy-preserving queries over outsourced data with access pattern protection," in *Proc IEEE ICDM Workshops*, Dec. 2014, pp. 581–588.

[24] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, nov. 1998.

[25] E. Stefanov, M. V. Dijk, E. Shi, T.-H. H. Chan, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: An extremely simple oblivious ram protocol," *J. ACM*, vol. 65, no. 4, pp. 18–26, 2018.

[26] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 664–675.

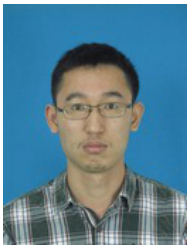
1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2983030, IEEE Transactions on Knowledge and Data Engineering

[27] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 633–644.
[28] K. Xue, S. Li, J. Hong, Y. Xue, N. Yu, and P. Hong, "Two-cloud secure database for numeric-related sql range queries with privacy preserving," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 7, pp. 1596–1608, 2017.
[29] N. Cui, X. Yang, L. Wang, B. Wang, and J. Li, "Secure range query over encrypted data in outsourced environments," in *Proc. DSFAA*, J. Pei, Y. Manolopoulos, S. Sadiq, and J. Li, Eds., 2018, pp. 112–129.
[30] M. J. Atallah and W. Du, "Secure multi-party computational geometry," in *Proc. WADS2001: 7th Int. Workshop on Algorithms and Data Structures*, 2001, pp. 165–179.
[31] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Pro. Eighth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2002, pp. 639–644.
[32] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explor. Newsl.*, vol. 4, no. 2, pp. 28–34, dec. 2002.
[33] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn. Advances Cryptology*, 1999, pp. 223–238.
[34] M. Lichman. (1999) UCI Data. [Online]. Available: http://archive.ics.uci.edu/ml
[35] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Third Int. Conf. Applied Cryptography and Network Security*, 2005, pp. 442–455.
[36] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 2110–2118.
[37] D. Papadopoulos, S. Papadopoulos, and N. Triandopoulos, "Taking authenticated range queries to arbitrary dimensions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, New York, NY, USA, 2014, pp. 819–830.
[38] J. Kawamoto and M. Yoshikawa, "Private range query by perturbation and matrix based encryption," in *Proc. 6th IEEE Int. Conf. Digital Information Management*, Sep. 2011, pp. 211–216.
[39] B. Wang, Y. Hou, M. Li, H. Wang, H. Li, and F. Li, "Tree-based multi-dimensional range search on encrypted data with enhanced privacy," in *Proc. 10th Int. Conf. Secur. Privacy Commun. Netw.*, 2015, pp. 374–394.

Wei Yang is an associate professor in School of Computer Science and Technology at the University of Science and Technology of China. In 2007, he received his Ph.D. degree in computer science from University of Science and Technology of China and was awarded the Deans Prize of Chinese Academy of Sciences. His research interests include information security, quantum information and wireless networks. He has authored or co-authored over 120 technical papers in major international journals and conferences. In 2014, he got the Natural Science Award of Ministry of Education of People's Republic of China. In 2016, he won the Best Paper Award at ACM UbiComp.

Yangyang Geng is a Ph.D. candidate in School of Computer Science and Technology, University of Science and Technology of China. His research interests include data privacy and cloud computing.



Lu Li is an associate professor at Yancheng Teachers University. He received his Ph.D. degree in computer science from University of Science and Technology of China, Hefei, in 2015. His research interests include secure multi-party computation, crowdsourcing, data security and privacy, and applied cryptography.

Xike Xie is currently a professor in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include data uncertainty, spatiotemporal databases, and mobile computing.

Liusheng Huang is a professor in School of Computer Science and Technology at the University of Science and Technology of China. His research interests include Internet of Things, cloud computing, and data privacy.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on May 06,2020 at 00:20:29 UTC from IEEE Xplore. Restrictions apply.