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/TSC.2021.3065240, IEEE Transactions on Services Computing

Achieving Efficient and Privacy-Preserving Set Containment Search over Encrypted Data

Yandong Zheng, Rongxing Lu, *Fellow, IEEE,* Yunguo Guan, Jun Shao, and Hui Zhu, *Senior Member, IEEE*

***Abstract*—Set containment search, which aims to retrieve all set records containing a specific query set, has received considerable attention. Meanwhile, due to the dramatic growth of data, data owners tend to outsource their data to the cloud and deploy the cloud server to offer the set containment search services. However, as the cloud server is not fully trustable and the data may be sensitive, a straightforward strategy for the data owners is to encrypt the data before outsourcing them. Although the encryption technique can preserve data privacy, it inevitably hinders the functionality of set containment search. Many existing studies on the set containment search over outsourced data still suffer from the search efficiency and security issues. In this paper, aiming at the above issues, we propose an efficient and privacy-preserving set containment search scheme. Specifically, we first deploy an asymmetric scalar-product-preserving encryption technique to design a set containment/intersection encryption (SCIE-Enc) scheme. Then, we build a radix tree to represent the set records. Based on the radix tree and SCIE-Enc construction, we present our scheme that can achieve efficient set containment search while preserving the privacy of set records, query sets, and query results, as indicated in our security analysis and performance evaluation.**

***Index Terms*—Set containment search, cloud computing, scalar product computation, radix tree, privacy-preserving**

I. Introduction

The advance of Internet of Things (IoT) [1], social networking [2], and applied artificial intelligence [3], among others, has promoted an increasing amount of data generated day by day. As reported in [4], there will be around 40 trillion gigabytes of data in 2020 and the big data market is expected to increase by 14%. Obviously, the huge volumes of data put a great pressure on data owners' storage and computing capability. As a result, many organizations and individuals tend to outsource their local data to a powerful cloud and deploy the cloud to manage the data. However, since the cloud server is not fully trustable and the data may contain some sensitive information, a straightforward strategy for the data owners is to outsource encrypted data to the cloud [5], [6]. Although the data encryption technique can protect the data confidentiality in cloud, it will inevitably affect the data utility, e.g., the popular set containment search. Essentially, the concept of set containment search is to find the containment relationship between sets, which has been employed in many real-world applications. For instance, in the field of job hiring, both the skills mastered by a job applicant and the required skills for a job can be regarded as sets. Then, for a given job recruiter, the set containment search can help him/her to identity which job applicants are qualified for the job, as shown in Fig. 1. Basically, the set containment search can be defined as "*return the identities of set records in the dataset $\mathcal{X}$ that contain the query set Q*", i.e., $\{\mathrm{id}_i | Q \subseteq X_i, X_i \in \mathcal{X}\}$, where id_i is the identity of X_i. Although the set containment search has been widely studies [7]–[10], most of existing studies focus on the set containment search over the plaintext domain and are not applicable to our encrypted outsourced data scenario.

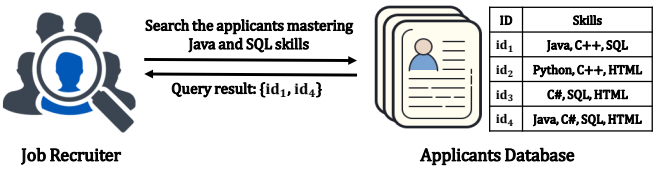


Fig. 1: An example of set containment search in job hiring

As for the set containment search over outsourced data, one possible solution is to transform set records into some fix-dimensional binary vectors. Then, the set containment search problem can be converted into a scalar product based vector search problem. For the clear description, we first give an example to illustrate how to transform a set containment search into a scalar product based vector search. Suppose that $\mathcal{X} = \{X_1 = \{e_1, e_2, e_4\}, X_2 = \{e_1, e_2\}, X_3 = \{e_2, e_4\}\}$ is a dataset with 3 set records and $Q = \{e_2, e_4\}$ is a query set. Then, $\mathcal{X}$ can be transformed into a collection of binary vectors $\{\mathbf{x}_1 = (1,1,0,1), \mathbf{x}_2 = (1,1,0,0), \mathbf{x}_3 = (0,1,0,1)\}$, where the j-th element of $\mathbf{x}_i$ is set to be 1 *iff* X_i contains e_j for $j = 1,2,3,4$. Similarly, the query set $Q = \{e_2, e_4\}$ can be transformed into $\mathbf{q} = (0,1,0,1)$. In this case, "$Q \subseteq X_i$" is equivalent to "$\mathbf{x}_i \circ \mathbf{q} = |\mathbf{q}|$", where "$\circ$" denotes the scalar product computation and the value $|\mathbf{q}| = \mathbf{q} \circ \mathbf{q}$. Then, searching the sets in $\mathcal{X}$ containing the query set Q is equivalent to search the binary vectors in $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ whose scalar product with $\mathbf{q}$ is equal to $|\mathbf{q}|$. In other words, the set containment search problem can be transformed into the scalar product based vector search problem.

The scalar product based vector search problem over encrypted outsourced data can be solved by function-hiding scalar product encryption schemes [11]–[15]. This is because the function-hiding scalar product encryption schemes can reveal the scalar product between two vectors according to their ciphertexts. With this nice property, the cloud server

Y. Zheng, R. Lu and Y. Guan are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: yzheng8@unb.ca, rlu1@unb.ca, yguan4@unb.ca).

J. Shao is with School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, 310018, China (e-mail: chn.junshao@gmail.com).

H. Zhu is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, 710071, China (e-mail: zhuhui@xidian.edu.cn).

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/TSC.2021.3065240, IEEE Transactions on Services Computing

can process the scalar product based vector search problem over encrypted data and find out the vectors satisfying the query vector. Nevertheless, these function-hiding scalar product encryption schemes suffer from the search efficiency issue, because they are either based on the time-consuming bilinear pairing [11]–[14] or built upon a variant of Paillier homomorphic encryption scheme [16]. For the scheme in [15], it is an asymmetric scalar-product-preserving encryption (ASPE) scheme and encrypts the vectors with a real domain matrix. The matrix encryption is efficient as it is a symmetric encryption and only involves matrix multiplication of real domain. Thus, compared with the schemes in [11]–[14], the ASPE scheme is the most efficient candidate to achieve scalar product based vector search. However, the ASPE scheme [15] still has a disadvantage in search efficiency. Specifically, in the ASPE scheme [15], each vector is encrypted to be a ciphertext and outsourced to the cloud. Given a query vector, the cloud server needs to traverse all vectors in the database to find out the query result. In other words, the computational complexity of search in ASPE scheme is linear to the size of the database. When the database is large, the search process is inefficient. Meanwhile, the ASPE scheme also suffers from the security issue, i.e., it cannot resist against the known-plaintext attacks as shown in [17]. Therefore, it is still challenging to achieve efficient and privacy-preserving set containment search.

Aiming at the above challenges, in this work, we propose an efficient and privacy-preserving set containment search scheme over encrypted data, which outperforms the existing schemes. Specifically, the main contributions of this paper are three-fold:

- First, we deploy the idea of the ASPE scheme to design a set containment/intersection encryption (SCIE-Enc) construction, which can support the privacy-preserving set containment query and set intersection query. Given the ciphertexts of two sets X_i and Q, the set containment query can determine whether Q is a subset of X_i, i.e., $Q \stackrel{?}{\subseteq} X_i$, while preserving the plaintext of X_i and Q. Similarly, the set intersection query can determine whether X_i has an intersection with Q, i.e., $X_i \cap Q \stackrel{?}{=} \emptyset$, without disclosing X_i and Q.
- Second, we propose an efficient and privacy-preserving set containment search scheme by designing a radix tree to represent the set records and employing our SCIE-Enc construction to encrypt and search the radix tree. The proposed scheme can support efficient set containment search while preserving the privacy of set records in the cloud, the query sets, as well as the query results.
- Finally, we analyze the security of our SCIE-Enc construction and our proposed scheme. The result shows that our SCIE-Enc construction is selectively simulation-secure and it can resist against the known-plaintext attacks. In addition, we conduct extensive experiments to validate the efficiency of our scheme. The results indicate that our proposed scheme is really privacy-preserving and efficient in set containment search.

The remainder of this paper is organized as follows. We first define our models and design goals in Section II, and describe some preliminaries in Section III. In Section IV, we present our scheme, followed by security analysis and performance evaluation in Section V and Section VI, respectively. Finally, we discuss some related work in Section VII and draw our conclusion in Section VIII.

II. Models and Design Goals

In this section, we formalize our system model, security model, and identify our design goals.

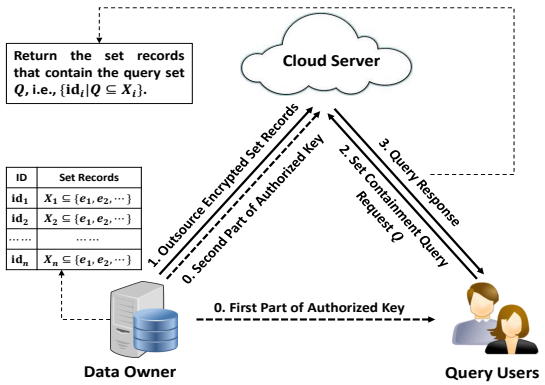


Fig. 2: System model under consideration

A. System Model

In our system model, we consider a typical cloud-based set containment search model, which consists of three kinds of entities, namely a data owner, a cloud server and a collection of authorized query users $\mathcal{U} = \{\mathtt{U}_1, \mathtt{U}_2, \cdots\}$, as shown in Fig. 2.

• *Data Owner:* The data owner has accumulated a collection of set records $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$. Each $X_i \in \mathcal{X}$ has a unique identity id_i and it is a subset of $\mathcal{E}$, i.e., $X_i \subseteq \mathcal{E}$, where $\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$ is the collection of all elements. For maximizing the utility of $\mathcal{X}$, the data owner is willing to offer the set containment search service to the users. Meanwhile, since the data owner is usually not powerful in computing and storage, it tends to outsource the data $\mathcal{X}$ to a powerful cloud and deploy the cloud server to manage the data. Since the data $\mathcal{X}$ may contain some sensitive information and the cloud server is not fully trustable, the data owner prefers to encrypt $\mathcal{X}$ before outsourcing them to the cloud.

• *Cloud Server:* The cloud server is powerful in both storage and computing capabilities, which can be regarded as a link between the data owner and the query users in our model. On the one hand, it is responsible for storing the data $\mathcal{X}$ outsourced by the data owner. On the other hand, it offers the set containment search service to the query users. Specifically, upon receiving a query request from a query user, the cloud server will search the outsourced data to find out the set records satisfying the set containment search criterion, and return them to the query user.

• *Authorized Query Users $\mathcal{U} = \{\mathtt{U}_1, \mathtt{U}_2, \cdots\}$:* In our system model, there are a collection of authorized query users $\mathcal{U} = \{\mathtt{U}_1, \mathtt{U}_2, \cdots\}$ and each $\mathtt{U}_i \in \mathcal{U}$ can enjoy the set containment search service offered by the cloud server. Meanwhile, for each query user, when he/she registers in the system, the data

owner will randomly split an authorized key into two parts, and securely distribute these two key parts to the query user and the cloud, respectively, as shown in Fig. 2.

B. Security Model

In our security model, we consider the data owner is trustable and will correctly outsource the encrypted set records to the cloud server. For the cloud server, we assume it to be *honest-but-curious*. Specifically, it will honestly store the encrypted set records outsourced by the data owner and offer the set containment service to the query users. However, it may be curious about some private information such as the plaintext of the set records stored in the cloud. In addition, when processing a set containment search request, it may attempt to obtain the plaintext of the query set Q and the query result (the identities of set records that contain the query set $\{\mathrm{id}_i | Q \subseteq X_i, X_i \in \mathcal{X}\}$). For the authorized query users, they are considered to be *honest-but-curious*. In other words, they will honestly launch set containment search requests to the cloud server, but they may be curious about the query sets and query results of other query users. Besides, we assume there is no collusion between the cloud server and the query users in our model. Note that there may be other passive or active attacks such as denial of service and data pollution attacks. However, since this work focuses on privacy and efficiency in set containment search over encrypted data, those attacks are beyond the scope of this paper and will be discussed in our future work.

C. Design Goals

In this paper, our goal is to design an efficient and privacy-preserving set containment search scheme under above-mentioned system model and security model. In particular, the following two objectives should be satisfied.

- *Privacy Preservation:* The basic requirement of our scheme is privacy preservation. That is, the encrypted set records stored in the cloud should be kept secret from the cloud server. The query sets and query results should also be kept private from the cloud server and other query users.
- *Efficiency:* In order to achieve the above privacy requirement, additional computational cost will be inevitably incurred, i.e., processing the set containment search over the encrypted data will undoubtedly increase the computational cost compared with those doing over the plaintext sets. Therefore, in the proposed scheme, we also aim to improve the efficiency of set containment search.

III. Preliminaries

In this section, we first formally define the set containment search problem and briefly review the radix tree and an asymmetric scalar-product-preserving encryption (ASPE) technique [15]. Then, we present a brief introduction of our SCIE-Enc construction and its security. For the clear description, we first list some used notations in Table I.

TABLE I: Summary of notations

Symbol	Description
$\mathcal{E} = \{e_1, \cdots, e_d\}$	The collection of all elements
$\mathcal{X} = \{X_1, \cdots, X_n\}$	The collection of all sets, $X_i \subseteq \mathcal{E}$
$\mathbf{x}_i = \{x_{i1}, x_{i2}, \cdots, x_{id}\}$	The binary representation of X_i
id_i	The identity of X_i
$Q \subseteq \mathcal{E}$	The query set
$\mathbf{M}$ and $\mathbf{M}^{-1}$	Matrix and its inverse matrix
CT_{X_i}	The ciphertext of X_i
TKC_Q	The set containment token of Q
TKIS_Q	The set intersection token of Q
T	The radix tree
X_{node}	The set in an internal tree node $node$
$\mathcal{ID}_{leafNode}$	Sets' identities in a leaf node $leafNode$
$\mathcal{L}_C$	Leakage of the set containment query
$\mathcal{L}_{IS}$	Leakage of the set intersection query
$\mathbf{M}_1$ and $\mathbf{M}_2$	The authorized key for the query users
ak	The access key for the query users
ssk	The session key for the query users

A. Definition of Set Containment Search

Let $\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$ be the collection of all elements and $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ denote n set records. Each $X_i \in \mathcal{X}$ has a unique identity id_i and it is a subset of $\mathcal{E}$, i.e., $X_i \subseteq \mathcal{E}$. Then, the set containment search can be defined as follows.

Definition 1 (Set Containment Search): Given a query set $Q \subseteq \mathcal{E}$ and a collection of set records $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$, the set containment search is to search all sets in $\mathcal{X}$ and find out the sets that contain Q, i.e., $\{\mathrm{id}_i \in \mathcal{X} | Q \subseteq X_i, X_i \in \mathcal{X}\}$.

TABLE II: A collection of sets

ID	Set
id_1	$X_1 = \{e_5, e_6, e_7\}$
id_2	$X_2 = \{e_1, e_2, e_4, e_5\}$
id_3	$X_3 = \{e_1\}$
id_4	$X_4 = \{e_1, e_2, e_4, e_6\}$
id_5	$X_5 = \{e_5, e_6, e_7\}$

Example 1: Table II lists a collection of sets $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5\}$. When the query set is $Q = \{e_2, e_4\}$, the query result will be $\{\mathrm{id}_2, \mathrm{id}_4\}$, because X_2 and X_4 contain the query set Q, i.e., $Q \subseteq X_2$ and $Q \subseteq X_4$.

B. Radix Tree

As radix tree is a compact prefix tree [18], we first introduce the concept of prefix tree. The prefix tree is a kind of ordered tree that can be deployed to store a collection of sets $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$, where $X_i \subseteq \mathcal{E}$ for $i = 1, 2, \cdots, n$. Due to the ordered nature of the prefix tree, the elements in each set should also be orderly placed. Suppose that $\mathcal{E} = \{e_1, e_2, \cdots, e_7\}$ is the collection of all elements. The order of elements is $\{e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_7\}$ and it is called global order. Then, for a given set, its elements should be placed in accordance with the global order. For example, $\{e_2, e_4, e_7\}$ should be placed as $\{e_2 \rightarrow e_4 \rightarrow e_7\}$ rather than $\{e_4 \rightarrow e_2 \rightarrow e_7\}$. Thus, the collection of ordered sets can be organized as a prefix tree. In the prefix tree, each internal node stores an element in $\mathcal{E}$. Each leaf node stores a collection of

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/TSC.2021.3065240, IEEE Transactions on Services Computing

sets' identities. Meanwhile, the sets whose identities store in a given leaf node contain the same elements and these elements are exactly the same as that storing in the path from the parent of the leaf node to the root node. In addition, when two paths share the same prefix elements, the paths corresponding to the prefix can be merged.

Example 2: Fig. 3 presents an example of a prefix tree, which is associated with the collection of sets in Table II. In this figure, the identities of sets $\{X_1, X_2, X_3, X_4, X_5\}$ are stored in the leaf nodes. For the leaf node storing id_2, the path from the parent of its leaf node to the root node exactly contains the same elements as X_2. Meanwhile, from this figure, we can see that the paths of id_2 and id_4 share a prefix $\{e_1, e_2, e_4\}$, so the paths corresponding to the prefix are merged.

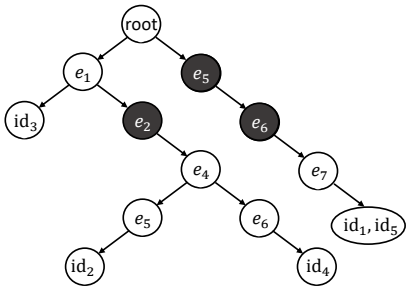


Fig. 3: The prefix tree for $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5\}$, where $X_1 = X_5 = \{e_5, e_6, e_7\}$, $X_2 = \{e_1, e_2, e_4, e_5\}$, $X_3 = \{e_1\}$, $X_4 = \{e_1, e_2, e_4, e_6\}$

The radix tree is a compact prefix tree. Specifically, in the prefix tree, when an internal node only has one child, it can be merged with its child. In this way, the prefix tree can be compressed to be a radix tree.

Example 3: As shown in Fig. 3, the shaded nodes with values e_2, e_5 and e_6 only have one child, so they can be merged with their children. As shown in Fig. 4, the node e_2 can be merged with its child e_4 and they form a new node $\{e_2, e_4\}$. The nodes e_5, e_6 and e_7 can be merged together to be a new node $\{e_5, e_6, e_7\}$. After the merging, the prefix tree in Fig. 3 can be compressed to be a radix tree in Fig. 4.

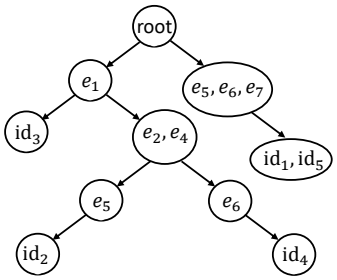


Fig. 4: The radix tree for $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5\}$, where each node with a single child has been merged with its child.

C. The ASPE Technique

The ASPE technique was proposed for the secure kNN queries in [15]. In the ASPE scheme, the data records are d-dimensional vectors and there are two types of vectors, i.e, (i) the vectors in the database; and (ii) the query vector. The main idea of the ASPE scheme is to encrypt these two types of vectors in different ways so that the scalar product between them can be preserved. Specifically, the ASPE technique $\Pi_{\mathsf{ASPE}} = (\mathsf{AspeSetup}, \mathsf{AspeEnc}, \mathsf{AspeTokenGen})$ can be defined as follows.

- $\mathsf{AspeSetup}(d)$: The setup algorithm takes the parameter d as input, and outputs a random invertible matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ as the secret key, where $\mathbb{R}$ denotes the real domain.
- $\mathsf{AspeEnc}(\mathbf{M}, \mathbf{x})$: The encryption algorithm takes the secret key $\mathbf{M}$ and a d-dimensional vector $\mathbf{x}$ in the database as input, and outputs a ciphertext $\mathrm{CT}_{\mathbf{x}} = \mathbf{x}\mathbf{M}$.
- $\mathsf{AspeTokenGen}(\mathbf{M}, \mathbf{q})$: The token generation algorithm takes $\mathbf{M}$ and a d-dimensional query vector $\mathbf{q}$ as input and outputs a query token $\mathrm{TK}_{\mathbf{q}} = r\mathbf{q}(\mathbf{M}^{-1})^T$, where $r \in \mathbb{R}$ is a random positive real number and $\mathbf{M}^{-1}$ is the inverse matrix of $\mathbf{M}$.

The ASPE technique can preserve the scalar product relationship between the query vector and the vectors in the database. Let $\mathrm{CT}_{\mathbf{x}_1}$ and $\mathrm{CT}_{\mathbf{x}_2}$ respectively denote the ciphertexts of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Let $\mathrm{TK}_{\mathbf{q}}$ be the query token of the query vector $\mathbf{q}$. Then, the ASPE technique satisfies the scalar-product-preserving property. That is,

$$\mathrm{CT}_{\mathbf{x}_1} \circ \mathrm{TK}_{\mathbf{q}} \geq \mathrm{CT}_{\mathbf{x}_2} \circ \mathrm{TK}_{\mathbf{q}} \Leftrightarrow \mathbf{x}_1 \circ \mathbf{q} \geq \mathbf{x}_2 \circ \mathbf{q}$$

where "$\circ$" denotes the scalar product operation.

It is worth noting that the ASPE technique can be used to achieve efficient scalar-product-based kNN queries over encrypted database. However, the ASPE technique cannot resist against the known-plaintext attacks as pointed out in [17]. In the following Subsection IV-A, we will propose our SCIE-Enc construction by introducing more random numbers into the ASPE technique, so that our construction can resist against the known-plaintext attacks.

D. A Brief Introduction of Our SCIE-Enc Construction

In this subsection, we briefly introduce our set containment/intersection encryption (SCIE-Enc) scheme, which can support both the set containment query and set intersection query. The set containment query is to determine whether the query set is a subset of a set in the database, and the set intersection query is to determine whether the query set has an intersection with a set in the database. The main idea of our SCIE-Enc construction is to encrypt the set records in the database, and generate the set containment query tokens as well as set intersection query tokens in different ways. In specific, our SCIE-Enc construction $\Pi_{\mathsf{SCIE-Enc}} = (\mathsf{ScieSetup}, \mathsf{ScieEnc}, \mathsf{ScieTokenC}, \mathsf{ScieQueryC}, \mathsf{ScieTokenIS}, \mathsf{ScieQueryIS})$ can be defined as follows.

- $\mathsf{ScieSetup}(d)$: Given the parameter d, the setup algorithm outputs a secret key sk.
- $\mathsf{ScieEnc}(sk, X_i)$: Given the secret key sk, a set $X_i \subseteq \mathcal{E}$ can be encrypted to be a ciphertext CT_{X_i}.
- $\mathsf{ScieTokenC}(sk, Q)$: Given the secret key sk and a query set Q, the set containment query token generation algorithm outputs a query token TKC_Q for Q.

- $\mathsf{ScieQueryC}(\mathrm{CT}_{X_i}, \mathrm{TKC}_Q)$: Given CT_{X_i} and TKC_Q, the set containment query algorithm returns 1 *iff* $Q \subseteq X_i$. Otherwise, it returns 0.
- $\mathsf{ScieTokenIS}(sk, Q)$: Given the secret key sk and a query set, the set intersection query token generation algorithm generates a query token TKIS_Q for Q.
- $\mathsf{ScieQueryIS}(\mathrm{CT}_{X_i}, \mathrm{TKIS}_Q)$: Given CT_{X_i} and TKIS_Q, the query algorithm returns 1 *iff* the intersection between X_i and Q is not empty, i.e., $X_i \cap Q \neq \emptyset$. Otherwise, it returns 0.

Security: The semantic security of our SCIE-Enc construction is proved in a real/ideal experiment setting, which can subsume the traditional security definitions in the indistinguishability setting as described in [19]. Before defining the security, we first define the leakage in our SCIE-Enc construction. In the set containment query phase, given the ciphertext CT_{X_i} and query token TKC_Q, the leakage is the query matching result $\mathcal{L}_C = \mathsf{ScieQueryC}(\mathrm{CT}_{X_i}, \mathrm{TKC}_Q)$. In the set intersection query phase, given the ciphertext CT_{X_i} and query token TKIS_Q, the leakage is the query matching result $\mathcal{L}_{IS} = \mathsf{ScieQueryIS}(\mathrm{CT}_{X_i}, \mathrm{TKIS}_Q)$. With the leakage $\mathcal{L}_C$ and $\mathcal{L}_{IS}$, the real and ideal experiments can be defined as follows.

Real experiment: In the real experiment, the challenger and a probabilistic polynomial-time adversary $\mathcal{A}$ interact as follows.

- *Setup:* In the setup phase, $\mathcal{A}$ chooses a plaintext set $X_i \in \mathcal{X}$ and sends it to the challenger. Meanwhile, the challenger executes the $\mathsf{ScieSetup}(d)$ algorithm to generate a secret key sk.
- *Set containment query phase 1:* $\mathcal{A}$ adaptively chooses p_1 query sets $\{Q_j\}_{j=1}^{p_1}$ and sends them to the challenger, where p_1 is a polynomial number. Then, the challenger executes the set containment query token generation algorithm to generate p_1 tokens, i.e., $\{\mathrm{TKC}_{Q_j} = \mathsf{ScieTokenC}(sk, Q_j)\}_{j=1}^{p_1}$ and returns them to $\mathcal{A}$.
- *Set intersection query phase 1:* $\mathcal{A}$ adaptively chooses p_1' query sets $\{Q_j\}_{j=1}^{p_1'}$ and sends them to the challenger, where p_1' is a polynomial number. Then, the challenger executes the set intersection query token generation algorithm to generate p_1' tokens, i.e., $\{\mathrm{TKIS}_{Q_j} = \mathsf{ScieTokenIS}(sk, Q_j)\}_{j=1}^{p_1'}$ and returns them to $\mathcal{A}$.
- *Challenge phase:* The challenger executes the encryption algorithm and outputs a ciphertext $\mathrm{CT}_{X_i} = \mathsf{ScieEnc}(sk, X_i)$ to $\mathcal{A}$.
- *Set containment query phase 2:* $\mathcal{A}$ runs the set containment query phase 1 again to obtain $p_2 - p_1$ query sets' set containment query tokens, i.e., $\{\mathrm{TKC}_{Q_j} = \mathsf{ScieTokenC}(sk, Q_j)\}_{j=p_1+1}^{p_2}$, where p_2 is a polynomial number.
- *Set intersection query phase 2:* $\mathcal{A}$ runs the set intersection query phase 1 again to obtain $p_2' - p_1'$ query sets' set intersection query tokens, i.e., $\{\mathrm{TKIS}_{Q_j} = \mathsf{ScieTokenIS}(sk, Q_j)\}_{j=p_1'+1}^{p_2'}$, where p_2' is a polynomial number.

In the real experiment, the view of $\mathcal{A}$ is $\mathrm{View}_{\mathcal{A},\mathrm{Real}} = \{X_i, \mathrm{CT}_{X_i}, \{Q_j, \mathrm{TKC}_{Q_j}\}_{j=1}^{p_2}, \{Q_j, \mathrm{TKIS}_{Q_j}\}_{j=1}^{p_2'}\}$.

Ideal experiment: In the ideal experiment, a PPT adversary $\mathcal{A}$ and a simulator Sim with leakage $\mathcal{L} = \{\mathcal{L}_C, \mathcal{L}_{IS}\}$ interact as follows.

- *Setup:* In the setup phase, $\mathcal{A}$ chooses a plaintext set $X_i \in \mathcal{X}$ and sends it to the simulator Sim. Then, Sim randomly selects a ciphertext CT_{X_i}.
- *Set containment query phase 1:* $\mathcal{A}$ adaptively chooses p_1 query sets $\{Q_j\}_{j=1}^{p_1}$ and sends them to Sim, where p_1 is a polynomial number. Then, Sim takes the leakage $\{\mathcal{L}_C = \mathsf{ScieQueryC}(\mathrm{CT}_{X_i}, \mathrm{TKC}_{Q_j})\}_{j=1}^{p_1}$ as input and generates p_1 tokens $\{\mathrm{TKC}_{Q_j}\}_{j=1}^{p_1}$. Finally, it returns them to $\mathcal{A}$.
- *Set intersection query phase 1:* $\mathcal{A}$ adaptively chooses p_1' query sets $\{Q_j\}_{j=1}^{p_1'}$ and sends them to Sim, where p_1' is a polynomial number. Then, Sim takes the leakage $\{\mathcal{L}_{IS} = \mathsf{ScieQueryIS}(\mathrm{CT}_{X_i}, \mathrm{TKIS}_{Q_j})\}_{j=1}^{p_1'}$ as input and generates p_1' tokens $\{\mathrm{TKIS}_{Q_j}\}_{j=1}^{p_1'}$. Finally, it returns them to $\mathcal{A}$.
- *Challenge phase:* Sim returns CT_{X_i} to $\mathcal{A}$.
- *Set containment query phase 2:* $\mathcal{A}$ runs the set containment query phase 1 again to obtain $p_2 - p_1$ query sets' set containment query tokens, i.e., $\{\mathrm{TKC}_{Q_j} = \mathsf{ScieTokenC}(sk, Q_j)\}_{j=p_1+1}^{p_2}$.
- *Set intersection query phase 2:* $\mathcal{A}$ runs the set intersection query phase 1 again to obtain $p_2' - p_1'$ query sets' set intersection query tokens, i.e., $\{\mathrm{TKIS}_{Q_j} = \mathsf{ScieTokenIS}(sk, Q_j)\}_{j=p_1'+1}^{p_2'}$.

In the ideal experiment, the view of $\mathcal{A}$ is $\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}} = \{X_i, \mathrm{CT}_{X_i}, \{Q_j, \mathrm{TKC}_{Q_j}\}_{j=1}^{p_2}, \{Q_j, \mathrm{TKIS}_{Q_j}\}_{j=1}^{p_2'}\}$.

Definition 2 (Security of SCIE-Enc Construction [19]): The SCIE-Enc construction is selectively simulation-secure with respect to the leakage $\mathcal{L} = \{\mathcal{L}_C, \mathcal{L}_{IS}\}$ *iff* for all PPT adversaries $\mathcal{A}$ issuing polynomial numbers of set containment queries and set intersection queries, there exists a simulator Sim such that the advantage that $\mathcal{A}$ can distinguish the real and ideal experiment is negligible.

IV. Our Proposed Scheme

In this section, we present our proposed set containment search scheme. Before delving into the details of our proposed scheme, we first introduce the detailed SCIE-Enc construction and an efficient set containment search algorithm, which serve as the building blocks of our proposed scheme.

A. Detailed SCIE-Enc Construction

In this subsection, we present the detailed SCIE-Enc construction, which can support both set containment queries and set intersection queries. Let $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ denote the collection of set records, where $X_i \subseteq \mathcal{E}$ and $\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$ is the collection of all elements. Then, the SCIE-Enc construction $\Pi_{\mathsf{SCIE-Enc}} = (\mathsf{ScieSetup}, \mathsf{ScieEnc}, \mathsf{ScieTokenC}, \mathsf{ScieQueryC}, \mathsf{ScieTokenIS}, \mathsf{ScieQueryIS})$ can be defined as follows.

- $\mathsf{ScieSetup}(d)$: Given the parameter d, i.e., the cardinality of $\mathcal{E}$, the setup algorithm outputs a random invertible matrix $\mathbf{M} \in \mathbb{R}^{(d+3)\times(d+3)}$ as the secret key, where $\mathbb{R}$ denotes the real domain.

- $\mathsf{ScieEnc}(\mathbf{M}, X_i)$: Given the secret key $\mathbf{M}$, a set $X_i \subseteq \mathcal{E}$ can be encrypted as follows.
  (a) Transform X_i to be a d-dimensional binary vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})$, where $x_{ij} = 1$ if $e_j \in X_i$ and $x_{ij} = 0$ otherwise.
  (b) Extend $\mathbf{x}_i$ to be a $d+3$ dimensional vector $\mathbf{x}'_i = (r_2 * \mathbf{x}_i, r_2, r_1, r'_1)$, where r_1, r'_1, and r_2 are random real numbers satisfying $r_1 > 0$, $r'_1 > 0$, and $r_2 > 2 * \max\{r_1, r'_1\}$.
  (c) Encrypt $\mathbf{x}'_i$ as $\mathrm{CT}_{X_i} = \mathbf{x}'_i\mathbf{M} = (r_2 * \mathbf{x}_i, r_2, r_1, r'_1)\mathbf{M}$.
- $\mathsf{ScieTokenC}(\mathbf{M}, Q)$: Given the secret key $\mathbf{M}$, a set containment query token for the query set Q can be generated as follows.
  (a) Transform Q to be a d-dimensional vector $\mathbf{q} = (q_1, q_2, \cdots, q_d)$, where $q_j = 1$ if $e_j \in Q$ and $q_j = 0$ otherwise.
  (b) Extend $\mathbf{q}$ to be a $d+3$ dimensional vector $\mathbf{q}' = (r_4 * \mathbf{q}, -r_4 * |\mathbf{q}|, r_3, r'_3)$, where r_3, r'_3, and r_4 are random real numbers satisfying $r_3 > 0$, $r'_3 > 0$, and $r_4 > \max\{r_3, r'_3\}$.
  (c) Generate the set containment query token for $\mathbf{q}'$ as $\mathrm{TKC}_Q = \mathbf{q}'(\mathbf{M}^{-1})^T = (r_4 * \mathbf{q}, -r_4 * |\mathbf{q}|, r_3, r'_3)(\mathbf{M}^{-1})^T$.
- $\mathsf{ScieQueryC}(\mathrm{CT}_{X_i}, \mathrm{TKC}_Q)$: Let CT_{X_i} and TKC_Q respectively denote the ciphertext of X_i and the set containment query token of Q. If $\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0$, the query algorithm returns 1. Otherwise, it returns 0.
- $\mathsf{ScieTokenIS}(\mathbf{M}, Q)$: Given the secret key $\mathbf{M}$, a set intersection token for Q can be generated as follows.
  (a) Transform Q to be a d-dimensional vector $\mathbf{q} = (q_1, q_2, \cdots, q_d)$, where $q_j = 1$ if $e_j \in Q$ and $q_j = 0$ otherwise.
  (b) Extend $\mathbf{q}$ to be a $d+3$ dimensional vector $\mathbf{q}' = (r_6 * \mathbf{q}, 0, -r_5, -r'_5)$, where r_5, r'_5, and r_6 are random real numbers satisfying $r_5 > 0$, $r'_5 > 0$, and $r_6 > \max\{r_5, r'_5\} > 0$.
  (c) Generate the set intersection query token for Q as $\mathrm{TKIS}_Q = \mathbf{q}'(\mathbf{M}^{-1})^T = (r_6 * \mathbf{q}, 0, -r_5, -r'_5)(\mathbf{M}^{-1})^T$.
- $\mathsf{ScieQueryIS}(\mathrm{CT}_{X_i}, \mathrm{TKIS}_Q)$: Let CT_{X_i} and TKIS_Q respectively denote the ciphertext of X_i and the set intersection token of Q. If $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0$, the query algorithm returns 1. Otherwise, it returns 0.

Correctness: The SCIE-Enc construction is correct *iff* both the set containment query and set intersection query can return the correct query result.

• *Set containment query*: According to the transformation method from sets to binary vectors, it is easy to deduce that if Q is a subset of X_i, we have $\mathbf{x}_i \circ \mathbf{q} = |\mathbf{q}|$. That is, $\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0$. Then, when $\mathbf{x}_i$ and $\mathbf{q}$ are respectively encrypted into a ciphertext CT_{X_i} and a token TKC_Q. The set containment query is correct *iff*

$$\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0 \Leftrightarrow \mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0. \tag{1}$$

Theorem 1: Eq. (1) holds.

Proof. We respectively prove the necessity and sufficiency of Eq. (1).

• Necessity: $\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0$. First, we have

$$\begin{aligned} &\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q = \mathbf{x}'_i \circ \mathbf{q}' \\ &= (r_2 * \mathbf{x}_i, r_2, r_1, r'_1) \circ (r_4 * \mathbf{q}, -r_4 * |\mathbf{q}|, r_3, r'_3) \\ &= r_2 * r_4 * (\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}|) + r_1 * r_3 + r'_1 * r'_3. \end{aligned}$$

Since $r_2 > 2 * \max\{r_1, r'_1\} > 0$, $r_4 > \max\{r_3, r'_3\} > 0$. Then, we can deduce that

$$\begin{aligned} r_2 * r_4 &> 2 * \max\{r_1, r'_1\} * \max\{r_3, r'_3\} \\ &> r_1 * r_3 + r'_1 * r'_3 > 0. \end{aligned}$$

Since $r_1 > 0$, $r'_1 > 0$, $r_3 > 0$, $r'_3 > 0$, we have $r_1 * r_3 + r'_1 * r'_3 > 0$. That is, $r_2 * r_4 > r_1 * r_3 + r'_1 * r'_3 > 0$. Thus, we have

$$\begin{aligned} \mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q &= r_2 * r_4 * (\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}|) + r_1 * r_3 + r'_1 * r'_3 \\ &< r_2 * r_4 * (\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}|) + r_2 * r_4. \end{aligned}$$

If $\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0$, we can deduce that

$$\begin{aligned} &r_2 * r_4 * (\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}|) + r_2 * r_4 > 0 \\ \Leftrightarrow\ &r_2 * r_4 * (\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| + 1) > 0 \\ \Leftrightarrow\ &\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| + 1 > 0 \ \ (\because r_2 * r_4 > 0) \\ \Leftrightarrow\ &\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| > -1. \end{aligned}$$

Since $\mathbf{x}_i$ and $\mathbf{q}$ are binary vectors, $\mathbf{x}_i \circ \mathbf{q}$ and $|\mathbf{q}|$ are integers. Then, $\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}|$ is also an integer. Thus, we can deduce that

$$\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| > -1 \Leftrightarrow \mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| \geq 0. \tag{2}$$

Meanwhile, since $\mathbf{x}_i \circ \mathbf{q}$ denotes the size of the intersection between $\mathbf{x}_i$ and $\mathbf{q}$, and $|\mathbf{q}|$ denotes the size of $\mathbf{q}$, we have

$$\mathbf{x}_i \circ \mathbf{q} \leq |\mathbf{q}| \Leftrightarrow \mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| \leq 0. \tag{3}$$

By combining Eq. (2) and Eq. (3), we can deduce that

$$\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0.$$

Hence, the necessary holds, i.e., $\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0$.

• Sufficiency: $\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0 \Rightarrow \mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0$. First, we have

$$\begin{aligned} &\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q \\ &= r_2 * r_4 * (\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}|) + r_1 * r_3 + r'_1 * r'_3. \end{aligned}$$

If $\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0$, we can deduce that

$$\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q = r_1 * r_3 + r'_1 * r'_3.$$

Since $r_1 > 0$, $r'_1 > 0$, $r_3 > 0$, and $r'_3 > 0$, we can deduce that

$$\begin{aligned} &\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q = r_1 * r_3 + r'_1 * r'_3 > 0 \\ \Leftrightarrow\ &\mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0. \end{aligned}$$

Hence, the sufficiency holds, i.e., $\mathbf{x}_i \circ \mathbf{q} - |\mathbf{q}| = 0 \Rightarrow \mathrm{CT}_{X_i} \circ \mathrm{TKC}_Q > 0$.

Therefore, Eq. (1) holds, and the set containment query is correct. ■

• *Set intersection query*: Similar to the set containment query, it is easy to deduce that if $Q \cap X_i$ is not an empty set, we have $\mathbf{x}_i \circ \mathbf{q} > 0$. Then, when $\mathbf{x}_i$ and $\mathbf{q}$ are respectively

encrypted into a ciphertext CT_{X_i} and a token TKIS_Q. The set intersection query is correct *iff*

$$\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0 \Leftrightarrow \mathbf{x}_i \circ \mathbf{q} > 0. \tag{4}$$

Theorem 2: Eq. (4) holds.

Proof. We respectively prove the necessity and sufficiency of Eq. (4).

• Necessity: $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} > 0$.

First, we have

$$\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q = r_2 * r_6 * (\mathbf{x}_i \circ \mathbf{q}) - r_1 * r_5 - r_1' * r_5'.$$

Since $r_1 > 0$, $r_1' > 0$, $r_5 > 0$, and $r_5' > 0$, we can deduce that

$$\begin{aligned}\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q &= r_2 * r_6 * (\mathbf{x}_i \circ \mathbf{q}) - r_1 * r_5 - r_1' * r_5' \\ &< r_2 * r_6 * (\mathbf{x}_i \circ \mathbf{q}).\end{aligned}$$

If $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0$, we can deduce that

$$r_2 * r_6 * (\mathbf{x}_i \circ \mathbf{q}) > 0. \tag{5}$$

Meanwhile, since $r_2 > 0$ and $r_6 > 0$, we have

$$\mathbf{x}_i \circ \mathbf{q} > 0.$$

Hence, the necessary of Eq. (4) holds, i.e., $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} > 0$.

• Sufficiency: $\mathbf{x}_i \circ \mathbf{q} > 0 \Rightarrow \mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0$.

First, since $\mathbf{x}_i$ and $\mathbf{q}$ are binary vectors, we can deduce that $\mathbf{x}_i \circ \mathbf{q}$ are non-negative integers. If $\mathbf{x}_i \circ \mathbf{q} > 0$, we have

$$\mathbf{x}_i \circ \mathbf{q} \geq 1. \tag{6}$$

Since $r_2 > 2 * \max\{r_1, r_1'\} > 0$ and $r_6 > \max\{r_5, r_5'\} > 0$, we have

$$\begin{aligned}r_2 * r_6 &> 2 * \max\{r_1, r_1'\} * \max\{r_5, r_5'\} \\ &> r_1 * r_5 + r_1' * r_5' > 0.\end{aligned} \tag{7}$$

By combining Eq (6) and Eq. (7), we can deduce that

$$\begin{aligned}\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q &= r_2 * r_6 * (\mathbf{x}_i \circ \mathbf{q}) - r_1 * r_5 - r_1' * r_5' \\ &\geq r_2 * r_6 - r_1 * r_5 - r_1' * r_5' > 0.\end{aligned}$$

That is, $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0$. Hence, the sufficiency of Eq. (4) holds, i.e., $\mathbf{x}_i \circ \mathbf{q} > 0 \Rightarrow \mathrm{CT}_{X_i} \circ \mathrm{TKIS}_Q > 0$.

Therefore, Eq. (4) holds, and the set intersection query is correct. ■

Note that, compared with the ASPE scheme, our SCIE-Enc construction introduces three random numbers for each ciphertext CT_{X_i}, each set containment query token TKC_Q, and each set intersection query token TKIS_Q. Specifically, CT_{X_i}, TKC_Q and TKIS_Q respectively introduce the random numbers $\{r_1, r_1', r_2\}$, $\{r_3, r_3', r_4\}$, and $\{r_5, r_5', r_6\}$. In Subsection V-A, we will show that these random numbers can guarantee the security of our SCIE-Enc construction.

B. Efficient Set Containment Search Algorithm

The set containment search is the most critical part of our proposed scheme. In this subsection, we will show how to efficiently determine whether a set X contains a query set Q or not. Suppose that $X = \{e_{l_1}, \cdots, e_{l_{|X|}}\}$, $Q = \{e_{t_1}, \cdots, e_{t_{|Q|}}\}$, and the order of the elements in X and Q follows the global order of elements in $\mathcal{E}$. Then, the set containment relationship between X and Q can be determined as follows, which is also shown in Algorithm 1.

- First, compare the first element of X with that of Q, we will have three cases,
  (a) If X's first element is equal to Q's first element, continue to compare their second elements.
  (b) If the global order of X's first element is larger than that of Q's first element, X cannot contain Q's first element. The algorithm returns 0.
  (c) If the global order of X's first element is smaller than that of Q's first element, continue to compare X's second element with Q's first element.
- Second, continue to compare other elements of X and Q in the same way until X contains all elements of Q.

Algorithm 1 Determine whether $Q \subseteq X$ or not

Input: A set $X = \{e_{l_1}, \cdots, e_{l_{|X|}}\}$ and the query set $Q = \{e_{t_1}, \cdots, e_{t_{|Q|}}\}$
Output: The matching result between X_i and Q

```
1: j = 1;
2: for i = 1 to |X| do
       // Case (a)
3:     if e_{l_i} == e_{t_j} then
4:         i = i + 1;
5:         j = j + 1;
       // Case (b)
6:     else if l_i > t_j then // X cannot contain e_{t_j}, so Q ⊈ X
7:         return 0; // Q is not a subset of X
       // Case (c)
8:     else
9:         i = i + 1;
10:    if j > |Q| then
11:        return 1; // Q is a subset of X
12: return 0;
```

The Algorithm 1 determines whether Q is a subset of X by comparing elements of X and Q. Since both X and Q follow the global order, the algorithm can decide that Q is not a subset of X_i when $l_i > t_j$. Similarly, it can decide that Q is a subset of X_i when $j > |Q|$. These early termination conditions help to improve the set containment search efficiency. However, since Algorithm 1 is based on the elements' comparison between X and Q, it is challenging to apply the privacy techniques to protect the privacy of X and Q. In our scheme, for preserving the privacy of both X and Q, we propose a new set containment search algorithm and its privacy can be easily preserved by our SCIE-Enc construction. The new algorithm is to transform X and Q into multiple sets and process the set containment search over these sets.

- First, transform $X = \{e_{l_1}, \cdots, e_{l_{|X|}}\}$ to be $|X|$ sets as

$$X_i' = \{e_{l_1}, \cdots, e_{l_i}\}, \quad 1 \leq i \leq |X|. \tag{8}$$

- Second, transform $Q = \{e_{t_1}, \cdots, e_{t_{|Q|}}\}$ to be $2|Q|$ sets as

$$\begin{cases} Q'_{2j-1} = \{e_{t_1}, e_{t_2}, \cdots, e_{t_j}\} & 1 \le j \le |Q| \\ Q'_{2j} = \{e_{t_j+1}, e_{t_j+2}, \cdots, e_d\} & 1 \le j \le |Q|. \end{cases} \quad (9)$$

- Third, transform the conditions $e_{l_i} = e_{t_j}$ and $l_i > t_j$ in Algorithm 1 to be

$$\begin{cases} e_{l_i} = e_{t_j} & \Rightarrow\ Q'_{2j-1} \subseteq X'_i \\ l_i > t_j & \Rightarrow\ X'_i \cap Q'_{2j} \neq \emptyset. \end{cases}$$

For the transformation of $e_{l_i} = e_{t_j}$, the correctness is obvious. For the transformation of $l_i > t_j$, since $Q'_{2j} = \{e_{t_j+1}, \cdots, e_{d_1}\}$ contains all of the elements whose global order is larger than t_j. Then, when $X'_i \cap Q'_{2j} \neq \emptyset$, we can learn that $l_i > t_j$. Thus, the transformation is correct.

After the above transformation, Algorithm 1 can be transformed to be Algorithm 2.

Algorithm 2 Determine whether $Q \subseteq X$ or not

Input: $\{X'_i\}_{i=1}^{|X|}$, $\{Q'_{2j-1}, Q'_{2j}\}_{j=1}^{|Q|}$
Output: The matching result between X and Q

```
j = 1;
for i = 1 to |X| do
    // Case (a)
    if Q'_{2j-1} ⊆ X'_i then
        i = i + 1;
        j = j + 1;
    // Case (b)
    else if X'_i ∩ Q'_{2j} ≠ ∅ then // X cannot contain e_{t_j}, so Q ⊈ X
        return 0; // Q is not a subset of X
    // Case (c)
    else
        i = i + 1;
    if j > |Q| then
        return 1; // Q is a subset of X
return 0;
```

Example 4: Suppose that $X = \{e_1, e_2, e_5, e_6\}$ and $Q = \{e_1, e_3, e_5\}$, where the collection of elements is assumed to be $\mathcal{E} = \{e_1, e_2, \cdots, e_6\}$. Then, we can transform X to be 4 sets, i.e., $X'_1 = \{e_1\}$, $X'_2 = \{e_1, e_2\}$, $X'_3 = \{e_1, e_2, e_5\}$ and $X'_4 = \{e_1, e_2, e_5, e_6\}$. Meanwhile, we transform Q to be 6 (i.e., $2 * |Q|$) sets. That is, $Q'_1 = \{e_1\}$, $Q'_2 = \{e_2, e_3, e_4, e_5, e_6\}$, $Q'_3 = \{e_1, e_3\}$, $Q'_4 = \{e_4, e_5, e_6\}$, $Q'_5 = \{e_1, e_3, e_5\}$, $Q'_6 = \{e_6\}$. Then, following the Algorithm 1, we can determine the set containment relationship between X and Q. Meanwhile, we list the equivalent conditions of Algorithm 1 and Algorithm 2 in Eq. (10).

$$\begin{cases} (l_1 = 1; t_1 = 1)\ e_{l_1} = e_{t_1} & \Leftrightarrow\ Q'_1 \subseteq X'_1 \\ (l_2 = 2; t_2 = 3)\ l_2 < t_2 & \Leftrightarrow\ Q'_4 \cap X'_2 = \emptyset \\ (l_3 = 5; t_2 = 3)\ l_3 > t_2 & \Leftrightarrow\ Q'_4 \cap X'_3 \neq \emptyset \end{cases} \quad (10)$$

C. Description of Our Proposed Scheme

In this subsection, we present our set containment search scheme, which consists of three phases, i.e., system initialization, local data outsourcing and set containment search.

1) System Initialization: The data owner is responsible for initializing the system. Suppose that $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ is the dataset collected by the data owner, where $X_i \subseteq \mathcal{E} = \{e_1, e_2, \cdots, e_d\}$. The data owner generates an invertible matrix $\mathbf{M} \in \mathbb{R}^{(d+3)\times(d+3)}$ and the corresponding inverse matrix $\mathbf{M}^{-1} \in \mathbb{R}^{(d+3)\times(d+3)}$ as the secret keys. At the same time, for each user $\mathtt{U}_i$, the data owner generates two random matrices, i.e., $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{R}^{(d+3)\times(d+3)}$, such that $(\mathbf{M}_1^{-1})^T \mathbf{M}_2 = (\mathbf{M}^{-1})^T$. In addition, the data owner randomly generates an access key ak for the AES algorithm. Then, the data owner authorizes $\mathtt{U}_i$ by respectively distributing $(\mathbf{M}_1, ak)$ and $\mathbf{M}_2$ to $\mathtt{U}_i$ and the cloud server. Note that different query users are authorized by different matrices.

2) Local Data Outsourcing: The data owner outsources the collection of sets $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ to the cloud server as the following steps.

Step 1: Build a radix tree T to represent the sets in $\mathcal{X}$.

Step 2: Extend the radix tree T. According to the transformation method of X in Algorithm 2, in the extended tree, we replace the elements of each internal node with all elements from the current node to the root node. In this way, the radix tree in Fig. 4 can be extended to be that in Fig. 5.

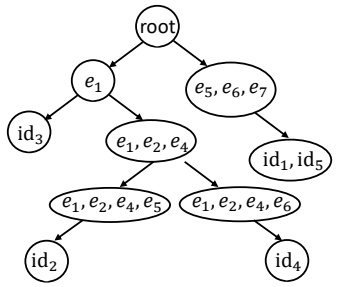


Fig. 5: Extended radix tree for $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5\}$

Step 3: Encrypt the extended radix tree T. In the extended tree T, each internal node is associated with a set, denoted by X_{node}. Then, the data owner encrypts this node by encrypting X_{node} as $\mathrm{CT}_{X_{node}} = \mathsf{ScieEnc}(\mathbf{M}, X_{node})$. For each leaf node, the data owner uses AES algorithm to encrypt the identities of the sets storing in it. Suppose that $\mathcal{ID}_{\mathrm{leafNode}}$ is the collection of the sets' identities storing in the leaf node. Then, the data owner encrypts the leaf node as $\mathrm{AES}_{ak}(\mathcal{ID}_{\mathrm{leafNode}})$.

Step 4: Outsource the encrypted tree T to the cloud server.

Step 5: On receiving the outsourced tree T, the cloud server stores it in the cloud.

3) Set Containment Search: Given a query set Q, the authorized query user $\mathtt{U}_i$ can enjoy the set containment search service to obtain the sets containing Q as the following steps.

Step 1: Suppose that $Q = \{e_{t_1}, e_{t_2}, \cdots, e_{t_{|Q|}}\}$ is the query set. The query user first constructs $2|Q|$ sets as Eq. (9), denoted by $\mathcal{Q} = \{Q_{2j-1}, Q_{2j}\}_{j=1}^{|Q|}$.

Step 2: The query user $\mathtt{U}_i$ generates the query token. In specific, for each $Q_{2j-1} \in \mathcal{Q}$, it uses its secret matrix $\mathbf{M}_1$ to generate a set containment query token $\mathrm{TKC}_{Q_{2j-1}} = \mathsf{ScieTokenC}(\mathbf{M}_1, Q_{2j-1})$. For each $Q_{2j} \in \mathcal{Q}$, it uses $\mathbf{M}_1$ to generate a set intersection query token $\mathrm{TKIS}_{Q_{2j}} = \mathsf{ScieTokenIS}(\mathbf{M}_1, Q_{2j})$, where $1 \le j \le |Q|$. Then, $\mathtt{U}_i$ randomly chooses a session key ssk. Finally, the query user launches

a query request and sends ssk together with the collection of tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{j=1}^{|Q|}$ to the cloud server, where the privacy of ssk can be preserved by a public key encryption scheme, e.g., RSA.

Step 3: On receiving the query request, the cloud server first validates whether the query user $\mathtt{U}_i$ is authorized or not. If not, reject the current search request. Otherwise, it runs the following steps to obtain the search result.

Step 4: Since the query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{j=1}^{|Q|}$ are encrypted by the matrix $\mathbf{M}_1$. In order to perform the set containment search, the cloud server first uses $\mathtt{U_i}$'s secret matrix $\mathbf{M_2}$ to update each query token as

$$\begin{aligned}\mathrm{TKC}_{Q_{2j-1}} &= \mathrm{TKC}_{Q_{2j-1}}\mathbf{M}_2 \\ &= \mathbf{q}'_{2j-1}(\mathbf{M}_1^{-1})^T\mathbf{M}_2 \\ &= \mathbf{q}'_{2j-1}(\mathbf{M}^{-1})^T\end{aligned}$$

$$\begin{aligned}\mathrm{TKIS}_{Q_{2j}} &= \mathrm{TKC}_{Q_{2j}}\mathbf{M}_2 \\ &= \mathbf{q}'_{2j}(\mathbf{M}_1^{-1})^T\mathbf{M}_2 \\ &= \mathbf{q}'_{2j}(\mathbf{M}^{-1})^T\end{aligned}$$

After the update, the query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{j=1}^{|Q|}$ become the tokens encrypted by the matrix $\mathbf{M}$. With the updated query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{j=1}^{|Q|}$, the cloud server performs a depth-first set containment search over the encrypted extended radix tree T to obtain the query result $\mathcal{R}$. The search algorithm is shown in Algorithm 3. Its main idea is to regard each leaf node as a set and the path from the leaf node to the root node stores its transformation sets. Then, the Algorithm 2 can be used to determine whether the set associated with the leaf node contains the query set or not. Since the cloud server only has the ciphertexts and tokens, the condition $Q'_{2j-1} \subseteq X_{node}$ of Algorithm 2 can be replaced by set containment query $\mathsf{ScieQueryC}(\mathrm{CT}_{X_{node}}, \mathrm{TKC}_{Q_{2j-1}}) = 1$. Meanwhile, the condition $X_{node} \cap Q'_{2j} \neq \emptyset$ of Algorithm 2 can be replaced by a set intersection query $\mathsf{ScieQueryIS}(\mathrm{CT}_{X_{node}}, \mathrm{TKIS}_{Q_{2j}}) = 1$.

In Algorithm 3, we use a stack S to store the nodes that should be searched later. For each node being searched, the search algorithm first matches the node with Q's query tokens, i.e., line 6 and line 7. The reason why we need to run the set containment query algorithm in line 6 multiple times is that each radix tree node may contain more than one element. In this case, the node may match multiple query tokens. After that, if the current node has matched all elements of Q, all its leaf nodes contain the query set Q and they will be added into the query result $\mathcal{R}$. If $\mathsf{ScieQueryIS}(\mathrm{CT}_{X_{node}}, \mathrm{TKIS}_{Q_{2j}}) = 1$, all its child nodes cannot contain Q and they will be pruned. Otherwise, add all its child nodes into the stack S and they will be searched later.

Step 5: The cloud server uses the session key ssk to encrypt the query result $\mathcal{R} = \{\mathrm{AES}_{ak}(\mathcal{ID}_{\mathrm{leafNode}})\}$ as $\mathcal{R}' = \{\mathrm{AES}_{ssk}(\mathrm{AES}_{ak}(\mathcal{ID}_{\mathrm{leafNode}}))\}$. Then, the cloud server returns $\mathcal{R}'$ to the query user $\mathtt{U}_i$.

Step 6: On receiving the query result $\mathcal{R}'$, $\mathtt{U}_i$ can use the session key ssk and access key ak to recover each $\mathcal{ID}_{\mathrm{leafNode}}$, where each $\mathrm{id}_j \in \mathcal{ID}_{\mathrm{leafNode}}$ contains the query set Q.

Algorithm 3 The set containment search

Input:
Query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{i=1}^{|Q|}$
The encrypted radix tree T
Output: The query result $\mathcal{R}$.

```
1: j = 1;
2: Stack S = ∅;
3: S.push(T.root, j);
4: while S ! = ∅ do
5:     (node, j) = S.pop();
6:     while ScieQueryC(CT_{X_node}, TKC_{Q_{2j-1}}) == 1 do
7:         j = j + 1;
8:     if j > |Q| then
9:         add all node's leaf nodes into the query result R;
10:    else if ScieQueryIS(CT_{X_node}, TKIS_{Q_{2j}}) == 1 then
11:        all node's child nodes can be pruned;
12:    else
13:        for each node's child node child do
14:            S.push(child, j);
15: return R;
```

D. Optimization of Query Token Generation Method

In our proposed scheme, when the query user launches a set containment search request, he/she first transforms the query set Q to be $2|Q|$ sets $\mathcal{Q} = \{Q_{2j-1}, Q_{2j}\}_{j=1}^{|Q|}$ and encrypts these sets to be query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{j=1}^{|Q|}$. However, this transformation method has some limitations. On the one hand, when the size of Q is large, the computational cost of query tokens generation and communication overhead of query request is large. On the other hand, from the query tokens, the cloud server can learn about the size of Q. In order to reduce the computational cost and communication overhead of the query user and preserve the privacy of the query set, the query user does not have to generate and send all of the query tokens $\mathcal{Q} = \{Q_{2j-1}, Q_{2j}\}_{j=1}^{|Q|}$ to the cloud server. In other words, he/she can randomly select several pairs of query tokens from $\{Q_{2j-1}, Q_{2j}\}_{j=1}^{|Q|-1}$ and send them together with $\{Q_{2|Q|-1}, Q_{2|Q|}\}$ to the cloud. It is easy to validate that the set containment search is still correct even if the query request only contains part of the query tokens. Since the optimized query token generation method is more secure, our scheme adopts this method to generate the query tokens by default.

V. Security Analysis

In this section, we analyze the security of our proposed set containment search scheme. Since the SCIE-Enc construction is the building block of our proposed scheme, we first analyze the security of our SCIE-Enc construction.

A. Security Analysis of Our SCIE-Enc Construction

In this subsection, we show that our scheme is selectively simulation-secure in the real and ideal experiments.

Theorem 3: Our SCIE-Enc construction is selectively simulation-secure with leakage $\mathcal{L} = \{\mathcal{L}_C, \mathcal{L}_{IS}\}$. $\mathcal{L}_C = \mathsf{ScieQueryC}(\mathrm{CT}_{X_i}, \mathrm{TKC}_Q)$ is the set containment leakage and $\mathcal{L}_{IS} = \mathsf{ScieQueryIS}(\mathrm{CT}_{X_i}, \mathrm{TKIS}_Q)$ is the set intersection leakage, where CT_{X_i}, TKC_Q and TKIS_Q are respectively the

ciphertext of X_i, the set containment query token, and the set intersection query token of Q.

Proof: First, we construct a simulator Sim for the ideal experiment as follows.

• *Setup:* In the setup phase, $\mathcal{A}$ chooses a plaintext set $X_i \in \mathcal{X}$ and sends it to the simulator Sim. Then, Sim randomly selects a $(d+3)$-dimensional vector as the ciphertext CT_{X_i}.

• *Set containment query phase 1:* $\mathcal{A}$ adaptively chooses p_1 query sets $\{Q_j\}_{j=1}^{p_1}$ and sends them to Sim. Then, for each Q_j, Sim takes $\mathcal{L}_C = \mathsf{ScieQueryC}(\mathrm{CT}_{X_i}, \mathrm{TKC}_{Q_j})$ as input.

- If $\mathcal{L}_C = 1$, Sim randomly generates a $(d+3)$-dimensional vector TKC_{Q_j} such that $\mathrm{CT}_{X_i} \circ \mathrm{TKC}_{Q_j} > 0$ as the query token.
- If $\mathcal{L}_C = 0$, Sim randomly generates a $(d+3)$-dimensional vector TKC_{Q_j} such that $\mathrm{CT}_{X_i} \circ \mathrm{TKC}_{Q_j} < 0$ as the query token.

Then, Sim returns $\{\mathrm{TKC}_{Q_j}\}_{j=1}^{p_1}$ to $\mathcal{A}$.

• *Set intersection query phase 1:* $\mathcal{A}$ adaptively chooses p_1' query sets $\{Q_j\}_{j=1}^{p_1'}$ and sends them to Sim. Then, for each Q_j, Sim takes the leakage $\mathcal{L}_{IS} = \mathsf{ScieQueryIS}(\mathrm{CT}_{X_i}, \mathrm{TKIS}_{Q_j})$ as input.

- If $\mathcal{L}_{IS} = 1$, Sim randomly generates a $(d+3)$-dimensional vector TKIS_{Q_j} such that $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_{Q_j} > 0$ as the query token.
- If $\mathcal{L}_{IS} = 0$, Sim randomly generates a $(d+3)$-dimensional vector TKIS_{Q_j} such that $\mathrm{CT}_{X_i} \circ \mathrm{TKIS}_{Q_j} < 0$ as the query token.

Then, Sim returns $\{\mathrm{TKIS}_{Q_j}\}_{j=1}^{p_1'}$ to $\mathcal{A}$.

• *Challenge phase:* Sim returns CT_{X_i} to $\mathcal{A}$.

• *Set containment query phase 2:* $\mathcal{A}$ runs the set containment query phase 1 again to obtain $p_2 - p_1$ query sets' set containment query tokens, i.e., $\{\mathrm{TKC}_{Q_j} = \mathsf{ScieTokenC}(sk, Q_j)\}_{j=p_1+1}^{p_2}$.

• *Set intersection query phase 2:* $\mathcal{A}$ runs the set intersection query phase 1 again to obtain $p_2' - p_1'$ query sets' set intersection query tokens, i.e., $\{\mathrm{TKIS}_{Q_j} = \mathsf{ScieTokenIS}(sk, Q_j)\}_{j=p_1'+1}^{p_2'}$.

In the ideal experiment, the view of $\mathcal{A}$ is $\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}} = \{X_i, \mathrm{CT}_{X_i}, \{Q_j, \mathrm{TKC}_{Q_j}\}_{j=1}^{p_2}, \{Q_j, \mathrm{TKIS}_{Q_j}\}_{j=1}^{p_2'}\}$ and all of them are random $(d+3)$-dimensional vectors. In the real experiment, the view of $\mathcal{A}$ is $\mathrm{View}_{\mathcal{A},\mathrm{Real}} = \{X_i, \mathrm{CT}_{X_i}, \{Q_j, \mathrm{TKC}_{Q_j}\}_{j=1}^{p_2}, \{Q_j, \mathrm{TKIS}_{Q_j}\}_{j=1}^{p_2'}\}$, where

$$\begin{cases} \mathrm{CT}_{X_i} = (r_{i,2} * \mathbf{x}_i, r_{i,2}, r_{i,1}, r'_{i,1})\mathbf{M} \\ \mathrm{TKC}_{Q_j} = (r_{j,4} * \mathbf{q}_j, -r_{j,4} * |\mathbf{q}_j|, r_{j,3}, r'_{j,3})(\mathbf{M}^{-1})^T \\ \mathrm{TKIS}_{Q_j} = (r_{j,6} * \mathbf{q}_j, 0, -r_{j,5}, -r'_{j,5})(\mathbf{M}^{-1})^T, \end{cases} \tag{11}$$

and $\{r_{i,1}, r'_{i,1}, r_{i,2}, r_{j,3}, r'_{j,3}, r_{j,4}, r_{j,5}, r'_{j,5}, r_{j,6}\}$ are positive random numbers.

In order to distinguish the real experiment and ideal experiment, the adversary $\mathcal{A}$ will attempt to distinguish $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ and $\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}}$. Since $\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}}$ only contains random vectors, $\mathcal{A}$ will attempt to distinguish $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ from random vectors. In Theorem 4, we prove that $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ is indistinguishable from random vectors.

Theorem 4: $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ is indistinguishable from random vectors, where $\mathrm{View}_{\mathcal{A},\mathrm{Real}} = \{X_i, \mathrm{CT}_{X_i}, \{Q_j, \mathrm{TKC}_{Q_j}\}_{j=1}^{p_2}, \{Q_j, \mathrm{TKIS}_{Q_j}\}_{j=1}^{p_2'}\}$.

Proof: We prove the correctness of Theorem 4 by contradiction. Suppose that $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ can be distinguished from random vectors. Then, $\mathcal{A}$ must be able to construct an equation system through vectors in $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ such that this equation system has a unique solution. Without loss of generality, we assume that there are n equations in this equation system. Then, $\mathcal{A}$ should try his/her best to minimize the number of unknown variables in each equation. It is easy to verify the following equation system has smaller or equal to number of unknown variables compared with other equation systems.

$$\begin{cases} \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TKC}_{\mathbf{q}_1} = & r_{i,2} * r_{1,4} * (\mathbf{x}_i \circ \mathbf{q}_1 - |\mathbf{q}_1|) \\ & + r_{i,1} * r_{1,3} + r'_{i,1} * r'_{1,3} \\ \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TKC}_{\mathbf{q}_2} = & r_{i,2} * r_{2,4} * (\mathbf{x}_i \circ \mathbf{q}_2 - |\mathbf{q}_2|) \\ & + r_{i,1} * r_{2,3} + r'_{i,1} * r'_{2,3} \\ & \cdots \\ \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TKC}_{\mathbf{q}_n} = & r_{i,2} * r_{n,4} * (\mathbf{x}_i \circ \mathbf{q}_n - |\mathbf{q}_n|) \\ & + r_{i,1} * r_{n,3} + r'_{i,1} * r'_{n,3}. \end{cases} \tag{12}$$

This is because each equation in Eq. (12) has removed unknown variables as much as possible, i.e., it removes unknown matrix $\mathbf{M}$ for $j = 1, 2, \cdots, p_2$. If the adversary replaces an equation in Eq. (12) with any other equation, the unknown variables of the equation system will be larger or equal to number of unknown variables in Eq. (12).

In Eq. (12), the values $\{\mathbf{x}_i \circ \mathbf{q}_j - |\mathbf{q}_j|\}_{j=1}^{n}$ can be computed with $\{X_i, \{Q_j\}_{j=1}^{n}\}$. Then, Eq. (12) totally has n equations with $3n+3$ unknown variables $\{r_{i,1}, r'_{i,1}, r_{i,2}, \{r_{j,3}, r'_{j,3}, r_{j,4}\}_{j=1}^{n}\}$. Since the number of unknown variables is larger than the number of equations, Eq. (12) has an infinite number of solutions, which contradicts to the assumption. Thus, the assumption is wrong, and Theorem 4 is correct. That is, $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ is indistinguishable from random vectors.

Thus, $\mathcal{A}$ cannot distinguish $\mathrm{View}_{\mathcal{A},\mathrm{Real}}$ from random vectors. Hence, $\mathcal{A}$ cannot distinguish the real and ideal experiments. Therefore, our SCIE-Enc construction is selectively simulation-secure with leakage $\mathcal{L}$.

B. Security of Our Proposed Scheme

In this subsection, we analyze the security of our proposed scheme, in which the query tokens are generated by the optimized query token generation method. As described in our security model, our security analysis mainly focuses on the privacy-preserving properties, so we will show that (i) the sets stored in the cloud are privacy-preserving; (ii) the query sets are privacy-preserving; (iii) the query results are privacy-preserving.

• *The sets stored in the cloud are privacy-preserving.* As described in Subsection IV-C, the sets are stored in an encrypted radix tree and outsourced to the cloud server. Since the cloud server is considered to be *honest-but-curious*, it may be curious about the plaintexts of sets. In this case, it may attempt to deduce some sets information from the encrypted radix tree. In the radix tree, each internal node is associated with a set, which is encrypted by the SCIE-Enc construction. Then, the adaptive security of our SCIE-Enc construction guarantees that the cloud server has no idea on the plaintext of set in the internal node. For the leaf node, it stores the ciphertext of sets' identities. The security of AES

algorithm guarantees that the cloud server has no idea on the plaintext sets' identities. Thus, the cloud server cannot obtain any information about the plaintext of sets from the encrypted radix tree.

At the same time, the cloud server may try to obtain some sets information from the radix tree structure. Specifically, in order to obtain the set information, the cloud server may try to analyze the number of elements in each tree path by counting the number of internal nodes of each tree path. However, in the radix tree, the internal node with one child node has been merged with its corresponding child node, so each internal node in the radix tree may contain more than one element. In other words, multiple set elements are possibly encrypted to be one internal node. In this case, the cloud server cannot obtain any information about the set elements information from the tree path. Therefore, the cloud server has no idea on the plaintexts of sets' identities storing in the cloud.

• *The query sets are privacy-preserving.* When the query user launches a set containment search, the plaintext of the query set should be kept secret from the cloud server and other query users.

For the cloud server, during a query, it can receive several pairs of query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}$. Since the query tokens have been encrypted with our SCIE-Enc construction, the adaptive security of SCIE-Enc construction can guarantee that the cloud server cannot obtain any information about the plaintext of Q. At the same time, in order to preserve the size of Q, the query user only sends several pairs of query tokens rather than $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}_{j=1}^{|Q|}$. Thus, the size of the query set is privacy-preserving. After receiving the query tokens, the cloud server needs to search the encrypted radix tree to find out the query result. The search process will leak which internal nodes satisfy the query tokens $\{\mathrm{TKC}_{Q_{2j-1}}, \mathrm{TKIS}_{Q_{2j}}\}$. When an internal node satisfies $\mathrm{TKC}_{Q_{2j-1}}$, it means that this node contains all of elements in Q_{2j-1}, i.e., $\{e_{t_1}, e_{t_2}, \cdots, e_{t_j}\}$. Meanwhile, when an internal node satisfies $\mathrm{TKIS}_{Q_{2j}}$, it means that this node contains at least one element whose global order is larger than j. This is because, it has intersection with $Q_{2j} = \{e_{t_j+1}, \cdots, e_d\}$. However, since both the internal node and the query sets may contain multiple elements, it is hard to guess the plaintext of the query set from the set containment and intersection relationship between the internal nodes and the query tokens. Thus, the cloud server has no idea on query sets' plaintexts.

For other query users, since the query tokens are generated with the users' secret matrix $\mathbf{M}_1$. Thus, the security of our SCIE-Enc construction guarantees that other query users also have no idea on the plaintext of the query sets. Therefore, the query sets are privacy-preserving.

• *The query results are privacy-preserving.* In our scheme, the query result is $\mathcal{R}' = \{\mathrm{AES}_{ssk}(\mathrm{AES}_{ak}(\mathcal{ID}_{\mathrm{leafNode}}))\}$, which are encrypted twice by the AES algorithm. For the cloud server, it only has access to ssk and has no idea on ak. For other query users, they have ak but they do not have access to the ssk. At the same time, the AES algorithm is secure, so none of the cloud server and other query users can learn about the plaintext of the query result. Thus, the query results are privacy-preserving.

VI. Performance Evaluation

In this section, we evaluate the performance of our scheme with respect to the computational cost of local data outsourcing, set containment search processing, query tokens generation and query result recovery. Since the function-hiding scalar product encryption schemes can also be deployed to achieve set containment search and the ASPE scheme is the most efficient one, we intend to compare our scheme with the ASPE scheme. However, the ASPE scheme cannot resist against the known-plaintext attacks. We compare our scheme with a modified ASPE scheme, where the modified ASPE scheme can resist against the known-plaintext attacks and does not incur too much additional computational cost compared with the original ASPE scheme.

In the modified ASPE scheme, we apply the idea of our SCIE-Enc construction to protect it from known-plaintext attacks. Specifically, a set X_i's binary vector $\mathbf{x}_i$ is encrypted as $\mathrm{CT}_{X_i} = \mathbf{x}_i'\mathbf{M} = (r_2 * \mathbf{x}_i, r_2, r_1, r_1')\mathbf{M}$. Meanwhile, a set Q's query token is generated by the set token generation algorithm, i.e., $\mathrm{TKC}_Q = \mathbf{q}'(\mathbf{M}^{-1})^T = (r_4 * \mathbf{q}, -r_4 * |\mathbf{q}|, r_3, r_3')(\mathbf{M}^{-1})^T$. When performing a set containment query, the cloud sever uses the query token TKC_Q to traverse all encrypted set records to obtain the query result. Based on our security analysis, the modified ASPE scheme is secure against know-plaintext attacks. Meanwhile, the modified ASPE scheme does not increase too much computational cost compared with the original ASPE scheme, because it only increases two additional dimensions.

In our experiments, both our scheme and the modified ASPE scheme were implemented by Java and the experiments were conducted on a machine with an Intel(R) Core(TM) i7-3770 CPU @3.40GHz, 16GB RAM and Windows 10 operating system. The access key and session keys for the AES algorithm are set to be 128 bits, i.e., $|ak| = |ssk| = 128$. Each experiment was conducted multiple times and the average runtime is reported. At the same time, the evaluated datasets are two real datasets, i.e., Jester [20] and MovieLen [21]. The Jeter dataset is a joke rating dataset and it contains the rating information of 100 jokes from 24,983 users. Each rating value ranges from -10.00 to 10.00. For each user, we transform his/his rating record to be a set, which contains the jokes that his/her rating value is larger than 7. After that, we obtain a dataset with 11,986 sets. For the MovieLen dataset, it contains the rating information of 3,952 movies from 6,040 users. Thus, it contains 6,040 sets in total. In order to better compare our proposed scheme with the modified ASPE scheme, we extend both the Jeter dataset and MovieLens dataset to contain 30,000 sets by generating some new sets for them. Note that the new generated sets are very similar to those of the original datasets, i.e., they follow the same distribution with the original datasets.

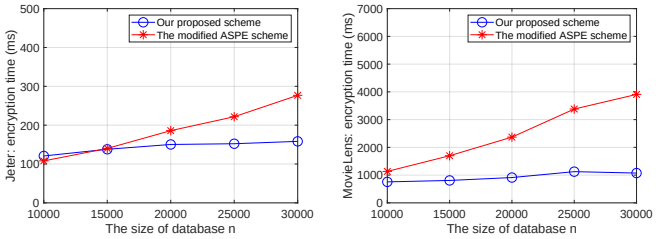
A. Computational Cost of Local Data Outsourcing

In our proposed scheme, the way of local data outsourcing is to represent the collection of sets $\mathcal{X}$ to be a radix tree, encrypt the radix tree, and outsource it to the cloud. The computational cost is mainly derived from encrypting the radix tree.

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/TSC.2021.3065240, IEEE Transactions on Services Computing

Encrypting an internal nodes requires $O(d^2)$ computational cost and encrypting a leaf node requires an AES encryption operation. Meanwhile, the number of internal nodes and leaf nodes in the radix tree heavily depends on the datasets, because different datasets have different radix tree structures. Then, the computational cost of encrypting the radix tree depends on the datasets. For the modified ASPE scheme, each set and its identity are respectively encrypted with our SCIE-Enc construction and AES algorithm. Thus, encrypting a dataset with n set records requires $O(nd^2)$ computational complexity to encrypt the sets and n AES operations to encrypt their identities.

Fig. 6(a) and Fig. 6(b) respectively plot the computational cost of local data outsourcing varying with the size of the dataset n on the Jeter dataset and MovieLens dataset. From these figures, we can see that the computational cost of our scheme and the modified ASPE scheme increases with n. However, the increasing rate of the modified ASPE scheme is far more larger than that of our scheme. Thus, the computational cost of the modified ASPE scheme is higher than that of our scheme. For example, when $n = 25,000$, the computational cost in our scheme on the MovieLens dataset is about 1,000 ms, while that in the modified ASPE scheme is about 3,200 ms, i.e., more than triple times higher than that in our scheme.



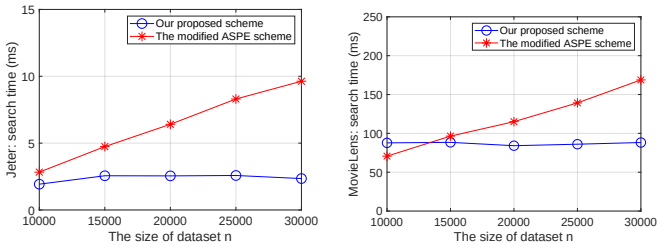
(a) Jeter dataset (b) MovieLens dataset

Fig. 6: The computational cost of local data outsourcing versus the size of dataset n

B. Computational Cost of Set Containment Search

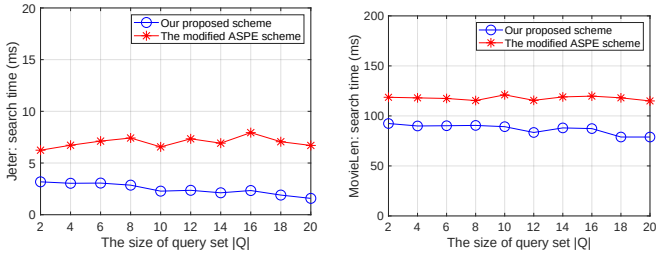
In this subsection, we evaluate the computational cost of set containment search in the cloud. In our scheme, the computation cost of query processing is related to the size of dataset, i.e., n, the number of elements in the database, i.e., d, and the size of the query set, i.e., $|Q|$. For the modified ASPE scheme, the set containment query is processed by traversing all set records, so its computational cost is related to n and d.

• *The search time varies with n:* Fig. 7(a) and Fig. 7(b) plot the computational cost of set containment search varying with n on Jeter and MovieLens datasets when the size of the query set is 10. From these figures, we can see that the computational cost in the modified ASPE scheme linearly increases with n while that in our scheme is very steady and irrelevant to n. This is because the search time in our scheme heavily depends on the tree structure. If the distribution of the set records is uniform, the increase of set records may not change the tree structure too much. In this case, the set containment search time over the tree will be steady and is not affected by the size of the dataset. However, in the modified ASPE scheme, each set record is encrypted separately, so it will be searched one by one. Then, the computational cost will be linear to the size of dataset. At the same time, we can see that the set containment search in our scheme is much more efficient than that in the modified ASPE scheme. For instance, when $n = 25,000$, the search time of our scheme on MovieLens dataset is about 80 ms, while that of the modified ASPE scheme is around 140 ms.



(a) Jeter dataset (b) MovieLens dataset

Fig. 7: The computational cost of set containment search varying with n when $|Q| = 10$



(a) Jeter dataset (b) MovieLens dataset

Fig. 8: The computational cost of set containment search varying with $|Q|$ when $n = 20,000$

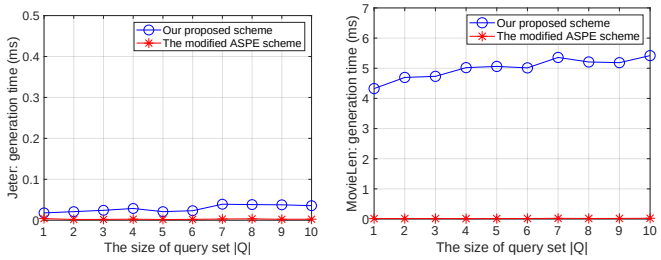
• *The search time varies with $|Q|$:* Fig. 8(a) and Fig. 8(b) plot the computational cost of set containment search varying with $|Q|$ on Jeter and MovieLens, when n is set to be 20,000. From these two figures, we can see that the computational cost of our scheme decreases with the increase of $|Q|$, while that of the modified ASPE scheme is not affected by the increase of $|Q|$ too much. This is because the search process of the modified ASPE scheme is to traverse all the set records and is irrelevant to $|Q|$. For our scheme, the search is processed by traversing the tree structure and it involves a pruning strategy. Then, when $|Q|$ increases, more tree paths are likely to be pruned, so the computational cost can be reduced correspondingly. In addition, the search efficiency of our scheme is more efficient than that of the modified ASPE scheme. For example, when $|Q| = 18$, the computational cost of our scheme in MovieLens is about 80 ms, while that of the ASPE-based scheme is around 120 ms.

C. Computational Cost of Query Token Generation

In this subsection, we evaluate the computational cost of query token generation at the query user side. For the ASPE-

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/TSC.2021.3065240, IEEE Transactions on Services Computing

based scheme, the query user just needs to generate one query token and the computational cost is very low. For our proposed scheme, the computational cost of the query token generation is related to the size of query set, i.e., $|Q|$, and the number of query tokens are randomly chosen, so we mainly evaluate the computational cost of query token generation varying with $|Q|$. As shown in Fig. 9(a) and Fig. 9(b), the computational cost of query token generation in our scheme increases with $|Q|$. Although the query token generation time of our scheme is higher than that in the ASPE-based scheme, it is still very low.



(a) Jeter dataset (b) MovieLens dataset

Fig. 9: The computational cost of query token generation varying with $|Q|$

D. Computational Cost of Query Result Recovery

In both our scheme and the ASPE-based scheme, the query user recovers the query result by AES decryption operations. Meanwhile, the computational cost is related to the number of returned AES ciphertexts. For each encrypted ciphertext, recovering the result requires two decryption operations, and one AES encryption/decryption operation in our experiment is about 9.9 μs. Thus, the query result recovering is very efficient in both our scheme and the ASPE-based scheme.

VII. Related Work

Set containment search is to find the containment relationship between the query set and the sets in the database, and it has attracted considerable attention from the academia and industry. At the same time, it has been extensively studied in [7]–[10]. However, most of the proposed schemes focus on improving the efficiency of the set containment search over plaintext domain, and do not take the data confidentiality into consideration. Thus, they are not applicable to our scenario.

As discussed in Section I, the set containment search problem can be transformed into the scalar product based search problem when the set records are represented to be fix-dimensional binary vectors. In the literature, many studies focus on achieving the privacy-preserving scalar product computation [11]–[15], [22]–[25]. Among these studies, the function-hiding scalar product encryption schemes [11]–[15] can be applied to achieve scalar product based search over outsourced data in the cloud environment. This is because that these schemes can compute the scalar product between the query vector and the vectors in the database over their ciphertexts. With this property, given a query vector, these schemes can filter out the vectors whose scalar product with the query vector satisfies the scalar product search criteria.

Bishop et al. [11] first used asymmetric bilinear maps to design a function-hiding scalar product scheme and the security of the proposed scheme is based on the symmetric external Diffie-Hellman assumption. Based on the work in [11], Datta et al. proposed a new scheme, which can improve the security of the scheme in [12] by removing the constraint of adversaries' queries. Later, Kim et al. [13] and Zhang et al. [14] respectively proposed another two function-hiding scalar product schemes. However, all of the above schemes are based on public key cryptographic techniques, where the schemes in [11]–[13] deployed the bilinear pairing maps and the scheme in [14] employed a Paillier variant homomorphic encryption scheme [16]. Thus, the computational cost of such schemes is heavy.

Wong et al. [15] designed an asymmetric scalar-product-preserving encryption (ASPE) scheme, which can also achieve the scalar product based search over outsourced data. In this scheme, the vectors in database and the query vectors are encrypted by a real domain matrix. The matrix encryption is efficient because it is a symmetric encryption and only involves matrix multiplication of real domain. Thus, compared with public key cryptographic techniques, the matrix encryption scheme is the most efficient candidate to achieve set containment search. Nevertheless, on the one hand, the ASPE scheme cannot resist against the known-plaintext attacks [17]. On the other hand, deploying the matrix encryption scheme [15] to achieve set containment search still has a disadvantage in query efficiency. In specific, when applying the scheme in [15] to achieve set containment search, each set record is encrypted into a ciphertext and outsourced to the cloud. Given a query set, the cloud server needs to traverse all of the set records in the database to find out the query result. Thus, the query efficiency is linear to the size of the database. When the database is large, it is inefficient to deploy the scheme in [15] to achieve the set containment search. Although the schemes [14], [26] introduced two pruning strategies to accelerate the ASPE-based similarity query processing, the introduced pruning strategies were designed for the scalar product based top-k similarity queries and conjunction queries. Thus, these pruning strategies are not applicable to our set containment search. Therefore, it is still challenging to achieve efficient and privacy-preserving set containment search.

VIII. Conclusion

In this paper, we have proposed an efficient and privacy-preserving set containment search scheme. In specific, we employed an asymmetric scalar-product-preserving encryption technique to design an SCIE-Enc construction, which can achieve both the set containment query and set intersection query. Then, we built a radix tree to represent the set records. Finally, we proposed an efficient and privacy-preserving set containment search scheme by applying our SCIE-Enc construction to encrypt and search the radix tree. The adoption of set intersection query enables some tree paths of the radix tree can be pruned when performing the set containment search

over outsourced set records in the cloud. Thus, our proposed scheme is much more efficient in set containment query compared with existing solutions. In addition, the security analysis and performance evaluation show that our proposed scheme is indeed privacy-preserving and efficient. In our future work, we will focus on improving the efficiency and security of the set containment search. In addition, based on the set containment search, we will design efficient set containment join computation schemes.

ACKNOWLEDGEMENTS

This research was supported in part by NSERC Discovery Grants (04009), LMCRF-S-2020-03, National Key Research and Development Program of China (2017YFB0802200), ZJNSF (LZ18F020003), NSFC (U1709217, 61972304), and Natural Science Foundation of Shaanxi Province (2019ZDLGY12-02).

REFERENCES

[1] R. Lu, "A new communication-efficient privacy-preserving range query scheme in fog-enhanced iot," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2497–2505, 2019.

[2] C. Gao, Q. Cheng, X. Li, and S. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," *Cluster Computing*, vol. 22, no. Suppl 1, pp. 1655–1663, 2019.

[3] S. S. Gill, S. Tuli, M. Xu, I. Singh, K. V. Singh, D. Lindsay, S. Tuli, D. Smirnova, M. Singh, U. Jain, H. Pervaiz, B. Sehgal, S. S. Kaila, S. Misra, M. S. Aslanpour, H. Mehta, V. Stankovski, and P. Garraghan, "Transformative effects of iot, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges," *Internet Things*, vol. 8, 2019.

[4] "Big Data Statistics 2020," https://techjury.net/stats-about/big-data-statistics/#gref.

[5] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, and K. R. Choo, "Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data," *Inf. Sci.*, vol. 498, pp. 91–105, 2019.

[6] Y. Zheng, R. Lu, X. Yang, and J. Shao, "Achieving efficient and privacy-preserving top-k query over vertically distributed data sources," in *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019*, 2019, pp. 1–6.

[7] R. Jampani and V. Pudi, "Using prefix-trees for efficiently computing set joins," in *Database Systems for Advanced Applications, 10th International Conference, DASFAA 2005, Beijing, China, April 17-20, 2005, Proceedings*, 2005, pp. 761–772.

[8] Y. Luo, G. H. L. Fletcher, J. Hidders, and P. D. Bra, "Efficient and scalable trie-based algorithms for computing set containment relations," in *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, 2015, pp. 303–314.

[9] J. Yang, W. Zhang, S. Yang, Y. Zhang, X. Lin, and L. Yuan, "Efficient set containment join," *VLDB J.*, vol. 27, no. 4, pp. 471–495, 2018.

[10] D. Deng, C. Yang, S. Shang, F. Zhu, L. Liu, and L. Shao, "Lcjoin: Set containment join via list crosscutting," in *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, 2019, pp. 362–373.

[11] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, 2015, pp. 470–491.

[12] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, 2016, pp. 164–195.

[13] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, 2018, pp. 544–562.

[14] Z. Zhang, K. Wang, C. Lin, and W. Lin, "Secure top-k inner product retrieval," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, 2018, pp. 77–86.

[15] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, 2009, pp. 139–152.

[16] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, 2003, pp. 37–54.

[17] W. Lin, K. Wang, Z. Zhang, and H. Chen, "Revisiting security risks of asymmetric scalar product preserving encryption and its variants," in *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*, 2017, pp. 1116–1125.

[18] M. Böhm, B. Schlegel, P. B. Volk, U. Fischer, D. Habich, and W. Lehner, "Efficient in-memory indexing with generalized prefix trees," in *Datenbanksysteme für Business, Technologie und Web (BTW), 14. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 2.-4.3.2011 in Kaiserslautern, Germany*, 2011, pp. 227–246.

[19] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, 2018, pp. 745–762.

[20] K. Y. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, 2001.

[21] "MovieLens dataset," https://grouplens.org/datasets/movielens/, 2003.

[22] G. Sheng, T. Wen, Q. Guo, and Y. Yin, "Privacy preserving inner product of vectors in cloud computing," *IJDSN*, vol. 10, 2014.

[23] L. Wang, T. Hayashi, Y. Aono, and L. T. Phong, "A generic yet efficient method for secure inner product," in *Network and System Security - 11th International Conference, NSS 2017, Helsinki, Finland, August 21-23, 2017, Proceedings*, 2017, pp. 217–232.

[24] F. Benhamouda, F. Bourse, and H. Lipmaa, "Cca-secure inner-product functional encryption from projective hash functions," in *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, 2017, pp. 36–66.

[25] O. Stan, R. Sirdey, C. Gouy-Pailler, P. Blanchart, A. B. Hamida, and M. Zayani, "Privacy-preserving tax calculations in smart cities by means of inner-product functional encryption," in *2nd Cyber Security in Networking Conference, CSNet 2018, Paris, France, October 24-26, 2018*, 2018, pp. 1–8.

[26] W. Lin, K. Wang, Z. Zhang, A. W.-C. Fu, R. C.-W. Wong, C. Long, and C. Miao, "Towards secure and efficient equality conjunction search over outsourced databases," *IEEE Transactions on Cloud Computing*, 2020.

Yandong Zheng received her M.S. degree from the Department of Computer Science, Beihang University, China, in 2017 and she is currently pursuing her Ph.D. degree in the Faculty of Computer Science, University of New Brunswick, Canada. Her research interest includes cloud computing security, big data privacy and applied privacy.

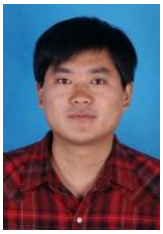
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2021.3065240, IEEE Transactions on Services Computing

Rongxing Lu (S'09-M'11-SM'15-F'21) is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. He is a Fellow of IEEE. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Yunguo Guan is a PhD student of the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Jun Shao received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008.

He was a Post-Doctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.



Hui Zhu (M'13-SM'19) received the B.Sc. degree from Xidian University, Xi'an, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, China, in 2005, and the Ph.D. degree from Xidian University, in 2009.

He was a Research Fellow with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2013. Since 2016, he has been a Professor with the School of Cyber Engineering, Xidian University. His current research interests include applied cryptography, data security, and privacy.