Achieving Efficient and Privacy-Preserving Exact Set Similarity Search over Encrypted Data

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

***Abstract*—Set similarity search, aiming to search the similar sets to a query set, has wide application in today's recommendation services. Meanwhile, the rapid advance in cloud technique has promoted the boom of data outsourcing. However, since the cloud is not fully trustable and the data may be sensitive, data should be encrypted before outsourced to the cloud. Undoubtedly, data encryption will hinder some basic functionalities, e.g., set similarity search. For achieving set similarity search over encrypted data, many solutions were proposed, yet they either only satisfy weak security requirements, or only achieve approximate similarity, or have low efficiency or under the model of two cloud servers. Therefore, in this paper, we propose a new efficient and privacy-preserving exact set similarity search scheme under a single cloud server. Specifically, we first design a symmetric-key predicate encryption (SPE-Sim) scheme, which can support similarity search over binary vectors. Then, we represent the set records to be binary vectors and employ the B+ tree to build an index for them. After that, based on SPE-Sim and the B+ tree based index, we propose our scheme and it can achieve efficient set similarity search while preserving the privacy of set records and query contents. Finally, security analysis and performance evaluation indicate that our scheme is privacy-preserving and efficient.**

***Index Terms*—Privacy-preserving, Exact set similarity search, Jaccard similarity, Predicate encryption, B+ tree.**

I. Introduction

The widespread application of information and communication technology has continuously promoted the explosive growth of data in various fields, such as smart home [1], eHealthcare [2], vehicular networks [3]. As predicted in [4], the global big data market will grow from $18.3 bn in 2014 to an incredible $92.2 bn by 2026. The boom of the data and the advance of cloud technique motivate an increasing number of individuals and organizations to outsource their data to the powerful cloud. However, since the data may contain some sensitive information, and the cloud storage data breaches happened from time to time due to attacks, malfunctions or misconfigurations, such as Dropbox password leak [5], it is not reliable to pin the cloud to protect the data confidentiality. An alternative solution to protect the data confidentiality in cloud is that the data owners deploy the encryption technique to encrypt the data before outsourcing them to the cloud [2], [6]. Nevertheless, the data encryption inevitably affects some basic functionalities on the data. Among these functionalities, the set similarity search is one of the most popular searches, which aims to search the similar sets to a given query set, and serves as the basis of today's collaborative filtering and recommendation services [7]–[10]. For example, the video and music service providers like YouTube, Netflix and Spotify can recommend some videos and songs to a specific user based on preferences of users similar to him/her.

In the literature, many studies have been dedicated to solve the similarity search over encrypted data and several privacy-preserving similarity search schemes have been proposed. Those schemes can be generally categorized into three types, i.e., (i) homomorphic encryption based schemes [11], [12]; (ii) order preserving encryption based schemes [13], [14]; and (iii) symmetric searchable encryption (SSE) and locality-sensitive hashing (LSH) based schemes [15]–[17]. However, most of them cannot well balance the query efficiency, security and accuracy. Specifically, the homomorphic encryption based schemes [11], [12] can achieve strong security, but they are computationally inefficient due to the time-consuming homomorphic encryption. The order preserving encryption based schemes [13], [14] are efficient in terms of similarity query, but the security of the two schemes are weak, because the employed order preserving encryption will leak the order information of the plaintext data.

For the SSE and LSH based schemes [15]–[17], they can just provide approximate similarity query rather than the exact one because the idea of LSH is to map similar data records to the same hash value. The "exact similarity search" means that the query results only contains the data points satisfying the search criteria. Nevertheless, in the SSE and LSH based schemes [15]–[17], the query results may contain false positive points. To achieve exact SSE and LSH based similarity search schemes, Zheng et al. [18] and Cui et al. [19] introduced Yao's garbled circuits [20] to filter the query result returned by the SSE and LSH based similarity search schemes. Although the similarity search schemes [18], [19] can return the exact query result, they were built under the model of two cloud servers. However, in order to reduce the cloud bills, the data owners may prefer to deploy a single cloud server in some real scenarios. Therefore, how to achieve exact efficient and privacy-preserving similarity search under the model of a single cloud server is still challenging.

Aiming at the above challenges, in this paper, we introduce a symmetric-key predicate encryption technique and B+ tree to propose an efficient and privacy-preserving exact set sim-

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).

ilarity search (SetSim) scheme under a single cloud server. Specifically, our contributions are three folds:

• First, we design a symmetric-key predicate encryption scheme supporting similarity search over binary vectors (SPE-Sim). In the SPE-Sim scheme, the data records are d-dimensional binary vectors, and are encrypted to be ciphertexts by an asymmetric scalar-product-preserving encryption (ASPE) technique. When performing the similarity search, a query token corresponding to the query vector and similarity threshold is generated, and it can be used to access the data records satisfying the query condition. Meanwhile, the Jaccard similarity, specially designed for measuring the similarity between two sets, is considered to be the similarity metric in the SPE-Sim scheme (defined in Subsection III-A).

• Second, we represent the set records to be binary vectors and employ the B+ tree to build an index for them. Then, based on the SPE-Sim scheme and B+ tree based index, we propose an efficient and privacy-preserving exact set similarity search scheme under a single cloud server. Our proposed scheme can achieve high efficiency in set similarity search while preserving the privacy of the set records, the query set as well as the similarity threshold.

• Third, we extensively analyze the security of our proposed scheme and also evaluate its performance. The results show that our proposed scheme is indeed privacy-preserving and efficient.

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 describe the related work in Section VII and draw our conclusion in Section VIII.

II. Models and Design Goals

In this section, we formalize the system model, security model, and identify design goals considered in this paper.

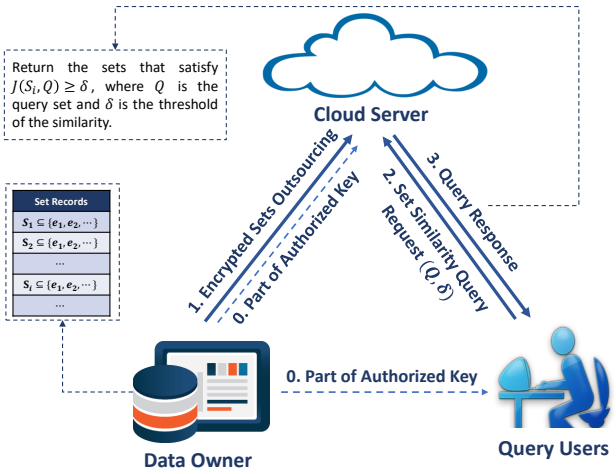

Fig. 1: System model under consideration

A. System Model

In our system model, we consider a cloud-based set similarity search model, which is comprised of three types of entities, namely a data owner, a cloud server and a set of query users, as shown in Fig. 1.

• *Data Owner:* The data owner has a collection of sets, denoted by $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$, and each $S_i \in \mathcal{S}$ is a subset of $\mathcal{E}$, i.e., $S_i \subseteq \mathcal{E}$, where $\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$ is the collection of all elements. Since the data owner has limited storage space and computing capability, he/she prefers to resort to a powerful cloud server to store and manage his/her data. However, as the cloud server is not always fully trustable and the data may contain some sensitive information, in order to preserve the privacy of the data, the data owner tends to encrypt the data before outsourcing them to the cloud.

• *Cloud Server:* The cloud server has abundant storage space and powerful computing capability, which is deployed as a link between the data owner and the query users. On the one hand, it provides the storage service to the data owner, i.e., store the outsourced data for the data owner. On the other hand, it also offers the set similarity query service to the query users. In specific, upon receiving a query set $Q \subseteq \mathcal{E}$ and a similarity threshold δ from a query user, the cloud server can search the outsourced sets and return a collection of sets whose similarity with Q is equal to or greater than δ, i.e., $\{S_i \in \mathcal{S} | J(Q, S_i) \geq \delta\}$.

• *Query Users $\mathcal{U} = \{U_1, U_2, \cdots\}$:* The query users can enjoy the set similarity search service from the cloud server, i.e., they can search the sets whose similarity with a given set Q is equal to or greater than a threshold δ. At the same time, the query users must be authorized by the data owner before they are eligible to enjoy the set similarity search service. In other words, only the authorized users can receive the returned sets from the cloud server after sending a set similarity search request. The data owner can authorize a query user by respectively distributing two parts of the authorization key to the cloud server and the query user as shown in Fig. 1.

B. Security Model

In our security model, the data owner is considered to be *honest* and he/she will sincerely follow the protocol to encrypt the sets and outsource them to the cloud server. However, the cloud server is considered to be *honest-but-curious*, i.e., it will honestly follow the protocol but may be curious about some private information. In specific, it will honestly store the outsourced data for the data owner and offer the set similarity query service to the query users. Nevertheless, it may be curious about the plaintext of the encrypted sets stored in the cloud. At the same time, when processing the set similarity search requests from the query users, it may be also curious about the plaintext of the query set Q, the threshold δ, as well as the query results (the sets whose similarity to Q is equal to or greater than δ). For the query users, the authorized users are considered to be *honest*, while the unauthorized users are likely to launch malicious passive attacks, e.g., eavesdropping, when they are curious about some private information or expected to enjoy the set similarity search service from the cloud server.

In addition, we consider there is no collusion between the cloud server and the query users. Note that there may be other passive or active attacks (e.g. denial of service and data pollution attacks), which are beyond the scope of this work and will be exploited in our future work.

C. Design Goals

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

• *Privacy Preservation:* The fundamental requirement of the proposed scheme is the privacy preservation, i.e., the encrypted sets stored in the cloud, the query set Q and the similarity threshold δ from the authorized query users, as well as the query result should be kept secret from unauthorized entities, including the cloud server and unauthorized users.

• *Efficiency:* In order to achieve the above privacy requirement, additional computational cost will be inevitably incurred, i.e., processing the set similarity search request 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 minimize the computational cost of processing the set similarity search request.

III. Preliminaries

In this section, we first formally define the problem of exact set similarity search and briefly review the ASPE technique, which will be used in our proposed scheme. After that, we present the overview of our SPE-Sim scheme and formally define its security. In order to facilitate the narrative, we first list some used notations in Table I.

A. Definition of Exact Set Similarity Search

Suppose that $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$ is a collection of sets and each set $S_i \in \mathcal{S}$ is a subset of $\mathcal{E}$, i.e., $S_i \subseteq \mathcal{E}$, where $\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$ is the collection of all elements. The exact set similarity search can be defined as follows.

Definition 1 (Exact set similarity search): Given a query set Q, the exact set similarity search allows the user to search the collection of sets $\mathcal{S}$ and find out the sets that are similar to the query set Q. The similarity criterion can be defined in two ways [8].

- Top-k set similarity search: it returns the top k sets that are the most similar to the query set;
- Set similarity range search: given a similarity threshold δ, it returns the sets whose similarity to the query set is equal to or greater than δ.

Meanwhile, the similarity between two sets can be measured in various metrics, such as Jaccard similarity, Inner product similarity, Euclidean distance, etc. [8]. Due to its popularity for measuring the set similarity, Jaccard similarity is deployed as the similarity metric in our scheme. For different types of data records, the Jaccard similarity can be defined in different ways. In the following, we respectively introduce the Jaccard similarity definitions over set records and binary vectors.

- For the set records S_i and S_j, the Jaccard similarity between them can be defined as

$$J(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{1}$$

- For two d-dimensional binary vectors $\mathbf{b_i}$ and $\mathbf{b_j}$, the Jaccard similarity between them can be defined as

$$J(\mathbf{b_i}, \mathbf{b_j}) = \frac{\mathbf{b_i} \circ \mathbf{b_j}}{|\mathbf{b_i}| + |\mathbf{b_j}| - \mathbf{b_i} \circ \mathbf{b_j}} \tag{2}$$

where "$\circ$" denotes the inner product operation, $|\mathbf{b_i}| = \mathbf{b_i} \circ \mathbf{b_i}$, and $|\mathbf{b_j}| = \mathbf{b_j} \circ \mathbf{b_j}$.

Note that each $S_i \in \mathcal{S}$ can be transformed to be a binary vector $\mathbf{b}_i = (x_1, x_2, \cdots, x_d)$ by setting

$$x_l = \begin{cases} 1 & e_l \in S_i \\ 0 & e_l \notin S_i \end{cases}$$

for $l = 1, 2, \cdots, d$. In this case, the Jaccard similarity over set records can be transformed to be that over binary vectors. That is, $J(S_i, S_j) = J(\mathbf{b}_i, \mathbf{b}_j)$, where $\mathbf{b}_i$ and $\mathbf{b}_j$ are respectively the binary vectors representation of S_i and S_j. This is also the key point of Jaccard similarity computation in our scheme, i.e., transform set records to binary vectors, and conduct similarity computation over the transformed binary vectors.

In this paper, we consider the set similarity range search as the similarity criterion and employ Jaccard similarity as the similarity metric. In this case, the set similarity range search can be formally defined in Definition 2 and an example is present in Example 1.

Definition 2 (Set similarity range search [8]): Given a query set Q and a similarity threshold δ, the set similarity range search is to identify the sets from $\mathcal{S}$ such that whose similarity to Q is equal to or greater than δ, i.e., the query result will be $\{S_i \in \mathcal{S} | J(S_i, Q) \geq \delta\}$.

TABLE II: A collection of sets

ID	Set
S_1	$\{e_1, e_2, e_3, e_4, e_6, e_7\}$
S_2	$\{e_1, e_3, e_5, e_6\}$
S_3	$\{e_1, e_2, e_4, e_5, e_6, e_7\}$
S_4	$\{e_1, e_3, e_4, e_5\}$
S_5	$\{e_1, e_2, e_3, e_5, e_7\}$

Example 1: Table II presents a collection of sets $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5\}$. If the query set is $Q = \{e_1, e_2, e_4, e_6, e_7\}$ and the similarity threshold is $\delta = 0.8$, the query result will be $\{S_1, S_3\}$, because only sets S_1 and S_3 satisfy $J(S_1, Q) = \frac{5}{6} = 0.833 \geq \delta$ and $J(S_3, Q) = \frac{5}{6} = 0.833 \geq \delta$.

B. ASPE Technique

The ASPE technique (described in [21]) was proposed for the secure kNN query computation over encrypted database, i.e., query the k nearest vectors in the database with a given query vector. The main idea of ASPE technique is to encrypt the data vectors in the database and generate the query token

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/TDSC.2020.3004442, IEEE Transactions on Dependable and Secure Computing

TABLE I: Summary of Notations

Symbol	Description
$\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$	The collection of all elements
$\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$	The collection of all sets, $S_i \subseteq \mathcal{E}$
$Q \subseteq \mathcal{E}$	The query set
$\mathbf{b}_i = (x_1, x_2, \cdots, x_d)$	The binary vector representation of S_i
$J(\cdot, \cdot)$	The Jaccard similarity
M	The secret key, which is an invertible matrix
M_1 and M_2	The authorized keys satisfying $(\mathrm{M}_1^{-1})^T\mathrm{M}_2 = (\mathrm{M}^{-1})^T$
$\mathrm{CT}_{\mathbf{x}}$	The ciphertext of $\mathbf{x}$
$\mathbf{q} = (v_1, v_2, \cdots, v_d)$ and δ	The query vector and similarity threshold
$\mathrm{TK}_{\mathbf{q}}$	The query token of $\mathbf{q}$
$\mathrm{TK}_{\mathbf{q},\delta}$	The query token of $(\mathbf{q}, \delta)$
$\mathbf{x}_{d_1} \in \{0,1\}^{d_1}$	A d_1-dimensional binary vector
$\mathbf{x}_{d_2} = \{*\}^{d_2}$	A d_2-dimensional vector and each element is either 0 or 1
$\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2})$, where $d_1 + d_2 = d$	A d-dimensional vector with prefix $\mathbf{x}_{d_1}$
$\widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}} \mid \widetilde{\mathbf{x}} \in \{0,1\}^d, \widetilde{\mathbf{x}}_{d_1} = \mathbf{x}_{d_1}\}$	The collection of vectors sharing the same prefix with $\mathbf{x}$
$\mathcal{W} = \{\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) \mid \mathbf{x}_{d_1} \in \{0,1\}^{d_1}, \mathbf{x}_{d_2} = \{*\}^{d_2}\}$	The class of plaintexts
$f_{\delta,\mathbf{q}}$	The predicate function, $f_{\delta,\mathbf{q}} = 1$ if $J(\mathbf{x}, \mathbf{q}) \geq \delta$. Otherwise, $f_{\delta,\mathbf{q}} = 0$
$\mathcal{F} = \{f_{\delta,\mathbf{q}} \mid \mathbf{q} \in \{0,1\}^d \setminus \mathbf{0}^d,\ 0 \leq \delta \leq 1\}$	The class of the predicates functions
$(K_1, K_2, \cdots, K_t)$	The key values of a B+ tree node
$(P_1, P_2, \cdots, P_{t+1})$	The children pointers of a B+ tree node
p_1 and p_2	Two polynomial numbers

for the query vector in different ways. Then, the order of the scalar products between the query vector and the vectors in the database can be preserved. In other words, for any two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in the database, if $\mathbf{x}_1 \circ \mathbf{q} \geq \mathbf{x}_2 \circ \mathbf{q}$, their encrypted data will also satisfy that $\mathrm{CT}_{\mathbf{x}_1} \circ \mathrm{TK}_{\mathbf{q}} \geq \mathrm{CT}_{\mathbf{x}_2} \circ \mathrm{TK}_{\mathbf{q}}$, where $\mathrm{CT}_{\mathbf{x}_1}$, $\mathrm{CT}_{\mathbf{x}_2}$ and $\mathrm{TK}_{\mathbf{q}}$ are respectively the ciphertexts of $\mathbf{x}_1$, $\mathbf{x}_2$, and the query token of $\mathbf{q}$. Specifically, the ASPE technique $\Pi_{\mathsf{ASPE}} = (\mathsf{AspeSetup}, \mathsf{AspeEnc}, \mathsf{AspeTokenGen}, \mathsf{AspeComparsion})$ can be defined as follows.

- $\mathsf{AspeSetup}(d)$: Suppose that all vectors in the database are d dimensional. Then, the setup algorithm outputs an invertible matrix $\mathrm{M} \in \mathbb{R}^{d \times d}$ as the secret key, where $\mathbb{R}$ denotes the real domain.
- $\mathsf{AspeEnc}(\mathrm{M}, \mathbf{x})$: Given the secret key M, a d-dimensional vector $\mathbf{x}$ in the database can be encrypted as $\mathrm{CT}_{\mathbf{x}} = \mathbf{x}\mathrm{M}$.
- $\mathsf{AspeTokenGen}(\mathrm{M}, \mathbf{q})$: Given the secret key M and a d-dimensional query vector $\mathbf{q}$, a query token can be generated as $\mathrm{TK}_{\mathbf{q}} = r\mathbf{q}(\mathrm{M}^{-1})^T$, where $r \in \mathbb{R}$ is a random positive real number.
- $\mathsf{AspeComparsion}(\mathrm{CT}_{\mathbf{x}_1}, \mathrm{CT}_{\mathbf{x}_2}, \mathrm{TK}_{\mathbf{q}})$: Given two ciphertexts $\mathrm{CT}_{\mathbf{x}_1} = \mathbf{x}_1\mathrm{M}$, $\mathrm{CT}_{\mathbf{x}_2} = \mathbf{x}_2\mathrm{M}$, and a query token $\mathrm{TK}_{\mathbf{q}} = r\mathbf{q}(\mathrm{M}^{-1})^T$, checking whether $\mathbf{x}_1 \circ \mathbf{q}$ is equal to or greater than $\mathbf{x}_2 \circ \mathbf{q}$ is equivalent to check whether $\mathrm{CT}_{\mathbf{x}_1} \circ \mathrm{TK}_{\mathbf{q}}$ is equal to or greater than $\mathrm{CT}_{\mathbf{x}_2} \circ \mathrm{TK}_{\mathbf{q}}$. This is because that

$$\begin{aligned} &\mathrm{CT}_{\mathbf{x}_1} \circ \mathrm{TK}_{\mathbf{q}} \geq \mathrm{CT}_{\mathbf{x}_2} \circ \mathrm{TK}_{\mathbf{q}} \\ \Rightarrow\ &\mathrm{CT}_{\mathbf{x}_1}\mathrm{TK}_{\mathbf{q}}^T \geq \mathrm{CT}_{\mathbf{x}_2}\mathrm{TK}_{\mathbf{q}}^T \\ \Rightarrow\ &\mathbf{x}_1\mathrm{M}r\mathrm{M}^{-1}\mathbf{q}^T \geq \mathbf{x}_2\mathrm{M}r\mathrm{M}^{-1}\mathbf{q}^T \\ \Rightarrow\ &r\mathbf{x}_1\mathrm{M}\mathrm{M}^{-1}\mathbf{q}^T \geq r\mathbf{x}_2\mathrm{M}\mathrm{M}^{-1}\mathbf{q}^T \\ \Rightarrow\ &r\mathbf{x}_1\mathbf{q}^T \geq r\mathbf{x}_2\mathbf{q}^T \end{aligned}$$

Due to $r > 0$, so we have

$$\mathbf{x}_1\mathbf{q}^T \geq \mathbf{x}_2\mathbf{q}^T \Rightarrow \mathbf{x}_1 \circ \mathbf{q} \geq \mathbf{x}_2 \circ \mathbf{q}$$

C. The Overview of SPE-Sim Scheme

Predicate encryption is a powerful encryption paradigm that allows for fine-grained access control over the encrypted data [22]. The owner of the secret key can release partial keys, called tokens, which can only decrypt a specific subset of encrypted data. In the literature, a predicate encryption scheme was designed in the public-key setting [23]. However, Shen et al. [22] pointed out that the predicate encryption in the public-setting is likely to violate the predicate privacy. If a predicate is encrypted by a public key, the adversary can encrypt any selected plaintexts to evaluate whether the plaintext satisfies the encrypted predicate or not, which possibly leaks the predicate information. Thus, in order to protect the predicate privacy, the predicate encryption should be inherently designed in the symmetric-key setting.

In this paper, we will design a symmetric-key predicate encryption scheme (SPE-Sim) supporting similarity search over binary vectors. In the SPE-Sim, each data record $(x_1, x_2, \cdots, x_d)$ is a d-dimensional binary vector. A naive solution to design the SPE-Sim is to encrypt each binary vector. Then, the similarity search can be processed as (i) generate a query token; (ii) with the query token, traverse all vectors in the database to find out those vectors satisfying the query criterion. However, the computational complexity of query processing in this case will be linear to the database size. In order to improve the query efficiency, we consider to encrypt a collection of vectors to be one ciphertext. When performing the similarity search, it only requires one query processing operation to determine whether there exists one vector in the collection that satisfies the query condition. If not, the collection of vectors can be pruned. Otherwise, we can continue to process corresponding subcollections.

Specifically, we aggregate the binary vectors sharing the same prefix into one vector. For example, the collection of vectors $\{(0,1,\mathbf{0},\mathbf{0}), (0,1,\mathbf{0},\mathbf{1}), (0,1,\mathbf{1},\mathbf{0}), (0,1,\mathbf{1},\mathbf{1})\}$ can be aggregated to be one vector $\mathbf{x} = (0,1,*,*)$, where "*"

denotes that the third and fourth element in $\mathbf{x}$ is either 0 or 1. Suppose that each aggregated vector is a d-dimensional vector and in the form of $\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2})$, where $d_1 + d_2 = d$. At the same time, $\mathbf{x}_{d_1} \in \{0,1\}^{d_1}$ is a d_1-dimensional binary vector. $\mathbf{x}_{d_2} = \underbrace{\{*, \cdots, *\}}_{d_2}$ is a d_2-dimensional vector. For simplicity, we denote $\mathbf{x}_{d_2}$ by $\{*\}^{d_2}$, where the symbol "*" denotes that each element in $\mathbf{x}_{d_2}$ can be either 0 or 1. That is, $\mathbf{x}$ is associated with a collection of d-dimensional binary vectors sharing the same prefix $\mathbf{x}_{d_1}$, which can be denoted by $\widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}} | \widetilde{\mathbf{x}} \in \{0,1\}^d, \widetilde{\mathbf{x}}_{d_1} = \mathbf{x}_{d_1}\}$.

Let $\mathcal{W} = \{\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) | \mathbf{x}_{d_1} \in \{0,1\}^{d_1}, \mathbf{x}_{d_2} = \{*\}^{d_2}\}$ denote the class of plaintexts. At the same time, each $\mathbf{x}$ is associated with a collection of vectors $\widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}} | \widetilde{\mathbf{x}} \in \{0,1\}^d, \widetilde{\mathbf{x}}_{d_1} = \mathbf{x}_{d_1}\}$. Let $\mathcal{F} = \{f_{\delta,\mathbf{q}} | \mathbf{q} \in \{0,1\}^d \setminus \mathbf{0}^d,\ 0 \le \delta \le 1\}$ denote the class of the predicates functions. The function $f_{\delta,\mathbf{q}}(\mathbf{x})$ is equal to 1 *iff* there exists at least one vector $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}$ such that $J(\widetilde{\mathbf{x}}, \mathbf{q}) \ge \delta$, i.e., the upper bound of the similarity between $\mathbf{q}$ and the vectors in $\widetilde{\mathcal{X}}$ is equal to or greater than δ, where $J(\widetilde{\mathbf{x}}, \mathbf{q}) = \frac{\widetilde{\mathbf{x}} \circ \mathbf{q}}{|\widetilde{\mathbf{x}}| + |\mathbf{q}| - \widetilde{\mathbf{x}} \circ \mathbf{q}}$ ($|\widetilde{\mathbf{x}}| = \widetilde{\mathbf{x}} \circ \widetilde{\mathbf{x}}, |\mathbf{q}| = \mathbf{q} \circ \mathbf{q}$).

Then, our SPE-Sim $\Pi_{\mathsf{SPE-Sim}} = (\mathsf{SpeSetup}, \mathsf{SpeEnc}, \mathsf{SpeTokenGen}, \mathsf{SpeQuery})$ can be defined as follows.

- $\mathsf{SpeSetup}(d)$: Given a parameter d, the setup algorithm outputs a secret key sk.
- $\mathsf{SpeEnc}(sk, \mathbf{x})$: Given the secret key sk, a d-dimensional vector $\mathbf{x} \in \mathcal{W}$ can be encrypted as a ciphertext $\mathrm{CT}_{\mathbf{x}}$.
- $\mathsf{SpeTokenGen}(sk, f_{\delta,\mathbf{q}})$: Given the secret key sk and a description of a predicate $f_{\delta,\mathbf{q}}$, the token generation algorithm outputs a token $\mathrm{TK}_{\delta,\mathbf{q}}$.
- $\mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$: Let $\mathrm{TK}_{\delta,\mathbf{q}}$ and $\mathrm{CT}_{\mathbf{x}}$ respectively denote the token of the predicate $f_{\delta,\mathbf{q}}$ and the ciphertext of $\mathbf{x}$. The query algorithm outputs 0 or 1 to indicate the value of $f_{\delta,\mathbf{q}}$ evaluated on the plaintext $\mathbf{x}$.

Correctness: Generally speaking, the SPE-Sim is correct *iff* $\mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$ can return the correct result. That is,

- If $f_{\delta,\mathbf{q}}(\mathbf{x}) = 1$, i.e., there exists at least one vector $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}$ such that $J(\widetilde{\mathbf{x}}, \mathbf{q}) \ge \delta$, the query algorithm $\mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$ returns 1.
- If $f_{\delta,\mathbf{q}}(\mathbf{x}) = 0$, i.e., the similarity between any vector $\widetilde{\mathbf{x}} \in \mathcal{X}$ and $\mathbf{q}$ is less than δ, i.e., $J(\widetilde{\mathbf{x}}, \mathbf{q}) < \delta$, the query algorithm $\mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$ returns 0.

Security: Same as [24], the semantic security of our SPE-Sim is proved in the real/ideal setting, which can subsume the traditional security definitions in the indistinguishability setting. Before formally defining the security, we first define the trivial leakage in the SPE-Sim. For a given predicate $f_{\delta,\mathbf{q}}$ and a given message $\mathbf{x}$, the leakage of SPE-Sim is the query matching result $\mathcal{L} = \mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$. With the leakage $\mathcal{L}$, the real and ideal experiments of SPE-Sim can be defined as follows.

Real experiment: In the real experiment, it involves a challenger and a PPT (probabilistic polynomial-time) adversary $\mathcal{A}$, who will interact as follows.

- *Setup*: In the setup phase, the adversary $\mathcal{A}$ chooses a plaintext vector $\mathbf{x} \in \mathcal{W}$ and sends it to the challenger. Then, the challenger runs the algorithm $\mathsf{SpeSetup}(d)$ to generate a secret key sk.
- *Query phase 1*: The adversary $\mathcal{A}$ adaptively chooses the predicate $f_{\delta_j,\mathbf{q}_j}$ for $1 \le j \le p_1$, where p_1 is a polynomial number. Then, the challenger takes sk and $f_{\delta_j,\mathbf{q}_j}$ as input to run the token generation algorithm and returns the token $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ to $\mathcal{A}$.
- *Challenge phase*: The challenger takes sk and $\mathbf{x}$ as inputs to run the encryption algorithm and returns $\mathrm{CT}_{\mathbf{x}} = \mathsf{SpeEnc}(sk, \mathbf{x})$ to $\mathcal{A}$.
- *Query phase 2*: The adversary $\mathcal{A}$ runs a protocol same as the query phase 1 and receives $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ for $p_1 < j \le p_2$, where p_2 is a polynomial number.

In the real experiment, the view of a PPT distinguisher $\mathcal{D}$ includes $\mathrm{CT}_{\mathbf{x}}$ and $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ for $1 \le j \le p_2$, i.e., $\mathrm{View}_{\mathcal{A},\mathrm{Real}} = \{\mathrm{CT}_{\mathbf{x}}, \mathrm{TK}_{\delta_j,\mathbf{q}_j} | 1 \le j \le p_2\}$.

Ideal experiment: In the ideal experiment, it involves a simulator with leakage $\mathcal{L}$ and a PPT adversary $\mathcal{A}$, who will interact as follows.

- *Setup*: In the setup phase, the adversary $\mathcal{A}$ chooses a plaintext vector $\mathbf{x} \in \mathcal{W}$ and sends it to the simulator. On receiving $\mathbf{x}$, the simulator randomly chooses a ciphertext $\mathrm{CT}_{\mathbf{x}}$.
- *Query phase 1*: The adversary $\mathcal{A}$ adaptively chooses the predicate $f_{\delta_j,\mathbf{q}_j}$ for $1 \le j \le p_1$. Then, the simulator takes the leakage $\mathcal{L} = \mathsf{SpeQuery}(\mathrm{TK}_{\delta_j,\mathbf{q}_j}, \mathrm{CT}_{\mathbf{x}})$ as inputs and outputs the token $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ to the adversary $\mathcal{A}$.
- *Challenge phase*: The simulator returns $\mathrm{CT}_{\mathbf{x}}$ to the adversary $\mathcal{A}$.
- *Query phase 2*: The adversary $\mathcal{A}$ runs a protocol same as the query phase 1 and receives $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ for $p_1 < j \le p_2$.

In the ideal experiment, the view of a PPT distinguisher $\mathcal{D}$ is also $\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}} = \{\mathrm{CT}_{\mathbf{x}}, \mathrm{TK}_{\delta_j,\mathbf{q}_j} | 1 \le j \le p_2\}$.

Definition 3 (Security of SPE-Sim): The SPE-Sim is adaptively secure with respect to the leakage $\mathcal{L}$ *iff* for all PPT adversaries $\mathcal{A}$ issuing polynomial numbers of predicate encryptions (i.e., p_2), there exists a PPT simulator such that the advantage of a PPT distinguisher $\mathcal{D}$ can distinguish the real and ideal experiments is negligible, i.e., $\Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{Real}}) = 1] - \Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}}) = 1]$ is negligible.

IV. Our Proposed Scheme

In this section, we present our proposed SetSim scheme. Before develing into the details of our proposed scheme, we first introduce our detailed SPE-Sim construction, which serves as the building block of our proposed scheme.

A. Detailed SPE-Sim Construction

Let $\mathcal{W} = \{\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) | \mathbf{x}_{d_1} \in \{0,1\}^{d_1}, \mathbf{x}_{d_2} = \{*\}^{d_2}\}$ and $\mathcal{F} = \{f_{\delta,\mathbf{q}} | \mathbf{q} \in \{0,1\}^d \setminus \mathbf{0}^d,\ 0 \le \delta \le 1\}$ respectively denote the class of plaintexts and the class of predicate functions. Suppose that $\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) \in \mathcal{W}$ is a vector in the database and $f_{\delta,\mathbf{q}} \in \mathcal{F}$ is a predicate function, where $\mathbf{q} \in \{0,1\}^d \setminus \mathbf{0}^d$ and $0 \le \delta \le 1$. Then, we have the following observation.

Observation 1: Given $\mathbf{x}$ and the collection of vectors sharing the same prefix with $\mathbf{x}$, i.e., $\widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}} | \widetilde{\mathbf{x}} \in \{0,1\}^d, \widetilde{\mathbf{x}}_{d_1} = \mathbf{x}_{d_1}\}$, the upper bound of the similarities between $\mathbf{q}$ and the vectors in $\widetilde{\mathcal{X}}$ is $J(\widetilde{\mathbf{x}}_{up}, \mathbf{q})$, where $\widetilde{\mathbf{x}}_{up} = (\mathbf{x}_{d_1}, \mathbf{q}_{d_2})$.

Based on the observation, we present the detailed SPE-Sim construction $\Pi_{\mathsf{SPE-Sim}} = (\mathsf{SpeSetup}, \mathsf{SpeEnc}, \mathsf{SpeTokenGen}, \mathsf{SpeQuery})$ as follows.

- $\mathsf{SpeSetup}(d)$: Given a parameter d, the setup algorithm outputs a secret key $sk = \mathrm{M}$, where $\mathrm{M} \in \mathbb{R}^{(3d+4)\times(3d+4)}$ is a random invertible matrix in the real domain.
- $\mathsf{SpeEnc}(sk = \mathrm{M}, \mathbf{x})$: Given the secret key $sk = \mathrm{M}$, the plaintext $\mathbf{x} \in \mathcal{W}$ can be encrypted as follows.
  Step 1: Extend d-dimensional $\mathbf{x}$ to be a $2d$-dimensional binary vector $\mathbf{x}'$. Suppose that $\mathbf{x}$ is in the form of $(\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) = (x_1, x_2, \cdots, x_{d_1}, *, \cdots, *)$. Then, $\mathbf{x}' = (x'_1, x'_2, \cdots, x'_{2i-1}, x'_{2i}, \cdots, x'_{2d-1}, x'_{2d})$ can be constructed by setting

$$\mathbf{x}'_{2i-1} = \begin{cases} x_i & 1 \le i \le d_1 \\ 0 & d_1 < i \le d \end{cases} \tag{3}$$

$$\mathbf{x}'_{2i} = \begin{cases} 0 & 1 \le i \le d_1 \\ 1 & d_1 < i \le d \end{cases} \tag{4}$$

  Step 2: According to $\mathbf{x}$, construct a d-dimensional vector $\mathbf{x}^{\#}$, where

$$\mathbf{x}_i^{\#} = \begin{cases} 0 & 1 \le i \le d_1 \\ 1 & d_1 < i \le d \end{cases} \tag{5}$$

  Step 3: Further extend $\mathbf{x}'$ to be a $(3d+4)$-dimensional vector $\mathbf{x}'' = (\mathbf{x}', \mathbf{x}^{\#}, 1, |\mathbf{x}_{d_1}|, r_1, r_1)$, where r_1 is a random real number.
  Step 4: Encrypt $\mathbf{x}''$ as $\mathrm{CT}_{\mathbf{x}} = \mathsf{AspeEnc}(\mathrm{M}, \mathbf{x}'') = \mathbf{x}''\mathrm{M}$.
- $\mathsf{SpeTokenGen}(sk, f_{\delta,\mathbf{q}})$: Given the secret key sk and the predicate $(\delta, \mathbf{q})$, a corresponding predicate token $\mathrm{TK}_{\delta,\mathbf{q}}$ can be generated as the following steps.
  Step 1: Extend d-dimensional predicate $\mathbf{q}$ to be a $2d$-dimensional vector $\mathbf{q}'$. Suppose that $\mathbf{q} = (v_1, v_2, \cdots, v_d)$, then it will be extended to be $\mathbf{q}' = (v'_1, v'_2, \cdots, v'_{2d-1}, v'_{2d})$ by setting

$$\begin{cases} v'_{2i-1} = v_i & 1 \le i \le d \\ v'_{2i} = v_i^2 & 1 \le i \le d \end{cases} \tag{6}$$

  Step 2: Extend $\mathbf{q}'$ to be a $(3d+4)$-dimensional vector $\mathbf{q}'' = ((1+\delta)\mathbf{q}', -\delta\mathbf{q}, -\delta|\mathbf{q}|, -\delta, r_2, -r_2)$, where r_2 is random real number.
  Step 3: Encrypt $\mathbf{q}''$ as a predicate token $\mathrm{TK}_{\delta,\mathbf{q}} = \mathsf{AspeTokenGen}(\mathrm{M}, \mathbf{q}'') = r\mathbf{q}''(\mathrm{M}^{-1})^T$, where r is a random positive real number.
- $\mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$: The query algorithm takes the query token $\mathrm{TK}_{\delta,\mathbf{q}}$ and the ciphertext $\mathrm{CT}_{\mathbf{x}}$ as inputs. Then, it computes the value of $\mathrm{CT}_{\mathbf{x}} \circ \mathrm{TK}_{\delta,\mathbf{q}}$. If $\mathrm{CT}_{\mathbf{x}} \circ \mathrm{TK}_{\delta,\mathbf{q}} \ge 0$, it outputs 1 else it outputs 0.

Correctness: The correctness of our SPE-Sim can be verified as follows. First, we have the following fact,

$$\begin{aligned}
&\mathrm{CT}_{\mathbf{x}} \circ \mathrm{TK}_{\delta,\mathbf{q}} \\
=&\mathrm{CT}_{\mathbf{x}} \mathrm{TK}_{\delta,\mathbf{q}}^T \\
=&r\mathbf{x}''\mathrm{M}\mathrm{M}^{-1}\mathbf{q}''^T \\
=&r\mathbf{x}'' \circ \mathbf{q}'' \\
=&r(\mathbf{x}', \mathbf{x}^{\#}, 1, |\mathbf{x}_{d_1}|, r_1, r_1) \circ ((1+\delta)\mathbf{q}', -\delta\mathbf{q}, -\delta|\mathbf{q}|, -\delta, r_2, -r_2) \\
=&r((1+\delta)\mathbf{x}' \circ \mathbf{q}' - \delta\mathbf{x}^{\#} \circ \mathbf{q} - \delta|\mathbf{q}| - \delta|\mathbf{x}_{d_1}|)
\end{aligned}$$

If $\mathrm{CT}_{\mathbf{x}} \circ \mathrm{TK}_{\delta,\mathbf{q}} \ge 0$, we have

$$r((1+\delta)\mathbf{x}' \circ \mathbf{q}' - \delta\mathbf{x}^{\#} \circ \mathbf{q} - \delta|\mathbf{q}| - \delta|\mathbf{x}_{d_1}|) \ge 0$$

Due to r is a positive real number, we have

$$((1+\delta)\mathbf{x}' \circ \mathbf{q}' - \delta\mathbf{x}^{\#} \circ \mathbf{q} - \delta|\mathbf{q}| - \delta|\mathbf{x}_{d_1}|) \ge 0$$

$$\frac{\mathbf{x}' \circ \mathbf{q}'}{|\mathbf{x}_{d_1}| + |\mathbf{q}| + \mathbf{x}^{\#} \circ \mathbf{q} - \mathbf{x}' \circ \mathbf{q}'} \ge \delta \tag{7}$$

On the one hand, we have

$$\begin{aligned}
&\mathbf{x}' \circ \mathbf{q}' \\
=&\sum_{i=1}^{d_1}(x_{2i-1} \cdot v_{2i-1} + x_{2i} \cdot v_{2i}) + \sum_{i=d_1+1}^{d}(x_{2i-1} \cdot v_{2i-1} + x_{2i} \cdot v_{2i}) \\
=&\sum_{i=1}^{d_1}(x_i \cdot v_i + 0 \cdot v_i^2) + \sum_{i=d_1+1}^{d}(0 \cdot v_i + 1 \cdot v_i^2) \\
=&(x_1, \cdots, x_{d_1}, v_{d_1+1}, \cdots, v_d) \circ \mathbf{q} \\
=&(\mathbf{x}_{d_1}, \mathbf{q}_{d_2}) \circ \mathbf{q} \\
=&\widetilde{\mathbf{x}}_{up} \circ \mathbf{q}
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
&|\mathbf{x}_{d_1}| + \mathbf{x}^{\#} \circ \mathbf{q} = |\mathbf{x}_{d_1}| + (\underbrace{0, \cdots, 0}_{d_1}, \underbrace{1, \cdots, 1}_{d_2}) \circ \mathbf{q} \\
=&|\mathbf{x}_{d_1}| + |\mathbf{q}_{d_2}| = |\widetilde{\mathbf{x}}_{up}|
\end{aligned}$$

Thus, we have

$$\frac{\mathbf{x}' \circ \mathbf{q}'}{|\mathbf{x}_{d_1}| + |\mathbf{q}| + \mathbf{x}^{\#} \circ \mathbf{q} - \mathbf{x}' \circ \mathbf{q}'} \tag{8}$$

$$= \frac{\widetilde{\mathbf{x}}_{up} \circ \mathbf{q}}{|\widetilde{\mathbf{x}}_{up}| + |\mathbf{q}| - \widetilde{\mathbf{x}}_{up} \circ \mathbf{q}} = J(\widetilde{\mathbf{x}}_{up}, \mathbf{q}) \tag{9}$$

As a result, by combining Eq. (7) and Eq. (9), we can obtain that $J(\widetilde{\mathbf{x}}_{up}, \mathbf{q}) \ge \delta$. As shown in the observation 1, $J(\widetilde{\mathbf{x}}_{up}, \mathbf{q})$ is the upper bound among the similarities between $\mathbf{q}$ and the vectors in $\widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}} | \widetilde{\mathbf{x}} \in \{0,1\}^d, \widetilde{\mathbf{x}}_{d_1} = \mathbf{x}_{d_1}\}$. Then, if $J(\widetilde{\mathbf{x}}_{up}, \mathbf{q}) \ge 0$, there exists at least one vector $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}$ such that $J(\widetilde{\mathbf{x}}, \mathbf{q}) \ge \delta$. In this case, $f_{\delta,\mathbf{q}}(\mathbf{x}) = 1$. Otherwise, $f_{\delta,\mathbf{q}}(\mathbf{x}) = 0$.

Note that when $\mathbf{x} \in \{0,1\}^d$ is a binary vector, the query algorithm $\mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$ is equivalent to check whether the vector $\mathbf{x}$ satisfies that $J(\mathbf{x}, \mathbf{q}) \ge \delta$ or not, where $\mathbf{q}$ is the query vector.

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/TDSC.2020.3004442, IEEE Transactions on Dependable and Secure Computing

B. The Description of Our Proposed Scheme

In this subsection, we present our SetSim scheme. Its main idea is to first transform the set records to integers and use these integers as key values to build a B+ tree. Then, we deploy our SPE-Sim construction to encrypt and search the B+ tree in a privacy-preserving way. In specific, our SetSim scheme is comprised of three phases, i.e., System Initialization, Local Data Outsourcing, and Set Similarity Search.

1) System Initialization: The data owner is responsible for bootstrapping the whole system. Given the parameter d, he/she first generates an invertible matrix $\mathrm{M} \in \mathbb{R}^{(3d+4)\times(3d+4)}$ and computes the inverse matrix of M, i.e., M^{-1}. Both M and M^{-1} are regarded as the secret key. Meanwhile, for each user U_i, the data owner authorizes him/her by respectively distributing random matrices $\mathrm{M}_1 \in \mathbb{R}^{(3d+4)\times(3d+4)}$ and $\mathrm{M}_2 \in \mathbb{R}^{(3d+4)\times(3d+4)}$ to U_i and the cloud server, where M_1 and M_2 satisfy that $(\mathrm{M}_1^{-1})^T\mathrm{M}_2 = (\mathrm{M}^{-1})^T$. Note that different query users have different authorized matrices.

2) Local Data Outsourcing: Suppose that the data owner has a collection of sets $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$ and each set $S_i \in \mathcal{S}$ is a subset of $\mathcal{E}$, i.e., $S_i \subseteq \mathcal{E}$, where $\mathcal{E} = \{e_1, e_2, \cdots, e_d\}$ is the collection of all elements. Then, he/she can outsource them to the cloud server as follows.

Step 1: Represent each $S_i \in \mathcal{S}$ to be a d-dimensional binary vector $\mathbf{b}_i = (x_1, x_2, \cdots, x_d)$ by setting each x_l as

$$x_l = \begin{cases} 1 & e_l \in S_i \\ 0 & e_l \notin S_i \end{cases} \tag{10}$$

for $1 \le l \le d$. After this step, the data owner has a collection of binary vectors, i.e., $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^n$.

Step 2: For each $\mathbf{b}_i = (x_1, x_2, \cdots, x_d)$ in $\mathcal{B}$, the data owner transforms it to an integer $val_i = \sum_{l=1}^{d} x_l * 2^{d-l}$. After this step, the data owner have a collection of integers corresponding to $\mathcal{B}$, i.e., $\{val_i\}_{i=1}^n$.

Step 3: Build an N-ary B+ tree for the values $\{val_i\}_{i=1}^n$ as the tree building algorithm in [25]. In the B+ tree, there are three kinds of nodes, i.e., root node, internal node and leaf node. For an N-ary B+ tree, the number of key values in the internal nodes and leaf nodes range from $\lceil \frac{N}{2} \rceil - 1$ to $N-1$. For the root node, it can be either internal node or leaf node, and the number of key values in it range from 1 to $N-1$. Apart from key values, each node also contains the pointers to its children nodes. Suppose that an internal node contains t keywords, i.e., $\{K_1, K_2, \cdots, K_t\}$, then it also contains $t+1$ pointers $\{P_1, P_2, \cdots, P_{t+1}\}$. P_1 points to the child node whose key values are less than K_1. Meanwhile, P_j points to a child node whose key values belong to $[K_{j-1}, K_j)$ for $2 \le j \le t$. For P_{t+1}, it points to the child node whose key values are equal to or greater than K_j.

Example 2: Suppose that the collection of key values are $\{4, 9, 16, 25, 1, 20, 13, 15, 10, 11, 12\}$. A 4-ary B+ tree can be built as shown in Fig. 2.

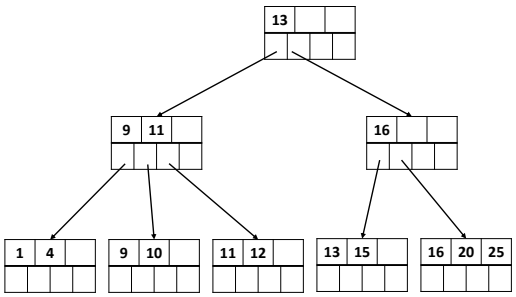

Fig. 2: An example of B+ tree for the collection of key values $\{4, 9, 16, 25, 1, 20, 13, 15, 10, 11, 12\}$

Step 4: Extend the B+ tree by extending its internal nodes. Each internal node with t values $\{K_1, K_2, \cdots, K_t\}$ will be extended by adding two additional key values K_0, K_{t+1}, where K_0 and K_{t+1} are respectively the leftmost value and rightmost value of the current node's children nodes. After extension, the node contains $t+2$ key values $\{K_0, K_1, \cdots, K_t, K_{t+1}\}$. For example, the root node with key values $\{13\}$ in Fig. 2 can be extended to ba a node with key values $\{1, 13, 25\}$. In this case, the B+ tree in Fig. 2 can be extended to be a B+ tree as shown in Fig. 3.

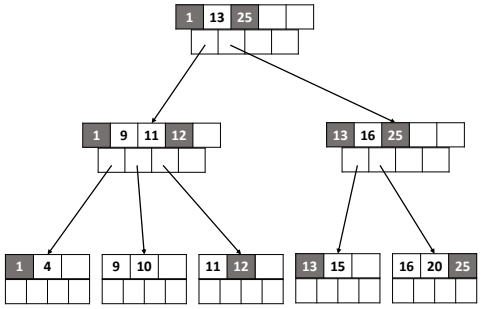

Fig. 3: An example of the extended B+ tree for the collection of key values $\{4, 9, 16, 25, 1, 20, 13, 15, 10, 11, 12\}$

Step 5: Encrypt the internal nodes in the B+ tree. Suppose that an internal node $node$ has $t+2$ key values $\{K_0, K_1, \cdots, K_t, K_{t+1}\}$ and $t+1$ pointers $\{P_1, \cdots, P_{t+1}\}$, denoted by $node = \{\{K_0, K_1, \cdots, K_t, K_{t+1}\}, \{P_1, \cdots, P_{t+1}\}\}$.

- First, represent $\{K_0, K_1, \cdots, K_t, K_{t+1}\}$ to be $t+1$ ranges, i.e., $\{[K_0, K_1), \cdots, [K_{t-1}, K_t), [K_t, K_{t+1}]\}$. In order to facilitate description, the last range $[K_t, K_{t+1}]$ can be transformed to be $[K_t, K_{t+1})$ by setting $K_{t+1} = K_{t+1} + 1$. Then, the internal node can be represented to be $node = \{[K_{j-1}, K_j), P_j\}_{j=1}^{t+1}$.
- Second, for the j-th range $[K_{j-1}, K_j)$, represent it to be a collection of d-dimensional binary vectors, i.e., $\{\mathrm{BinVec}(val) | val \in [K_{j-1}, K_j)\}$, where $\mathrm{BinVec}(val)$ is a function to transform an integer value to be a d-dimensional vector.
- Third, when some vectors in $\{\mathrm{BinVec}(val) | val \in [K_{j-1}, K_j)\}$ share the same prefix, they can be aggregated to be one vector with "*" as suffix. Then, all binary vectors can be aggregated to be several vectors, i.e., $\mathcal{X}_j = \{\mathbf{x} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) | \mathbf{x}_{d_1} \in \{0,1\}^{d_1}, \mathbf{x}_{d_2} = \{*\}^{d_2}\}$, where $\mathbf{x}_{d_1}$ and $\mathbf{x}_{d_2}$ respectively denote the first d_1 elements and the last d_2 elements of $\mathbf{x}$, and $d_1 + d_2 = d$.

  Example 3: As shown in Fig. 3, the root node has key values $\{1, 13, 25\}$ and the second range

is $[13, 26)$. Meanwhile, each value in $[13, 26)$ can be represented to be a binary vector, e.g., 20 can be represented to be $(1, 0, 1, 0, 0)$. Then, these binary vectors can be aggregated to be a set of binary vectors with suffix, i.e., $\mathcal{X}_2 = \{(0, 1, 1, 0, 1), (0, 1, 1, 1, *), (1, 0, *, *, *), (1, 1, 0, 0, *)\}$, where $(0, 1, 1, 0, 1)$ represents 13, $(0, 1, 1, 1, *)$ represents $\{14, 15\}$, (1, 0, *, *, *) represents the values from 16 to 23, and (1, 1, 0, 0, *) represents $\{24, 25\}$.
In this case, the internal node can be represented to be $node = \{\mathcal{X}_j, P_j\}_{j=1}^{t+1}$.

- Furthermore, for each $\mathbf{x} \in \mathcal{X}_j$, it can be encrypted as $\mathrm{CT}_{\mathbf{x}} = \mathsf{SpeEnc}(\mathrm{M}, \mathbf{x})$ by the encryption algorithm in Subsection IV-A. In this case, $\mathcal{X}_j$ can be encrypted to be $\mathsf{E}(\mathcal{X}_j) = \{\mathrm{CT}_{\mathbf{x}} = \mathsf{SpeEnc}(\mathrm{M}, \mathbf{x}) | \mathbf{x} \in \mathcal{X}_j\}$. Then, the internal node $node$ will be encrypted as $\mathsf{E}(node) = \{\mathsf{E}(\mathcal{X}_j), P_j\}_{j=1}^{t+1}$.

Step 6: Encrypt the leaf nodes in the B+ tree. Suppose that a leaf node $node$ contains t values $\{K_j\}_{j=1}^{t}$, it can be encrypted as follows.

- First, represent $\{K_j\}_{j=1}^{t}$ to be t d-dimensional vectors, i.e., $\{\mathrm{BinVec}(K_j) | j = 1, 2, \cdots, t\}$, where $\mathrm{BinVec}(K_j)$ is a function to transform an integer value to be a d-dimensional vector.
- Second, for each $\mathrm{BinVec}(K_j)$, it can be encrypted as $\mathrm{CT}_j = \mathsf{SpeEnc}(\mathrm{M}, \mathrm{BinVec}(K_j))$ by the encryption algorithm in Subsection IV-A for $j = 1, 2, \cdots, t$.
- Then, the leaf node $node$ is encrypted as $\mathsf{E}(node) = \{\mathrm{CT}_j\}_{j=1}^{t}$.

Step 7: Finally, the data owner encrypts the B+ tree by replacing each internal node and leaf node with their corresponding ciphertexts. Then, in order to preserve the order privacy of the values in the B+ tree, the data owner permutates the values in internal nodes and leaf nodes. Finally, he/she outsources the encrypted B+ tree T to the cloud server.

3) Set Similarity Search: Given a query set Q and similarity threshold δ, the authorized user U can perform the set similarity search as the following steps.

Step 1: U first represents the query set Q to be a d-dimensional binary vector as the Eq. (10), i.e., $\mathbf{q} = (v_1, v_2, \cdots, v_d)$.

Step 2: U uses its authorized key M_1 to encrypt the vector $\mathbf{q}$ and δ as the token generation algorithm in Subsection IV-A, i.e., $\mathrm{TK}_{\delta,\mathbf{q}} = \mathsf{SpeTokenGen}(\mathrm{M}_1, f_{\delta,\mathbf{q}}) = r\mathbf{q}''(\mathrm{M}_1^{-1})^T$.

Step 3: U sends the query request together with $\mathrm{TK}_{\delta,\mathbf{q}}$ to the cloud server.

Step 4: On receiving the query request and $\mathrm{TK}_{\delta,\mathbf{q}}$, the cloud server first verifies whether U is authorized or not. The query user authentication can be achieved by an existing authentication protocol, e.g., password authentication protocol [26]. If not, it will reject the current request. Otherwise, it will update the query token $\mathrm{TK}_{\delta,\mathbf{q}}$ as

$$\begin{aligned}\mathrm{TK}'_{\delta,\mathbf{q}} &= \mathrm{TK}_{\delta,\mathbf{q}} \times \mathrm{M}_2 \\ &= r\mathbf{q}''(\mathrm{M}_1^{-1})^T \times \mathrm{M}_2 \\ &= r\mathbf{q}''(\mathrm{M}^{-1})^T\end{aligned}$$

Then, the cloud server searches the encrypted B+ tree T to obtain the targeted sets as the range search algorithm in Algorithm 1. The main idea is to first obtain the candidate leaf nodes that may contain the query result by iteratively traversing T with a depth-first search algorithm, as shown in Algorithm 2. Then, check the candidate leaf nodes one by one to get the finial query result as shown in lines 3-7 in Algorithm 1.

Algorithm 1 $RangeSearch(T, \mathrm{TK}'_{\delta,\mathbf{q}})$

Input: The encrypted B+ tree T and the updated query token $\mathrm{TK}'_{\delta,\mathbf{q}}$
Output: $\mathcal{R} = \{\mathrm{CT}_j | \mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) = 1\}$
1: $\mathcal{R} = \emptyset$
2: $LeafSet = IterSearch(\mathrm{TK}'_{\delta,\mathbf{q}}, T.root)$
3: **for** each $LNode$ in $LeafSet$ **do**
4: $\quad$ *$LNode$ is encrypted as $\{\mathrm{CT}_j\}_{j=1}^{t}$
5: $\quad$ **for** $j = 1, 2, \cdots, t$ **do**
6: $\quad\quad$ **if** $\mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) == 1$ **then**
7: $\quad\quad\quad$ $\mathcal{R} = \mathcal{R} \cup \{\mathrm{CT}_j\}$
return $\mathcal{R}$

Algorithm 2 $IterSearch(\mathrm{TK}'_{\delta,\mathbf{q}}, node)$

Input: The query token $\mathrm{TK}'_{\delta,\mathbf{q}}$ and the node needed to be searched $node$
Output: The set of leaf nodes containing the possible query result
1: $LeafSet = \emptyset$
2: **if** $node$ is a leaf node **then**
3: $\quad$ $LeafSet = LeafSet \cup node$
4: **for** $j = 1, 2, \cdots, t + 1$ **do**
5: $\quad$ *The node $node$ is encrypted as $\mathsf{E}(node) = \{\mathsf{E}(\mathcal{X}_j), P_j\}_{j=1}^{t+1}$
6: $\quad$ **for** each $\mathrm{CT}_{\mathbf{x}} \in \mathsf{E}(\mathcal{X}_j)$ **do**
7: $\quad\quad$ **if** $\mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}}) == 1$ **then**
8: $\quad\quad\quad$ $IterSearch(\mathrm{TK}'_{\delta,\mathbf{q}}, node.P_j)$
9: $\quad\quad\quad$ **break**
10:
return $LeafSet$

- Iteratively traversing: For each node $node$, if it is a leaf node, put it into the set of candidate leaf nodes, i.e., $LeafSet = LeafSet \cup node$. Otherwise, if it is an internal node $\mathsf{E}(node) = \{\mathsf{E}(\mathcal{X}_j), P_j\}_{j=1}^{t+1}$, the cloud server checks each $\mathsf{E}(\mathcal{X}_j)$ for $j = 1, 2, \cdots, t + 1$. If there exists a $\mathrm{CT}_{\mathbf{x}} \in \mathsf{E}(\mathcal{X}_j)$ such that $\mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}}) = 1$, based on the SPE-Sim construction, there may exist a set $S_i \in \mathcal{S}$ such that the integer corresponding to its binary representation falls in the range $[K_{j-1}, K_j)$ and $J(S_i, Q) \geq \delta$. In this case, the cloud server needs to search the child node pointed by P_j. Otherwise, the child node pointed by P_j can be pruned.
- Checking: For each leaf node $LNode$ in candidate set $LeafSet$, it has been encrypted to be $\{\mathrm{CT}_j\}_{j=1}^{t}$. For each CT_j, the cloud server computes $\mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j)$. If $\mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) = 1$, the cloud server adds CT_j to the query result $\mathcal{R}$.

Finally, the cloud server obtains the query result $\mathcal{R} = \{\mathrm{CT}_j | \mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) = 1\}$.

Step 5: The cloud server uses U's authorized key M_2 to partly decrypt the query result. That is, for each $\mathrm{CT}_j \in \mathcal{R}$, the cloud server computes $\mathrm{CT}_j\mathrm{M}_2^T$. Then, the cloud server returns $\mathcal{R}' = \{\mathrm{CT}_j\mathrm{M}_2^T | \mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) = 1\}$ to U.

Step 6: When the user U receives the query result, he/she uses the authorized key M_1 to recover the vector encrypted in CT_j. That is, $\mathbf{x}'' = \mathrm{CT}_j\mathrm{M}_2^T\mathrm{M}_1^{-1}$. From $\mathbf{x}''$, U_i can recover

the d-dimensional vector $\mathbf{x}$ by setting $x_i = x''_{2i-1}$ for $i = 1, 2, \cdots, d$. Finally, the user transforms each vector $\mathbf{x}$ to be a set $S_j \in \mathcal{S}$. In specific, S_j contains e_i if x_i is 1 for $i = 1, 2, \cdots, d$.

Correctness: The query result is correct because

$$\begin{aligned}\mathrm{CT}_j \mathrm{M}_2^T \mathrm{M}_1^{-1} &= \mathrm{CT}_j((\mathrm{M}_1^{-1})^T \mathrm{M}_2))^T \\ &= \mathrm{CT}_j((\mathrm{M}^{-1})^T)^T \\ &= \mathbf{x}''\mathrm{M}\mathrm{M}^{-1} \\ &= \mathbf{x}''\end{aligned}$$

V. Security Analysis

In this section, we analyze the security of our proposed SetSim scheme. Since our SPE-Sim construction is its critical building block, we first prove the security of our SPE-Sim construction.

A. Security Analysis of Our SPE-Sim Construction

Theorem 1: Our SPE-Sim construction is adaptively simulation-secure with leakage $\mathcal{L} = \mathsf{SpeQuery}(\mathrm{TK}_{\delta,\mathbf{q}}, \mathrm{CT}_{\mathbf{x}})$, where $\mathrm{TK}_{\delta,\mathbf{q}}$ is the query token of predicate $f_{\mathbf{q},\delta}$ and $\mathrm{CT}_{\mathbf{x}}$ is the ciphertext of $\mathbf{x}$.

Proof: We first construct a simulator for the ideal experiment as follows.

- *Setup*: In the setup phase, the adversary $\mathcal{A}$ chooses a plaintext vector $\mathbf{x} \in \mathcal{W}$ and sends it to the simulator. On receiving $\mathbf{x}$, the simulator randomly chooses a $(3d+4)$-dimensional vector as $\mathrm{CT}_{\mathbf{x}}$.
- *Query phase 1*: The adversary $\mathcal{A}$ adaptively chooses the predicate $f_{\delta_j,\mathbf{q}_j}$ for $1 \leq j \leq p_1$. For each predicate, the simulator takes the leakage $\mathcal{L} = \mathsf{SpeQuery}(\mathrm{TK}_{\delta_j,\mathbf{q}_j}, \mathrm{CT}_{\mathbf{x}})$ as inputs and outputs the token $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ as follows.
  - If $\mathsf{SpeQuery}(\mathrm{TK}_{\delta_j,\mathbf{q}_j}, \mathrm{CT}_{\mathbf{x}}) == 1$, the simulator randomly selects a vector $(3d+4)$-dimensional vector $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ such that $\mathrm{TK}_{\delta_j,\mathbf{q}_j} \circ \mathrm{CT}_{\mathbf{x}} \geq 0$.
  - If $\mathsf{SpeQuery}(\mathrm{TK}_{\delta_j,\mathbf{q}_j}, \mathrm{CT}_{\mathbf{x}}) == 0$, the simulator randomly selects a vector $(3d+4)$-dimensional vector $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ such that $\mathrm{TK}_{\delta_j,\mathbf{q}_j} \circ \mathrm{CT}_{\mathbf{x}} < 0$.

  Then, the simulator returns $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ to the adversary $\mathcal{A}$.
- *Challenge phase*: The simulator provides $\mathrm{CT}_{\mathbf{x}}$ to the adversary $\mathcal{A}$.
- *Query phase 2*: The adversary $\mathcal{A}$ runs a protocol same as the query phase 1 and receives $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ for $p_1 < j \leq p_2$.

In the real experiment, the view of a PPT distinguisher $\mathcal{D}$ is $\mathrm{CT}_{\mathbf{x}}$ and $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$. Before $\mathbf{x}$ is encrypted to be $\mathrm{CT}_{\mathbf{x}}$, $\mathbf{x}$ needs to be transformed to be a $(3d+4)$-dimensional vector $\mathbf{x}'' = (\mathbf{x}', \mathbf{x}^{\#}, 1, |\mathbf{x}_{d_1}|, r_1, r_1)$ and r_1 is a random real number. The random number r_1 makes the ciphertext $\mathrm{CT}_{\mathbf{x}}$ be a random vector. Similarly, the query vector $\mathbf{q}_j$ is also extended to contain two random numbers r_2 and r, so the query token $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ looks like a random vector. Meanwhile, when a same query vector is encrypted twice, the involvement of random numbers enables that the generated two ciphertexts look like two different random vectors. In the ideal experiment, $\mathrm{CT}_{\mathbf{x}}$ and $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ in the distinguisher's view are also random numbers. Thus, $\mathrm{CT}_{\mathbf{x}}$ and $\mathrm{TK}_{\delta_j,\mathbf{q}_j}$ for $1 < j \leq p_2$ in both the real experiment and ideal experiment involve random number, the advantage of a PPT distinguisher $\mathcal{D}$ in the real and ideal experiments is negligible, i.e., $\Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{Real}}) = 1] - \Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}}) = 1]$ is negligible. ■

B. Security Analysis of Our SetSim Scheme

In this subsection, we analyze the security of our SetSim scheme. In particular, we focus on the privacy-preserving properties, i.e., (i) the encrypted sets stored in the cloud server are privacy-preserving; (ii) the query predicate is privacy-preserving; (iii) the query result is privacy-preserving.

• *The encrypted sets stored in the cloud server is privacy-preserving:* As described in Section IV, the collection of sets is encrypted to be a B+ tree and outsourced to the cloud server, which mainly includes two kinds of nodes, i.e., internal nodes and leaf nodes. The internal node $node = \{[K_{j-1}, K_j), P_j\}_{j=1}^{t+1}$ will be encrypted as $\mathsf{E}(node) = \{\mathsf{E}(\mathcal{X}_j), P_j\}_{j=1}^{t+1}$, where $\mathsf{E}(\mathcal{X}_j)$ is a set of ciphertexts encrypted by the encryption algorithm of our SPE-Sim construction. If the adversary intends to obtain the information about the sets, it may first attempt to obtain some information about the range $[K_{j-1}, K_j)$ from the ciphertexts in $\mathsf{E}(\mathcal{X}_j)$. Since each ciphertext $\mathrm{CT}_{\mathbf{x}} \in \mathsf{E}(\mathcal{X}_j)$ is encrypted by the encryption algorithm of our SPE-Sim construction, the adaptive security of our SPE-Sim construction can guarantee that the cloud server cannot obtain any plaintexts information about the range $[K_{j-1}, K_j)$ from the ciphertexts. For each leaf node $node = \{K_1, \cdots, K_t\}$, it is encrypted to be $\{\mathrm{CT}_j\}_{j=1}^{t}$. Each CT_j is encrypted by the encryption algorithm of our SPE-Sim construction. Similar to the internal nodes, the adaptive security of SPE-Sim construction can guarantee that the cloud server has no idea on the plaintext of K_j from CT_j for $j = 1, 2, \cdots, t$. Thus, from each internal node and leaf node, the cloud server cannot obtain any information about the key values from corresponding ciphertexts. For the whole encrypted B+ tree, since the key values in each internal node and leaf node have been permutated before outsourcing to the cloud, the order information of the values among sibling nodes in the B+ tree can also be preserved. Although the order information between the parent nodes and their child nodes is leaked, the permutation technique can still guarantee that it is difficult for the cloud server to learn about the key values information of internal nodes and leaf nodes. This is because once the key values in the internal nodes and leaf nodes are permutated, the possible range of the key values in the internal nodes and leaf nodes will be the whole domain. In this case, the order information between the parent nodes and their child nodes cannot contribute to deducing the key values of B+ tree. Therefore, the encrypted sets stored in the cloud server is privacy-preserving.

• *The query predicate is privacy-preserving:* During the process of the set similarity search, our scheme should keep the query predicate secret from the unauthorized entities, including the cloud server and unauthorized users. For the cloud server, since the query predicate $f_{\delta,\mathbf{q}}$ has been encrypted

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/TDSC.2020.3004442, IEEE Transactions on Dependable and Secure Computing

to be a query token $\mathrm{TK}_{\delta,\mathbf{q}}$ before sending to the cloud server, the security of our SPE-Sim construction guarantees that the cloud server has no idea on the plaintext of the query predicate.

After receiving the query token, the cloud server is likely to deduce the query predicate from the process of searching B+ tree, which contains two steps, i.e., iteratively traverse to obtain the candidate leaf nodes and check the candidate leaf nodes. First, during the process of traversing, for each internal node $\mathsf{E}(node) = \{\mathsf{E}(\mathcal{X}_j), P_j\}_{j=1}^{t+1}$, the cloud server will compute the inner product of the query token $\mathrm{TK}'_{\delta,\mathbf{q}}$ with each $\mathrm{CT}_{\mathbf{x}} \in \mathsf{E}(\mathcal{X}_j)$. The inner product only leaks whether there may be sets whose integer representation falls into the corresponding plaintext range. However, since the key values range in each internal node is permutated, the cloud server has no idea on the plaintext range of this internal node. In this case, it also has no idea on the plaintext of the query predicate. Second, during the process of checking, the cloud server needs to check whether each key value in a leaf node is the query result. In this process, the only leakage is the inner product between the query token $\mathrm{TK}'_{\delta,\mathbf{q}}$ and each ciphertext CT_j in the leaf node. Similar to the internal node, the permutation of the B+ tree prevents the cloud server from deducing the plaintext of the query predicate, i.e., $(\delta, \mathbf{q})$. Thus, during the query process, the cloud server has no idea on the plaintext of the predicate.

For the unauthorized users, each of them is authorized by a random invertible matrix M_1. At the same time, different users are authorized by different random matrices, and each user encrypts his/her query request using his/her authorized key. In this case, the information that other unauthorized users can use to deduce the query predicate is no more than the information that the cloud server can use. Thus, other unauthorized users also have no idea on the plaintext of the query predicate. Therefore, both the cloud server and the unauthorized users have no idea on the query predicate.

• *The query result is privacy-preserving:* The query result $\mathcal{R}' = \{\mathrm{CT}_j \mathrm{M}_2^T | \mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) = 1\}$ has been encrypted. For the cloud server, it can also see the result $\mathcal{R} = \{\mathrm{CT}_j | \mathsf{SpeQuery}(\mathrm{TK}'_{\delta,\mathbf{q}}, \mathrm{CT}_j) = 1\}$. However, since it has no idea on the encryption matrix M, it also has no idea on the plaintext of query result. For other authorized users, they may eavesdrop the encrypted query result $\mathcal{R}'$. However, since they have no idea on the the authorized key M_1 held by the query user, they cannot recover the plaintext of the query result. Therefore, the query result is kept secret from both the cloud server and other authorized users.

VI. Performance Evaluation

In this section, we experimentally evaluate the performance of our proposed scheme with respect to the computational cost of local data outsourcing and set similarity search. We implemented our scheme in Java and conducted experiments on a machine with an Intel(R) Core(TM) i7-3770 CPU @3.40GHz, 16GB RAM and Windows 10 operating system. In our experiments, we evaluate our scheme on a real joke rating dataset Jester [27] from 24,983 users who have rated 36 or more jokes (there are 100 jokes in total). Each rating is a real number ranging from -10.00 to 10.00. We transform the Jester dataset to be a collection of sets and each set is comprised of the jokes that a specific user is rated and the rating value is equal to or larger than 7. That is, each set contains a set of jokes that a specific user likes. At the same time, we remove the duplicate sets. After the data processing, the dataset contains 11,987 sets and 100 elements, i.e., $|\mathcal{S}| = 11,987$ and $|\mathcal{E}| = 100$.

Since most of the existing similarity query schemes are either weak in security or cannot support exact set similarity search, in our experiment, we compare our proposed scheme with a secure kNN query protocol, i.e., the basic protocol in [11]. The basic protocol was initially proposed to achieve the Euclidean distance based kNN search, and it can be deploy to achieve the set similarity range search by representing each set record to be a binary vector as described in Subsection IV-B.

A. Local Data Outsourcing

In our scheme, the data owner is responsible for encrypting and outsourcing its local data to the cloud as described in Subsection IV-B. The computational cost of local data outsourcing involves (i) representing the collection of sets to be a B+ tree; (ii) encrypting each key values in the B+ tree with the encryption of our SPE-Sim construction, which is linear to the total number of sets, i.e., $O(|\mathcal{S}|)$. For the basic protocol in [11], the local data outsourcing phase is to encrypt each binary vector by encrypting each value in the vector with Paillier encryption algorithm, i.e., a vector $(b_1, b_2, \cdots, b_d)$ is encrypted to be $(\mathsf{PEnc}(b_1), \mathsf{PEnc}(b_1), \cdots, \mathsf{PEnc}(b_d))$, where $\mathsf{PEnc}(\cdot)$ is the Paillier encryption algorithm. Thus, outsourcing $|\mathcal{S}|$ set records to the cloud requires $|\mathcal{S}| \times d$ Paillier encryption operations. When the Paillier encryption key size is set to be 1024, each Pailler encryption operation takes about 6 ms. Fig. 4 plots the computational cost of local data outsourcing versus the number of sets (i.e., $|\mathcal{S}|$) over the Jester dataset, where $d = |\mathcal{E}| = 100$. From this figure, we can see that the computational cost of local data outsourcing in both the basic protocol [11] and our proposed scheme linearly increases with $|\mathcal{S}|$, while our proposed scheme is much more efficient. For example, when $|\mathcal{S}| = 5000$, the computational cost in our scheme is 2058 ms while that of basic protocol is 3×10^6 ms.

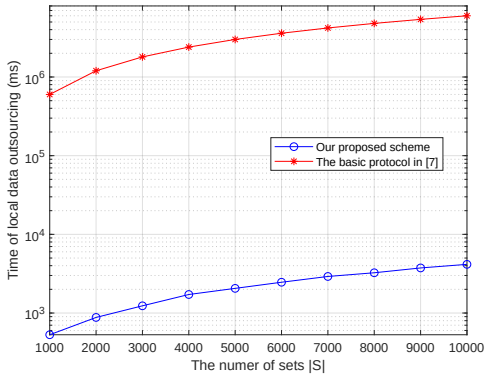

Fig. 4: The computational cost of local data outsourcing versus the number of sets $|\mathcal{S}|$, where $|\mathcal{E}| = 100$

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/TDSC.2020.3004442, IEEE Transactions on Dependable and Secure Computing

B. Set Similarity Search

1) The computational cost of query token generation: In the set similarity range search phase, the query user first generates the query token. In our scheme, the computational complexity of query token generation is $O(d^2)$, where d is the total number of unique elements in the dataset. In the basic protocol [11], the query set first transforms a query set to be a binary query vector in the form of $(v_1, v_2, \cdots, v_d)$ and then uses Paillier to encrypt each element of the query vector, i.e., $(\mathsf{PEnc}(v_1), \mathsf{PEnc}(v_2), \cdots, \mathsf{PEnc}(v_d))$. Thus, generating a query token in the basic protocol requires d Paillier encryption operations. Fig. 5 compares the computational cost of query token generation in our scheme and the basic protocol versus the parameter d. From this figure, we can see that the computational cost of query token generation in both our proposed scheme and the basic protocol increases with d, while our proposed scheme is much more efficient. For example, when $d = 100$, generating a query token in our proposed scheme just takes 0.01235 ms while that in basic protocol is 60000 ms. Thus, the query token generation in our proposed scheme is highly efficient and light weight.

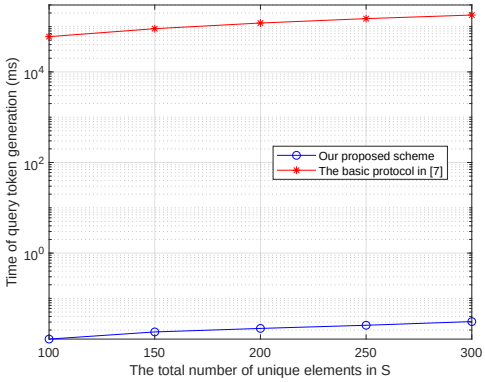

Fig. 5: The computational cost of query token generation versus d (the number of unique elements in $|\mathcal{S}|$)

2) The computational cost of query processing over the encrypted data in the cloud: After receiving the query token from the user, the cloud server searches the encrypted B+ tree stored in the cloud and returns the query result to the query user. The computational cost of the set similarity search is closely related to two parameters, i.e., the height of the B+ tree and the similarity threshold δ. In our experiment, the B+ tree is set to be 4-ary tree, so the height of the B+ tree will be related to $|\mathcal{S}|$, i.e., $\log_3 |\mathcal{S}|$. Fig. 6 plots the computational cost of set similarity search over the encrypted Jester dataset in the cloud varying with $|\mathcal{S}|$ and δ. From this figure, we can see that the computational cost of set similarity search increases as $|\mathcal{S}|$ and δ increase. This is because the increase of $|\mathcal{S}|$ will result in the increase of the height of the B+ tree, which is positively correlated to the computational cost of set similarity search. For instance, when $\delta = 0.6$, the computational cost of set similarity search over an encrypted B+ tree with $|\mathcal{S}| = 1000$ is only 7.52 ms, while that over an encrypted B+ tree with $|\mathcal{S}| = 10000$ is about 54.04 ms. For the δ, when it increases, the number of children nodes that can be pruned in the encrypted B+ tree probably increase. Thus, the computational cost of set similarity search decreases as the δ increases. For example, when $|\mathcal{S}| = 5000$, the computational cost of set similarity for $\delta = 0.5$ is about 31.57 ms, while that for $\delta = 0.9$ is only 5.48 ms.

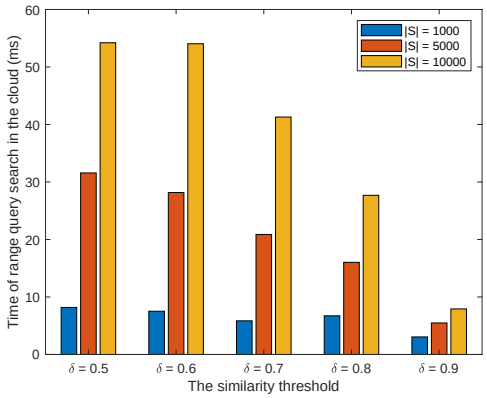

Fig. 6: The computational cost of set similarity search in the cloud versus $|\mathcal{S}|$ and the similarity threshold δ

Then, we compare our proposed scheme with the basic protocol [11] in terms of the query efficiency in the cloud. The basic protocol is initially proposed for the secure kNN query by scanning each encrypted data record and the computational cost of computing the similarity between two encrypted data is about d decryption operations and d encryption operations. In specific, given the ciphertexts of vectors $\mathbf{a}$ and $\mathbf{q}$, i.e., $\mathsf{E}(\mathbf{a}) = (\mathsf{PEnc}(a_1), \mathsf{PEnc}(a_2), \cdots, \mathsf{PEnc}(a_d))$ and $\mathsf{E}(\mathbf{q}) = (\mathsf{PEnc}(v_1), \mathsf{PEnc}(v_2), \cdots, \mathsf{PEnc}(v_d))$, the Jaccard similarity between $\mathbf{a}$ and $\mathbf{q}$ is $J(\mathbf{a}, \mathbf{q}) = \frac{\mathbf{a} \circ \mathbf{q}}{|\mathbf{a}|+|\mathbf{q}|-\mathbf{a} \circ \mathbf{q}}$. With the encrypted data $\mathsf{E}(\mathbf{a})$ and $\mathsf{E}(\mathbf{q})$, $\mathbf{a} \circ \mathbf{q}$ can be computed as $\prod_{i=1}^{d} \mathsf{PEnc}(a_i * v_i)$ and $|\mathbf{a}| + |\mathbf{q}| - \mathbf{a} \circ \mathbf{q}$ can be computed as $\prod_{i=1}^{d} \mathsf{PEnc}(a_i) + \prod_{i=1}^{d} \mathsf{PEnc}(v_i) - \prod_{i=1}^{d} \mathsf{PEnc}(a_i * v_i)$ as described in [11]. At the same time, the main computational cost is to compute $\prod_{i=1}^{d} \mathsf{PEnc}(a_i * v_i)$, which requires d encryption operations and d decryption operations. After computing the similarity between each data with the query data, the cloud requires k decryption operations to return the result to the query user. Thus, the total computational cost of the basic protocol is $(d \times |\mathcal{S}|)$ encryption operations and $(d \times |\mathcal{S}| + k)$ decryption operations. During the process of query, the computational cost of the basic protocol increases with the parameter k increases. Thus, the least computational cost is $(d \times |\mathcal{S}|)$ encryption operations and $(d \times |\mathcal{S}| + 1)$ decryption operations, i.e., $k = 1$.

At the same time, in our scheme, the computational cost of similarity query increases as the similarity threshold δ decreases. When $\delta = 0.5$, the computational cost is larger than that when $\delta = 0.6, 0.7, 0.8, 0.9$. In order to validate the query efficiency of our proposed scheme, we compare the least computation cost of the basic protocol when $k = 1$ with that of our proposed scheme when $\delta = 0.5$ versus the number of data records in the cloud (i.e., $|\mathcal{S}|$), as shown in Fig. 7. From this figure, we can see that out proposed scheme is far more efficient than the basic protocol in terms of query processing.

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/TDSC.2020.3004442, IEEE Transactions on Dependable and Secure Computing

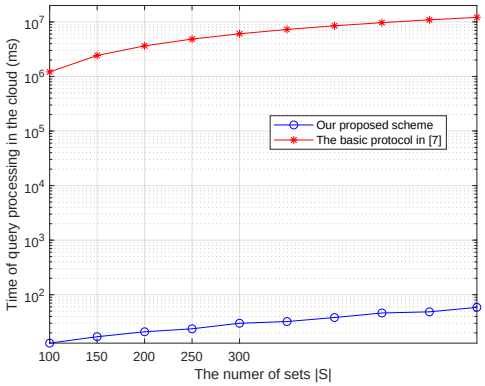

Fig. 7: The computational cost of query processing in the cloud versus the number of sets $|\mathcal{S}|$, where $d = |\mathcal{E}| = 100$

3) The computational cost of query result recovery at the query user side: In our proposed scheme, the final step of the set similarity search is that the query user uses his/her access key M_1 to recover the plaintext of the query result. The basic operation during the process of decryption is a multiplication operation between a matrix and a vector. At the same time, the computational cost in this step is closely related to the number of sets in the query result and also the size of each set. In our scheme, we evaluated the average computational cost of recovering one plaintext set from the query result, and it is about 0.24 ms, which is pretty efficient. For the basic protocol, the query result recovery is the minus operation over plaintexts, so it is also efficient.

VII. Related Work

The set similarity search is to search a collection of sets to find out the similar sets with a given query set, and there are two types of set similarity search, i.e., (i) top-k set similarity search, which returns k sets that are most similar to the current query set; (ii) set similarity range search with a similarity threshold δ, which returns the sets whose similarity to the query set is equal to or greater than δ. In the literature, the set similarity search is widely studied and many proposals are proposed [8]–[10], [28], [29]. Especially, Zhang et al. [8] proposed a B+ tree based index to achieve the set similarity, which is very efficient. In this paper, we use a similar index structure to represent the set records. However, different from this work, we focus on the privacy preservation property. Same as the work in [8], most of existing set similarity search works focus on the set similarity search over the plaintext domain.

In 2014, Blundo et al. [30] used private set intersection cardinality protocol to design a secure set similarity computation protocol and it can be deployed to achieve secure set similarity search. However, the proposed scheme is a two-party computation protocol and is not suitable for the data outsourcing scenario. For other works, most of them focus on the similarity query over fixed dimensional data point records. Actually, the set similarity search can be achieved by similarity query when each set record is transformed to be a fixed dimensional data point record. As for the similarity query, Elmehdwi et al. [11] and Rane et al. in [12] respectively designed a privacy-preserving set similarity search scheme, both of which are based on the homomorphic encryption technique and achieve similarity query by scanning each data record. Although, the proposed schemes can achieve strong security, but they are computationally inefficient due to the homomorphic encryption and linear scan. Yiu et al. [13] and Liu et al. [14] respectively introduced a privacy-preserving set similarity search over outsourced data. However, the security of these two schemes are too weak and unacceptable in some scenarios due to the adoption of the order preserving encryption technique. In these two schemes, the order information of the plaintext data will be leaked once the data records are outsourced to the cloud.

In order to balance the security and efficiency of the similarity search, the SSE and LSH based similarity search were proposed in [8], [15]–[17]. The main idea of such schemes is to first map each data record to a hash value by the locality-sensitive hash function, regard the hash values as the "keywords", and use SSE schemes to build index for these keywords. Then, the similarity query can be achieved by the keyword query of the SSE schemes. However, since the idea of locality-sensitive hashing is to map similar data into a same LSH value, it can just provide approximate set similarity search rather than exact one. To achieve exact SSE and LSH based similarity search schemes, Zheng et al. [18] and Cui et al. [19] introduced Yao's garbled circuits [20] to filter the query result. Although the similarity search schemes [18], [19] can return the exact query result, they have another flaw, i.e., they were built under the model of two cloud servers. However, in order to reduce the cloud bills, the data owners may prefer to deploy a single cloud server in some real scenarios.

VIII. Conclusion

In this paper, we have proposed an efficient and privacy-preserving exact set similarity search scheme under a single cloud server. First, we deployed ASPE technique to design a symmetric-key predicate encryption scheme, which can determine whether the upper bound of the similarity between a query vector and a collection of vectors is equal to or greater than a similarity threshold. Then, we represented the set records to be binary vectors and employed the B+ tree to build an index for them. Finally, based on the SPE-Sim scheme and B+ tree based index, we proposed an efficient and privacy-preserving set similarity search scheme, which can achieve efficient set similarity search while preserving the privacy of set records in the database, query set as well as the similarity threshold. In our future work, we will (i) take the set similarity search based on other similarity metric, such as cosine similarity, into consideration; (ii) consider other privacy-preserving methods to improve the set similarity search efficiency.

Acknowledgements

This research was supported in part by NSERC Discovery Grants (04009), ZJNSF LZ18F020003, NSFC U1709217, National Key Research and Development Program of China (2017YFB0802200), National Natural Science Foundation of

China (61972304), and Natural Science Foundation of Shaanxi Province (2019ZDLGY12-02).

REFERENCES

[1] Y. Lee, W. Hsiao, Y. Lin, and S. T. Chou, "Privacy-preserving data analytics in cloud-based smart home with community hierarchy," *IEEE Trans. Consumer Electronics*, vol. 63, no. 2, pp. 200–207, 2017.

[2] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-nn query for outsourced ehealthcare data," *J. Medical Systems*, vol. 43, no. 5, pp. 123:1–123:13, 2019.

[3] Q. Kong, R. Lu, M. Ma, and H. Bao, "A privacy-preserving sensory data sharing scheme in internet of vehicles," *Future Generation Comp. Syst.*, vol. 92, pp. 644–655, 2019.

[4] "Data-sharing and cloud: A big data match made in heaven," https://www.computerweekly.com/blog/Ahead-in-the-Clouds.

[5] S. Khandelwal, "Download: 68 million hacked dropbox accounts are just a click away!" https://thehackernews.com/2016/10/dropbox-password-hack.html, 2016.

[6] 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.

[7] D. Amagata, T. Hara, and C. Xiao, "Dynamic set knn self-join," in *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, 2019, pp. 818–829.

[8] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan, "An efficient framework for exact set similarity search using tree structure indexes," in *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, 2017, pp. 759–770.

[9] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, 2007, pp. 131–140.

[10] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, 2006, pp. 918–929.

[11] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, 2014, pp. 664–675.

[12] S. Rane and P. T. Boufounos, "Privacy-preserving nearest neighbor methods: Comparing signals without revealing them," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 18–28, 2013.

[13] M. L. Yiu, I. Assent, C. S. Jensen, and P. Kalnis, "Outsourced similarity search on metric data assets," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 338–352, 2012.

[14] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, 2013, pp. 733–744.

[15] X. Yuan, H. Cui, X. Wang, and C. Wang, "Enabling privacy-assured similarity retrieval over millions of encrypted records," in *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part II*, 2015, pp. 40–60.

[16] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, 2012, pp. 1156–1167.

[17] X. Yuan, X. Wang, C. Wang, C. Yu, and S. Nutanong, "Privacy-preserving similarity joins over encrypted data," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 11, pp. 2763–2775, 2017.

[18] Y. Zheng, H. Cui, C. Wang, and J. Zhou, "Privacy-preserving image denoising from external cloud databases," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 6, pp. 1285–1298, 2017.

[19] H. Cui, X. Yuan, Y. Zheng, and W. Cong, "Towards encrypted in-network storage services with secure near-duplicate detection," *IEEE Transactions on Services Computing*, 2018.

[20] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, 1986, pp. 162–167.

[21] 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.

[22] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, 2009, pp. 457–473.

[23] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, 2007, pp. 535–554.

[24] 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 *ACM SIGSAC*, 2018, pp. 745–762.

[25] D. P. Mehta and S. Sahni, Eds., *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004.

[26] R. Amin, S. H. Islam, N. Kumar, and K. R. Choo, "An untraceable and anonymous password authentication protocol for heterogeneous wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 104, pp. 133–144, 2018.

[27] "Jeter dataset," http://eigentaste.berkeley.edu/dataset/.

[28] J. Kim and H. Lee, "Efficient exact similarity searches using multiple token orderings," in *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, 2012, pp. 822–833.

[29] D. Deng, G. Li, H. Wen, and J. Feng, "An efficient partition based method for exact set similarity joins," *PVLDB*, vol. 9, no. 4, pp. 360–371, 2015.

[30] C. Blundo, E. D. Cristofaro, and P. Gasti, "Espresso: Efficient privacy-preserving evaluation of sample set similarity," *Journal of Computer Security*, vol. 22, no. 3, pp. 355–381, 2014.

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.

Rongxing Lu (S'09-M'11-SM'15) is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. He is presently a senior member of IEEE Communications Society. 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 currently 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 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, Xian, 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.