Similarity Search for Encrypted Images in Secure Cloud Computing

Yingying Li, Jianfeng Ma, Yinbin Miao, Yue Wang, Ximeng Liu, and Kim-Kwang Raymond Choo, *Senior Member, IEEE,*

Abstract—With the emergence of intelligent terminals, the Content-Based Image Retrieval (CBIR) technique has attracted much attention from many areas (i.e., cloud computing, social networking services, etc.). Although existing privacy-preserving CBIR schemes can guarantee image privacy while supporting image retrieval, these schemes still have inherent defects (i.e., low search accuracy, low search efficiency, key leakage, etc.). To address these challenging issues, in this paper we provide a similarity Search for Encrypted Images in secure cloud computing (called SEI). First, the feature descriptors extracted by the Convolutional Neural Network (CNN) model are used to improve search accuracy. Next, an encrypted hierarchical index tree by using K-means clustering based on Affinity Propagation (AP) clustering is devised, which can improve search efficiency. Then, a limited key-leakage k-Nearest Neighbor (kNN) algorithm is proposed to protect key from being completely leaked to untrusted image users. Finally, SEI is extended to further prevent image users' search information from being exposed to the cloud server. Our formal security analysis proves that SEI can protect image privacy as well as key privacy. Our empirical experiments using a real-world dataset illustrate the higher search accuracy and efficiency of SEI.

Index Terms—Content-based image retrieval, hierarchical index tree, k-Nearest Neighbor, search accuracy, search efficiency.

1 Introduction

With the rapid development and popularization of cloud computing, people enjoy various conveniences brought by cloud services, such as storing images on the cloud. However, directly outsourcing images to the public cloud inevitably raises privacy concerns. Once the massive images (e.g., patients' medical images) containing highly sensitive information have been leaked to unauthorized entities, it will incur serious consequences or unnecessary trouble. The encryption mechanism can alleviate image data security and privacy concerns to some extent, but it invalidates the Content-Based Image Retrieval (CBIR) technique over ciphertext, and even causes other concerns discussed in the following example.

Example. Alice outsources the encrypted images $\mathcal{C}$ of the local image database $\mathcal{M}$ to the cloud server. The authenticated Bob generates an encrypted search request $\mathcal{T}$ according to query image m_q by using the searchable key sk shared by Alice when he queries images similar to m_q. Then, the cloud server searches $\mathcal{C}$ after receiving $\mathcal{T}$ and returns relevant search results $\mathcal{R}$ to Bob. Finally, Bob decrypts the $\mathcal{R}$ with image encryption key k_{ie} from Alice.

- *Y. Li, J. Ma, Y. Miao (Corresponding author), and Y. Wang are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China; State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China. E-mail: lyylyingying@163.com, jfma@mail.xidian.edu.cn, ybmiao@xidian.edu.cn, ywang_86@stu.xidian.edu.cn.*
- *X. Liu is with the Key Laboratory of Information Security of Network Systems, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China. E-mail: snbnix@gmail.com.*
- *K.-K. R. Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA. E-mail: raymond.choo@fulbrightmail.org.*

In this encrypted image search process, the cloud server is regarded as semi-trusted. Alice stores her images on this cloud server, and entrusts the cloud server to perform similarity search tasks. When Bob comes to query Alice's images, the cloud server will honestly provide Bob with search service under Alice's arrangement. The performance of search services, such as search accuracy and efficiency, will profoundly affect Bob's search experience. Assuming that Bob is a doctor who relies on search results to diagnose a certain patient's condition, the incorrect search results will lead to a wrong diagnose, which endangers the patient's health and even life. Besides, the time-consuming search process will prolong the waiting time of the Bob. If Bob is a mobile image user who requires high real-time responses, such a long search time is hard to bear and easily makes search time lose timeliness. Moreover, the above search mechanism still requires Alice to share sk and k_{ie} with Bob, which cannot completely protect image privacy. This is because we cannot promise that Bob is fully trusted and does not share sk and k_{ie} with other unauthorized image users due to interest incentives in practice.

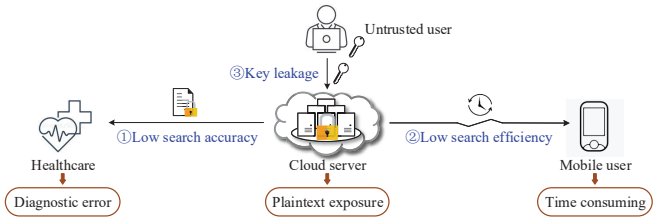


Fig. 1: Challenging issues in privacy-preserving CBIR.

Fortunately, various schemes related to privacy-preserving CBIR have been studied, like [1]–[8]. However,

in practice, these schemes still face many challenges (i.e., low search accuracy, low search efficiency, key leakage, etc.). Specifically, schemes [1], [2], [3], [5] directly distributed keys to users, leading to the risk of image users leaking keys, schemes [3], [5] sacrificed accuracy to improve efficiency, and schemes [1], [4], [6], [7], [8] brought a lot of overhead to achieve high security. These challenges limiting their practical applications are shown in Fig. 1 (namely: ① low search accuracy; ② low search efficiency; ③ key leakage).

As for the first challenge indicated by ① in Fig. 1, there are two primary reasons affecting the search accuracy. One is the feature descriptors type, the other is the similarity calculation method. As for the former, image feature descriptors are mainly classified into global features (e.g., Local Binary Pattern (LBP), Histogram of Oriented Gradients (HOG), etc.) and local features (e.g., Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), etc.). The local features can achieve higher search accuracy than global features due to the better robustness. There are also works [9], [10] that combine global features and local features with certain weights to form new features or apply Convolutional Neural Network (CNN) model mimicking human visual cognition to extract feature vectors, which achieve acceptable accuracy. For the later, the similarity between images is measured by Euclidean distance, Cosine distance, Hamming distance, and Jaccard similarity coefficient. Especially, Asymmetric Scalar-product-Preserving Encryption (ASPE) algorithm [11] using random numbers and matrices to encrypt feature vectors can calculate the Euclidean distance of high-dimension space more accurately. At the same time, other works [4], [12] using Secure Multi-party Computation (SMC), Homomorphic Encryption (HE) to calculate Euclidean distance can also improve search accuracy, but result in inefficiency due to complex garbled circuit operations.

In order to support the large-scale image search in actual applications, it is indispensable to improve the *search efficiency* (see ② in Fig. 1). As far as we know, a feasible search scheme can be achieved by constructing the efficient linear index, inverted index, hash table or tree index. In particular, the hash table and tree index perform better than the other two regarding search efficiency. For example, the two-level hash table [3] constructed based on the Locality Sensitive Hashing (LSH) algorithm greatly speeds up the search process by reducing the dimension of the high-dimension feature vectors. For tree index, scheme [10] classifies images by using K-means clustering algorithm and then builds the index tree based on clustering results, which narrows the search scope and avoids traversing the entire image database during the search phase.

Although some prevalent schemes based on ASPE and index tree achieve high search efficiency without deteriorating search accuracy, they do not solve the *key leakage* challenge (see ③ in Fig. 1). Like [2], [10], [13], the image owner should share sk and k_{ie} with all authorized image users. However, it is impractical and overloaded to establish a secure channel to transmit these keys. To avoid key transmission, the latest scheme [14] focusing on improving the ASPE algorithm can guarantee that sk used to encrypt vectors will not be completely disclosed to any image user. Unfortunately, it does not support image search. Although some image search solutions [4], [15] can avoid leaking sk, these solutions still have low search accuracy and search efficiency due to the use of homomorphic encryption and secure multi-party computation.

As far as we know, no existing work is dedicated to solving the above three challenges simultaneously. Hence, in this paper we propose a similarity Search for Encrypted Images in secure cloud computing (SEI) to solve the above challenges. Specifically, we employ the CNN model to extract feature vectors to improve search accuracy, and then build a hierarchical index tree in a bottom-up manner based on the clustering algorithm to improve search efficiency. Besides, we design an optimized ASPE algorithm, which does not require the image owner to share sk with image users, to achieve limited key-leakage for untrusted image users. To summarize, our contributions are set out as follows.

- **High search accuracy.** SEI achieves a high search accuracy by using the pre-trained CNN model to extract feature vectors. The CNN model simulating the human visual perception process can more accurately represent the image content, which makes the similarity measurement between images more accurate and search results more accurate.
- **High search efficiency.** SEI uses the optimized K-means clustering algorithm to classify images and constructs a hierarchical index tree in a bottom-up manner based on the clustering results. Therefore, SEI avoids traversing the entire image database when performing search operation and reduces the search time to sublinear time. Moreover, the Principal Component Analysis (PCA) algorithm is utilized to lower the dimensions of the image feature vectors extracted by CNN model, which further improves search efficiency.
- **Limited key leakage.** SEI provides a secure trapdoor generation process with limited key-leakage[1], which not only prevents untrusted image users from completely leaking keys privacy but also avoids the online help of the image owner when the image user generates trapdoors locally. Moreover, even if any unauthenticated image user illegally obtains partial information about searchable keys from a certain untrusted user, he cannot generate a valid query trapdoor or decrypt the ciphertext of the image owner.

The remainder of this paper is organized as follows. Section 2 reviews existing work on encrypted image retrieval. Then, Section 3 discusses relevant knowledge materials and Section 4 formulates the problem including system model, threat model, etc. Later on, the concrete construction of proposed scheme SEI is introduced in Section 5, followed by Section 6 which outlines the security and performance analysis. Finally, we conclude our work in Section 7.

2 Related Work

Since Song *et al.* [16] first proposed the notion of SE, more and more SE schemes enriched with different features have

1. It must be admitted that extra interactions between image owner and image users will be introduced in the key generation phase to implement a secure trapdoor generation process. In other words, SEI can provide stronger security by sacrificing some communication overhead.

been presented [17]–[20]. Most of them are related to textual data search based on a single keyword, multiple keywords or fuzzy keywords. There are few schemes for privacy-preserving CBIR.

Lu *et al.* [21] proposed the first privacy-preserving CBIR scheme based on order-preserving encryption and min-hash techniques, but this scheme only supports visual feature word description and has lower search accuracy when compared with CBIR schemes using Fish vector [22]. To improve search accuracy, Huang *et al.* [23] developed a new private feedback method based on the K-anonymity principle and Support Vector Machine (SVM) technique to further improve search accuracy by constantly correcting the returned search results. Likewise, Hazra *et al.* [24] designed a secure image retrieval system that used HSV histogram as image features and combined k-Nearest Neighbor (kNN) with SVM to search similar images, which achieved high search accuracy. Recently, Xia *et al.* [25] presented a privacy-preserving CBIR scheme by designing a novel Bag-Of-Encrypted-Words (BOEW) model. This scheme extracted features on encrypted images and achieved high search accuracy.

Although the search accuracy was improved in these schemes, the search efficiency was not optimal. Hu *et al.* [26] introduced a scheme with a tree-based data structure and ASPE, which implemented faster search than linear search. However, the use of homomorphic encryption makes this scheme impractical. As a result, Li *et al.* [2] designed a novel data structure called sub-simhash, which is suitable for the inverted index, to weaken the efficiency impact caused by homomorphic calculations. Besides, Abduljabbar *et al.* [27] constructed a searchable index by using the Locality Sensitive Hash (LSH), thereby increasing the proficiency and effectiveness of the system. According to LSH, Xia *et al.* [3] also presented an efficient and privacy-preserving scheme with high search efficiency. However, the extracted MPEG-7 feature descriptors result in lower accuracy in this scheme.

Nevertheless, the aforementioned various encrypted image search schemes cannot prevent untrusted image users from revealing key privacy. Zhu *et al.* [28] proposed a system to support the kNN query without sharing keys among authorized image users by using the additive homomorphic property of Paillier encryption system. Also, Nair *et al.* [29] presented a searchable encryption scheme with eliminating the need for key sharing based on the bilinear pairing property and Diffie-Hellman key exchange protocol. Unfortunately, these schemes do not support image search. For image retrieval, Zhang *et al.* [15] employed a multi-level homomorphic encryption protocol to support key conversion between image owner and image users, which avoided the image owner's feature encryption key leakage. However, the method of encrypting each dimension feature vector with homomorphism does not apply to high-dimension feature vectors and large-scale image sets.

Compared with previously proposed schemes, SEI scheme has several advantages (i.e., high accuracy, high efficiency, unshared key, image owner offline etc.) shown in TABLE 1.

TABLE 1: Compare among various schemes

Schemes	Function 1	Function 2	Function 3	Function 4
[3]	MPEG-7	LSH	×	×
[7]	DCT	Linear	√	√
[10]	CNN	Tree	×	×
[13]	Fisher	Tree	×	×
[15]	SIFT	Linear	√	√
[28]	–	Linear	√	×
[29]	–	Linear	√	√
SEI	CNN	Tree	√	√

Notes. Function 1: Feature type; Function 2: Index structure; Function 3: Unshare the feature vectors encryption key; Function 4: Image owner offline when image users generate trapdoors.

3 PRELIMINARIES

We first present the basic operations of SEI scheme before we elaborate on it.

3.1 CAK-means Clustering

K-means clustering is widely used in image classification scenarios as a clustering analysis method, such as dividing n images into K clusters. Nevertheless, the K value and cluster centers are fixed and sensitive to its initial conditions. Compared with K-means clustering, Affinity Propagation (AP) clustering not only does not require the number K and centers of clusters before execution, but also can quickly produce a good set of cluster exemplars. Considering the convergence property of K-means and the outstanding performance of AP, the CAK-means (Combination of AP and K-means clustering) method is proposed by Zhu *et al.* [30] to achieve faster and more accurate searchable encryption. The specific process of the CAK-means is illustrated in Fig. 2.

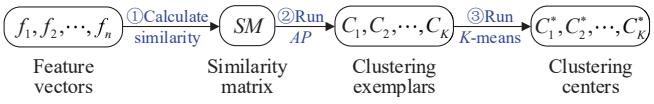


Fig. 2: The specific process of CAK-means.

As shown in Fig. 2 in step ①, for each feature vector f_i, f_j in $f = \{f_1, f_2, \cdots, f_n\}$ (n is the total of images), we calculate the similarity score between f_i and f_j according to

$$S_{f_i, f_j} = f_i \cdot f_j^{\top}. \tag{1}$$

Then, a $d \times d$ dimensional similarity matrix SM is available. Next, we take SM as input and run AP clustering algorithm to get cluster exemplars $\{C_1, C_2, \cdots, C_K\}$ as shown in step ②. Finally, given the number of cluster K and the initial centers $\{C_1, C_2, \cdots, C_K\}$, we run K-means clustering algorithm as shown in step ③ to get final cluster centers $\{C_1^*, C_2^*, \cdots, C_K^*\}$. In CAK-means algorithm, initializing K-means by using cluster exemplars produced by AP effectively improves the initial cluster centers stability of K-means. Obviously, CAK-means outperforms both AP and K-means clustering in aspect of minimizing total squared error and producing stable clustering results.

3.2 Hierarchical Index Tree Building

We build an index tree in a bottom-up manner so that the cloud server only needs to search the subsets of images

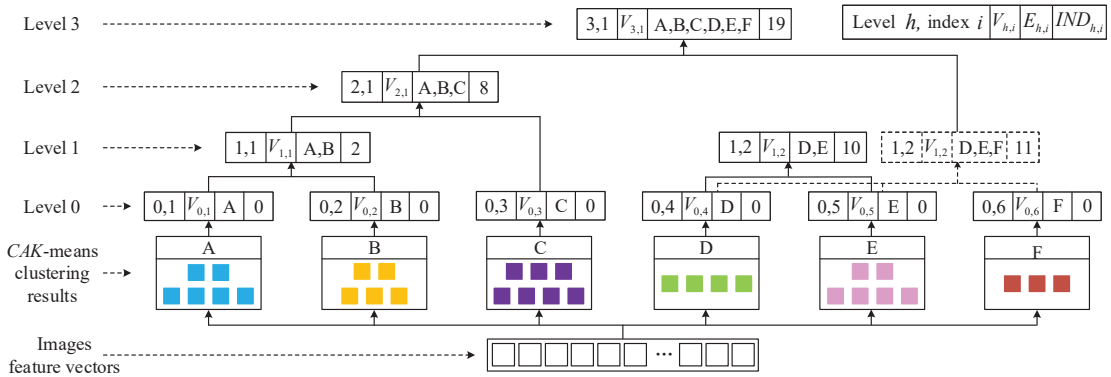


Fig. 3: The construction process of hierarchical index tree.

instead of the entire image database, while saving the retrieval time. The main idea is to group similar images into a cluster and similar clusters into higher ones. We use $C_{h,i} = \{V_{h,i}, E_{h,i}, IND_{h,i}\}$ to represent a cluster, where h is the level at which the cluster is located and i is the index at the level h. The first component $V_{h,i}$ represents the vector of the current cluster, the second component $E_{h,i} = \{e_1, e_2, \cdots, e_{|E_{h,i}|}\}$ represents a set of children nodes of the current cluster node, and the third component $IND_{h,i}$ represents the maximum distance between all children nodes in this cluster. After n images are divided into K clusters by CAK-means clustering, we regard K cluster centers as K leaf nodes of index tree, denoted as $C_{0,1}, C_{0,2}, \cdots, C_{0,K}$, where $C_{0,i} = \{C_i^*, E_{0,i}, 0\}$. According to the vector stored in the leaf nodes, we calculate the Euclidean distance between the leaf nodes. The similar clusters with minimum distance are merged into a new cluster, that is, a parent node. Finally, a hierarchical index tree is constructed iteratively until all leaf nodes are clustered into one root node.

In order to explain the index tree construction process more clearly, we give an example as shown in Fig. 3. First, we assume that all images are clustered into 6 groups using the CAK-means clustering algorithm. Therefore, as shown in Fig. 3 level 0, the clusters represented by leaf nodes can be denoted as $C_{0,1}, C_{0,2}, \cdots, C_{0,6}$. Next, we merge similar clusters into a new cluster (shown in Fig. 3 level 1), e.g., clusters $C_{0,1}$ and $C_{0,2}$ are merged into a new cluster $C_{1,1}$, where vector $V_{1,1}$ of $C_{1,1}$ is $\frac{V_{0,1}+V_{0,2}}{2}$, child nodes $E_{1,1}$ are $\{A, B\}$, maximum distance between child nodes is $||V_{0,1} - V_{0,2}||_2 = 2$ ($||\cdot||_2$ means 2-norm). Then, clusters $C_{0,4}$ and $C_{0,5}$ are merged into a new cluster $C_{1,2}$ in the same way. Suppose clusters $C_{0,3}$ and $C_{1,1}$ are closest and their distance is 8. Thus, $DM = \frac{|D(C_{0,3},C_{1,1}) - max(IND_{0,3},IND_{1,1})|}{max(IND_{0,3},IND_{1,1})} = \frac{8-2}{2} = 3 > 0.5$ ($|\cdot|$ means absolute value, $max(\cdot,\cdot)$ means maximum value, Th_t is 0.5), so we merge $C_{0,3}$ and $C_{1,1}$ upwards into a new cluster $C_{2,1}$ (shown in Fig. 3 level 2). After that, suppose cluster $C_{0,6}$ and cluster $C_{1,2}$ are closest and their distance is 11. Thus, $DM = \frac{|D(C_{0,6},C_{1,2}) - max(IND_{0,6},IND_{1,2})|}{max(IND_{0,6},IND_{1,2})} = \frac{11-10}{10} = 0.1 < 0.5$, so we merge $C_{0,6}$ into the cluster $C_{1,2}$ and update the $V_{1,2}$ of $C_{1,2}$ to $\frac{V_{0,4}+V_{0,5}+V_{0,6}}{3}$ (shown in Fig. 3 level 1). We iteratively perform the merge process until all clusters are merged into one cluster (shown in Fig. 3 level 3).

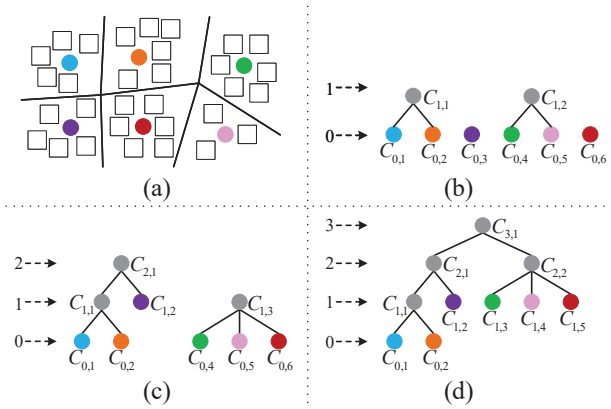


Fig. 4: The nodes variation during the construction of the hierarchical index tree.

Compared with Fig. 3, Fig. 4 details the nodes variation during the construction of the hierarchical index tree. In Fig. 4, (a) indicates that the CAK-means clustering algorithm divides the images into 6 clusters; (b) indicates that $C_{0,1}$ and $C_{0,2}$ are similar and are merged upwards into a new cluster $C_{1,1}$, $C_{0,4}$ and $C_{0,5}$ are merged into $C_{1,2}$ in the same way; (c) indicates that $C_{1,1}$ and $C_{0,3}$ are similar and are merged upward into a new cluster $C_{2,1}$, while $C_{0,5}$ is merged into $C_{1,2}$ because the DM values calculated by the clusters $C_{0,5}$ and $C_{1,2}$ are less than the threshold Th_t; (d) indicates that $C_{2,1}$ and $C_{2,2}$ are merged into $C_{3,1}$ as the root node of the entire index tree. Thus, the hierarchical index tree is formed.

4 Problem Formulations

In this section, we will depict the system model, threat model and design goals, respectively.

4.1 System Model

The system model of SEI scheme mainly contains three entities, the image owner, the image user, and the cloud server. Specifically, the image owner is responsible for uploading the encrypted images and the encrypted index, and the image user generates and sends a query request to the cloud server, and the cloud server is in charge of

retrieving the result of the query request in the encrypted images according to the index. The cooperation among the three entities is illustrated in Fig. 5. At first, in the step ①, to prevent the image user from leaking the encryption key of feature vector, the image owner and the image user cooperate to complete the key generation, so that each image user obtains a different key. Then, in the step ②, the image owner outsources the encrypted images and index to the cloud server. Apart from that, the image user who has been authenticated from image owner uses the feature vector encryption key to encrypt the feature vector of the query image to generate a query trapdoor for similarity search and submits it to the cloud server in the step ③. Upon receiving a valid request from the image user, the cloud server retrieves the encrypted index and images, returning the top k images that most likely match the image user's query in the step ④. Our system aims to protect images and keys from being leaked to the cloud server or malicious third-party entities while improving search accuracy and efficiency. The specific role of each entity is described as follows:

- `Image owner`. The image owner owns an image database with n images: $\mathcal{M} = \{m_1, m_2, \cdots, m_n\}$. Before uploading images to the cloud server, the image owner encrypts each image m_i into c_i and extracts feature vectors to build a secure index tree $\mathcal{I}$. The image owner submits the encrypted images $\mathcal{C}$ and the corresponding index tree $\mathcal{I}$ to the cloud server. Besides, the image owner assists image users to generate the key used to generate query trapdoor.
- `Image user`. An authorized image user first selects an interesting image m_q, then generates the corresponding trapdoor locally, and submits the trapdoor to the cloud server. After receiving the search results returned from the cloud server, the image user decrypts results with the key.
- `Cloud server`. The cloud server with unlimited storage space and computation resources performs image retrieval and returns top k similar images as search results to the image user.

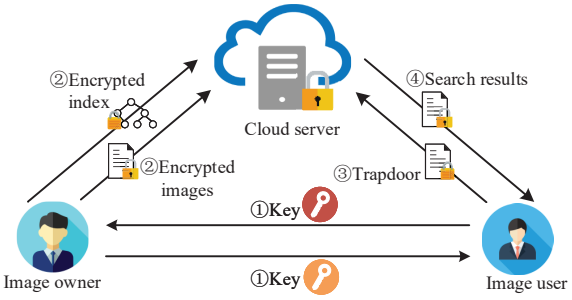


Fig. 5: System model of our scheme.

4.2 Threat Model

In our scheme, the cloud server is considered as an "honest-but-curious" entity. Specifically, the cloud server will honestly execute the protocol, but also curiously analyze the data and index tree to capture more information associated with plaintext images. Based on the available information to the cloud server, the following two threat models are concluded.

—`Known ciphertext model`. In this model, the cloud server can obtain encrypted images, index, and query trapdoors from image owner and image users. Meanwhile, the search results returned to image users are available to the cloud server.

—`Known background model`. In this stronger model, the cloud server learns about more information than what is accessible in `known ciphertext model`, such as some background information related to image users. The cloud server collects image users' activity information on other websites to help guess image types that image users might be interested in, and then analyzes the query purpose of image users even infers plaintext and ciphertext pairs based on a large number of query requests and corresponding search results.

Concerning image users, most SE schemes assume that image users are fully trusted. However, in SEI, image users may collude with the semi-trusted cloud server, such as leaking encryption keys to the cloud server. Besides, it is necessary to mention that the cloud server can infer privacy information based on the access pattern. For security concerns, SEI can utilize the ORAM [31] technology to address this issue, but the high communication and storage overhead can drastically lower the efficiency of the actual application. This is a trade-off between security and efficiency. Our work focuses on ensuring low overhead and high security in a simplicity and effective way, so we will not consider ORAM technology.

4.3 Design Goals

To solve the threat models we have proposed earlier, we design an encrypted image retrieval system that not only allows the image user to accurately and efficiently search similar images, but also protects the privacy of image owner and image user from the cloud server. SEI is designed with the following goals in mind:

- `Search accuracy`. For the sake of ensuring that search results meet the user's similarity requirements, SEI scheme should achieve high retrieval accuracy.
- `Search efficiency`. To adapt to the large scale image retrieval scenario, the search time complexity of SEI scheme should be as low as possible without deteriorating search accuracy.
- `Privacy requirements`. While ensuring high search accuracy and efficiency, SEI should also meet the following privacy requirements.
  1) `Image privacy`. The image privacy should be well protected so that the cloud server would not know the plaintext information from the stored encrypted images.
  2) `Index privacy`. To guarantee the confidentiality of the index tree, SEI should prevent the cloud server from relying on the index tree to gain valid knowledge in addition to the access pattern and search pattern.
  3) `Trapdoor unlinkability`. To prevent the cloud server from deducing sensitive information accord-

ing to the relationship among trapdoors, SEI should ensure that the cloud server cannot distinguish differences between trapdoors, even from the same query.

5 Similarity Search for Encrypted Images in Secure Cloud Computing Scheme

Consider that many image owners have a large amount of images, but they possess limited computation and storage resources. For privacy protection, images are encrypted before being outsourced to the cloud server. After that, the image user selects an interesting image and generates the corresponding trapdoor locally, then sends an encrypted search request (called trapdoor) to the cloud server. Finally, the cloud server retrieves the outsourced data and returns the top k similar images to the image user. Accuracy, efficiency and security are crucial throughout the process. However, the traditional solutions cannot provide high-accuracy and high-efficiency retrieval services, and even expose completely the searchable keys to image users. To this end, we propose a similarity Search for Encrypted Images in secure cloud computing (SEI) scheme. First, ideas on solving these problems are put forward as follows.

- To ensure that the search results are similar, we attempt to utilize the pre-trained CNN model to extract image features and use the *AP* clustering algorithm to overcome the instability of *K*-means clustering algorithm.
- Since it takes more time to retrieve similar images in the large-scale image database, taking advantage of the tree structure, we try to build a hierarchical index tree based on cluster centers so that the entire image database is not retrieved.
- To avoid the trouble that keys are completely leaked by untrusted image users, we can construct a limited key-leakage mechanism based on the kNN algorithm to support that untrusted image users generate trapdoor locally without image owner online.

With these ideas in mind, a secure and feasible similarity search scheme for encrypted images was produced in our work and presented as follows. We first give some notations used in SEI scheme and then propose a brief description of the framework, finally demonstrate the concrete construction of SEI scheme. Compared with prior image search schemes, SEI can not only achieve high search accuracy and efficiency but also solve the key leakage problem caused by untrusted image users. However, SEI cannot prevent the untrusted cloud server from guessing the query purpose of image users based on the search results. Hence, in the extended version, we will address the issue of untrusted cloud server obtaining private information from search results.

5.1 Notations

In this paper, notation definitions introduced in TABLE 2 are used.

TABLE 2: Notations

Notations	Descriptions
$\mathcal{M}, \mathcal{C}$	Plaintext/ciphertext image database
f, d	Extracted feature vector and its dimension
f'	Encrypted feature vector
r, S	Random vectors for image owner
$M, \boldsymbol{\pi}$	Random matrix and permutation for image owner
θ, X	Random number and vector for image users
$\bar{A}, U$	Encryption keys of query feature vector
W	Random matrix from image user
k_{ie}	Image encryption key
$C_1, C_2, \cdots, C_K$	Cluster exemplars of AP clustering
$C_1^*, C_2^*, \cdots, C_K^*$	Cluster centers of CAK-means clustering
$\mathcal{I}$	Index tree for image database $\mathcal{M}$
$\mathcal{T}_q$	Trapdoor for query image m_q
$\mathcal{R}, \mathcal{R}'$	Ciphertext/plaintext of search results
k	The total number of images in search results

5.2 Framework of SEI Scheme

SEI framework is a tuple of several algorithms, namely **GenKey**, **EncImage**, **GenIndex**, **GenTrap**, **Search** and **DecImage**, which are depicted in Fig. 6. In the step ①-③, the image owner first generates keys and assigns the key to the image user, then encrypts images in a traditional encryption method and builds the corresponding encrypted index, finally sends encrypted images and index to the cloud server for storage. When an authenticated image user wants to issue the search query, he submits a trapdoor generated from the query image to the cloud server with the step ④. After gaining the search query, the cloud server performs the search operation and returns the correct search results to the image user, which is shown by the step ⑤. In the step ⑥, the image user decrypts the search results to obtain images similar to the query image. As for the specific process in different algorithms in SEI, we will give a brief definition as follows.

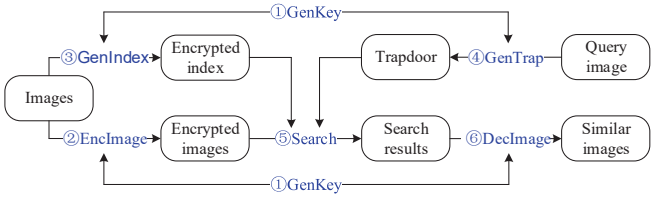


Fig. 6: Overview of SEI.

- **GenKey**$(1^\lambda) \rightarrow (sk, k_{ie}, \bar{A}, U, W)$. In this stage, the image owner creates his private key $sk = \{r, S, M, \boldsymbol{\pi}\}$ to encrypt outsourced images and feature vectors, then he computes $\bar{A}$ and U to the authorized image user according to W so that the image user can generate query request. W is a $d \times d$ invertible matrix randomly generated by the image user. Here, U is the same for all image users, while $\bar{A}$ and W are different, and each image user cannot learn the value of $\bar{A}$ and W of others.
- **EncImage**$(\mathcal{M}, k_{ie}) \rightarrow \mathcal{C}$. The image owner encrypts the each image in $\mathcal{M}$ with k_{ie} by a symmetric encryption algorithm which has semantic security, then sends encrypted images $\mathcal{C}$ to the cloud server.
- **GenIndex**$(\mathcal{M}, sk) \rightarrow I$. Depending on the feature vector extracted from each image in $\mathcal{M}$, the image owner generates encrypted index $\mathcal{I}$ which can help the cloud server speed up the search process.
- **GenTrap**$(m_q, \bar{A}, U, W) \rightarrow \mathcal{T}_q$. The image user encrypts the query vector extracted from query image m_q to generate trapdoor $\mathcal{T}_q$ for image search without the online

help of the image owner. Moreover, the image owner does not share searchable keys with images users.

- **Search**$(\mathcal{I}, \mathcal{T}_q) \rightarrow \mathcal{R}$. Upon receiving an encrypted trapdoor $\mathcal{T}_q$, the cloud server matches the index $\mathcal{I}$ and finds the top k images similar to the query image as search results $\mathcal{R}$, then returns $\mathcal{R}$ to the corresponding image user.
- **DecImage**$(\mathcal{R}, k_{ie}) \rightarrow \mathcal{R}'$. The image user decrypts search results $\mathcal{R}$ with the key k_{ie} to get plaintext images $\mathcal{R}'$.

5.3 Construction of SEI Scheme

The concrete functions of different components are described below.

5.3.1 Key generation

Given a secure parameter λ, the image owner first randomly generates a image encryption key k_{ie}, a $(2d+2) \times (2d+2)$ dimensional invertible matrix M, two $d+1$-dimensional vectors $r = (r_1, r_2, \cdots, r_{d+1})$ and $S = (S_1, S_2, \cdots, S_{d+1})$, and a permutation $\boldsymbol{\pi}$ of $d+1$ values. Then, the image owner arbitrarily selects a positive real number θ and a $d+1$-dimensional vector $X = (X_1, X_2, \cdots, X_{d+1})$. For each $i \in [1, 2d+2]$, the image owner computes the matrix M', where the i-th row M'_i of M' is

$$M'_i = \boldsymbol{\pi}^{-1}(\cdots, M_{i,2j-1} + M_{i,2j}, \cdots), j \in [1, d+1].$$

Next, the image owner calculates $(2d+2) \times d$ dimensional matrix A and $2d+2$-dimensional vector U as follows,

$$A_i = \theta \cdot (M'_{i,1}, M'_{i,2}, \cdots, M'_{i,d}),$$
$$U_i = \theta \cdot M'_{i,d+1} + \sum_{j=1}^{d+1} X_j M_{i,2j}.$$

Here, $M_{i,j}$ represents the i-th row, j-th column element of M, $M'_{i,j}$ denotes the j-th dimension of M'_i, and A_i denotes the i-th row of A. When an image user registers SEI system, he randomly generates a $d \times d$ dimensional matrix W and sends W to the image owner. The image owner calculates $\bar{A} = AW^{-1}$ and returns $\bar{A}$, U, k_{ie} to the image user.

5.3.2 Image encryption

Given the image database $\mathcal{M}$, the image owner encrypts $\mathcal{M}$ to $\mathcal{C}$ with the symmetric key k_{ie} as

$$\begin{aligned}\mathcal{C} &= Enc(\mathcal{M}, k_{ie}) \\ &= (Enc(m_1, k_{ie}), \cdots, Enc(m_n, k_{ie})) \\ &= (c_1, \cdots, c_n),\end{aligned}$$

and then sends encrypted images $\mathcal{C}$ to the cloud server for storage and search.

5.3.3 Index generation

The algorithm **GenIndex** is performed in two steps, the first step is to encrypt feature vectors, and the second step is to build an index tree.

`Step 1`: Suppose $r_{-(d+1)} = (r_1, r_2, \cdots, r_d)$, for feature vector $f_i = (f_{i,1}, f_{i,2}, \cdots, f_{i,d})$ extracted by the pre-trained CNN model from image m_i in $\mathcal{M}$, the image owner first computes

$$\tilde{f}_i = \boldsymbol{\pi}(r_{-(d+1)} - 2f_i, r_{d+1} + \|f_i\|^2). \quad (2)$$

Here, $\|f_i\|^2$ is the length of vector f_i. Then the image owner converts $d+1$-dimensional vector $\tilde{f}_i$ into a $2d+2$-dimensional vector $\hat{f}_i = (\hat{f}_{i,1}, \hat{f}_{i,2}, \cdots, \hat{f}_{i,2d+2})$, where $\hat{f}_{i,2j-1} = \tilde{f}_{i,j}$, $\hat{f}_{i,2j} = S_j$ for each $j \in [1, d+1]$. Finally, the image owner gets the encrypted feature vector

$$f'_i = \hat{f}_i M^{-1} \quad (3)$$

and sends $f' = \{f'_1, f'_2, \cdots, f'_n\}$ to the cloud server.

`Step 2`: To speed up the image search process, the image owner builds an index tree for all images, in which leaf nodes store cluster centers of encrypted image. For feature vectors extracted from images $\mathcal{M}$, the image owner calculates the number K of clusters and cluster centers $C = \{C_1, C_2, \cdots, C_K\}$ using the AP algorithm. Next, K and C are used to initialize the K-means algorithm. Finally, K-means algorithm outputs a new set of cluster centers $C^* = \{C_1^*, C_2^*, \cdots, C_K^*\}$. The index tree is built by the image owner in a bottom-up manner using the procedures described earlier (as shown in Section 3). $C_{0,1}, C_{0,2}, \cdots, C_{0,K}$, where $C_{0,i} = \{C_i^*, E_{0,i}, 0\}$, as the K leaf nodes of index tree. The image owner calculates the Euclidean distance between C_i^* and C_j^* $(i, j \in [1, K],\ i \neq j)$ and finds out the nodes with minimum distance to merged into a parent node. Then, for the remaining leaf nodes, two nodes with the smallest distance should be found according to the distance between the leaf nodes and the existing parent node. If the two nodes with the smallest distance are at the same level, then the two nodes are directly merged upward to generate a new parent node. If they are not at the same level, the owner calculates the DM value and determines whether the DM value is greater than the threshold. According to the DM value, the owner merges them to generate a new parent node or merges the low-level node into the high-level node. Until all the leaf nodes are merged into one node, a hierarchical index tree is formed. To protect the feature vector values stored in the nodes of the index tree, the image owner encrypts the vectors as shown in `Step 1`, and associates both the encrypted feature vectors f' and image IDs to each leaf node. Finally, a secure index tree $\mathcal{I}$ is built and sent to the cloud server by the image owner.

5.3.4 Trapdoor generation

When a certain image user wants to query similar images, he needs to generate a trapdoor $\mathcal{T}_q$ based on the query image m_q. The image user first extracts feature vector $f_q = (f_{q,1}, f_{q,2}, \cdots, f_{q,d})$ from query image m_q and then picks up a random number γ to encrypt f_q into f'_q by calculating

$$f'_q = \gamma \cdot (f_q \cdot W^\top \cdot \bar{A}^\top + U). \quad (4)$$

Finally, the image user submits the trapdoor $\mathcal{T}_q = \{f'_q\}$ to the cloud server for similarity retrieval.

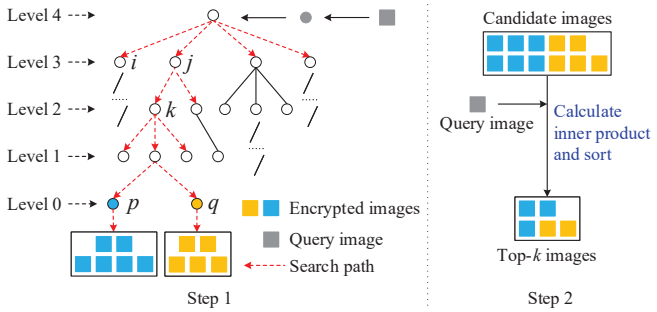


Fig. 7: The specific process of **Search**.

5.3.5 Search

The **Search** process is divided into two steps as shown in Fig. 7, the first step is to find the matched leaf nodes, and the second step is to search the similar images.

`Step 1`: After receiving the query request $\mathcal{T}_q$ submitted by the image user, the cloud server finds the matched nodes in a top-down manner in the hierarchical index tree $\mathcal{I}$. At each level, the cloud server finds the entry point of the next level by locating the node with a vector that has the smallest distance to the vector f'_q. For example, the cloud server separately calculates the inner product values between the vectors EV_i, EV_j stored in the nodes i and j at level-3 in Fig. 7 and the vector f'_q. The smaller the inner product value is, the smaller the distance between the vectors will be. The cloud server finds the entry point j of the level-2 by comparing the distances of the nodes of the level-3 with the query vector f'_q. Likewise, the cloud server finds the entry point k of level-1. Iteratively searching the index tree until the cloud server finds the matched leaf nodes, such as nodes p, q.

`Step 2`: After finding the matched leaf nodes as shown in Fig. 7, the candidate search results are locked in the clusters represented by these nodes. The cloud server calculates the inner product values between the feature vector of each image in the clusters and that of the query image as

$$S_{i,q} = f'_i \cdot {f'_q}^{\top}, \tag{5}$$

and then accurately sorts these inner product values to obtain the identifiers of k encrypted images that have top relevances with query image m_q. Finally, the cloud server returns the corresponding top k most similar images to the image user.

In this `Step 2`, if the cloud server does not find k images within the clusters, it can go back to the parent node of the leaf node and return to the child node of the next nearest neighbor node. This is repeated until the cloud server finds out k images.

5.3.6 Image decryption

After the image user receives the search results $\mathcal{R}$ returned by the cloud server, the private key k_{ie} transmitted by the image owner is used to decrypt $\mathcal{R}$ to obtain the plaintext results $\mathcal{R}'$ as

$$\begin{aligned}\mathcal{R}' &= Dec(\mathcal{R}, k_{ie}) \\ &= (Dec(c_1, k_{ie}), \cdots, Dec(c_k, k_{ie})) \\ &= (m_1, \cdots, m_k).\end{aligned}$$

Remarks. In our SEI, we use the feature vectors extracted by a pre-trained CNN model rather than global or local features to improve search accuracy. At times we implement a hierarchical index tree based on CAK-means clustering that combined AP and K-means clustering to improve search efficiency. Besides, the limited key-leakage mechanism in our SEI distributes $\bar{A}$ and U to image users rather than r, S, M, $\boldsymbol{\pi}$, which evades the risk of untrusted image users leaking keys compared with schemes that unreservedly sharing the encryption keys of feature vector with image users. Moreover, as shown in Section 5.3.4, when the image user generates a query trapdoor, the online help of the image owner is not required because the image owner has transmitted $\bar{A}$ and U to the image user during the **GenKey** stage. Also, the feature vector encryption keys of each image user are different owing to the uniqueness of W and are kept secret to others to prevent key leakage.

5.4 A Toy Example of SEI Scheme

To vividly show the flow of the kNN algorithm with limited key-leakage, we give a toy example in three steps as follows. Consider a simple scenario where the image owner has two images, i.e., $M = \{m_1, m_2\}$, and the image user has one query image m_q. Here $d = 2$, the extracted feature vectors are $f_1 = (2, 3)$, $f_2 = (8, 6)$, and $f_q = (3, 4.2)$. Suppose the image user wants to query the nearest similar image, i.e., $k = 1$. The process of similarity calculation will be implemented as follows.

`Step 1`: Since $d = 2$, the encrypted feature vector f' is 6-dimension. Hence, the image owner randomly selects an invertible 6×6 matrix

$$M = \begin{bmatrix} 17 & 1 & 26 & 38 & 8 & 26 \\ 33 & 38 & 5 & 12 & 1 & 5 \\ 18 & 43 & 9 & 13 & 22 & 32 \\ 36 & 28 & 6 & 41 & 23 & 14 \\ 25 & 44 & 21 & 13 & 5 & 16 \\ 22 & 24 & 6 & 28 & 27 & 28 \end{bmatrix},$$

and obtains the inverse of the matrix M is $M^{-1} =$

$$\begin{bmatrix} -0.028 & 0.067 & -0.175 & -0.121 & 0.060 & 0.241 \\ 0.010 & -0.024 & 0.115 & 0.079 & -0.034 & -0.156 \\ -0.030 & -0.041 & -0.140 & -0.060 & 0.129 & 0.153 \\ 0.047 & -0.025 & 0.166 & 0.129 & -0.092 & -0.241 \\ -0.068 & -0.051 & -0.160 & -0.055 & 0.120 & 0.214 \\ 0.038 & 0.051 & 0.057 & -0.035 & -0.069 & -0.018 \end{bmatrix},$$

two 3-dimensional vectors $r = (1, 3, 1)$ and $S = (2, 12, 3)$, a permutation $\boldsymbol{\pi} = (1, 2, 3)$, a random number $\theta = 3$, and a 3-dimensional vector $X = (6, 1, 2)$. Then, the image owner works out

$$M' = \begin{bmatrix} 18 & 34 & 64 \\ 71 & 6 & 17 \\ 61 & 54 & 22 \\ 64 & 37 & 47 \\ 69 & 21 & 34 \\ 46 & 55 & 34 \end{bmatrix}, A = \begin{bmatrix} 54 & 102 \\ 213 & 18 \\ 183 & 162 \\ 192 & 111 \\ 207 & 63 \\ 138 & 165 \end{bmatrix},$$

and $U = (288, 301, 401, 378, 411, 330)$.

The image user selects random matrix $W = \begin{bmatrix} 4 & 5 \\ 3 & 1 \end{bmatrix}$ for image owner and then receives

$$\bar{A} = \begin{bmatrix} 22.909 & -12.545 \\ -14.455 & 90.273 \\ 27.545 & 24.273 \\ 12.818 & 46.909 \\ -1.636 & 71.182 \\ 32.455 & 2.727 \end{bmatrix}$$

from image owner.

`Step 2`: Our scheme encrypts feature vector one by one with Eq. 2 and Eq. 3. Here, we give the details of encrypting f_1. Given r and $\boldsymbol{\pi}$, the image owner first calculates

$$\begin{aligned} \tilde{f}_1 &= \boldsymbol{\pi}(1 - 2 \cdot 2, 3 - 2 \cdot 3, 1 + 2^2 + 3^2) \\ &= (-3, -3, 14), \end{aligned}$$

then expands $\tilde{f}_1$ to $\hat{f}_1 = (-3, 2, -3, 12, 14, 3)$, finally obtains $f'_1 = (-0.077, -0.988, 1.104, 1.371, -0.265, -1.440)$ by using M^{-1} to multiply $\hat{f}_1$. Similarly, for $f_2 = (8, 6)$, the image owner has the encrypted feature vector $f'_2 = (-5.509, -5.970, -9.828, -1.623, 8.666, 13.380)$.

When querying similar images, the image user encrypts the feature vector f_q of the query image m_q. The image user randomly picks up a number $\gamma = 3.7$, and calculates f'_q according to Eq. 4. Then, the encrypted query feature vector is $f'_q = (3250.08, 3757.72, 6032.48, 5254.74, 4797.42, 5316.9)$.

`Step 3`: After receiving the encrypted f'_1, f'_2 and f'_q, the cloud server calculates the inner products $f'_1 \cdot f'^{\top}_q = 970.14$, $f'_2 \cdot f'^{\top}_q = 4559.88$. Because $f'_1 \cdot f'^{\top}_q - f'_2 \cdot f'^{\top}_q < 0$, the cloud server can infer that f_1 is closer to f_q than f_2, that is, the nearest similar image is m_1, which completes the query process.

In the toy example, the real distances are $D(f_1, f_q) = \sqrt{(2-3)^2 + (3-4.2)^2} = 1.562$, and $D(f_2, f_q) = \sqrt{(8-3)^2 + (6-4.2)^2} = 5.463$. Thus, f_1 is indeed the closer one. It shows that the search result of SEI is correct.

5.5 Extension of SEI Scheme

Although SEI scheme not only provides high accuracy and efficiency, but also avoid sharing keys to image users completely. Unfortunately, since $\mathcal{R}$ is returned by the cloud server, it is inevitable that the cloud server knows all search results and infers the relationship among the submitted trapdoors and even deduces images of interest to the image user. For instance, the image user selects m_q and $m_{q'}$ as query images, the returned top-3 search results are $\mathcal{R}'_q = \{m_1, m_{15}, m_6\}$ and $\mathcal{R}'_{q'} = \{m_1, m_{15}, m_6\}$. The cloud server can still infer that query images m_q and $m_{q'}$ are similar, although it does not know the plaintext information. The image user's query intention is completely exposed to the cloud server. To preserve the image user's query privacy, we introduce some disturbances to query images.

K–anonymity, one of the mainstream privacy protection rules, is mainly used in the data sharing scenario and requires each data to be indistinguishable from at least K–1 other data. Mixing other K–1 indistinguishable data into target data when releasing data not only protects the data privacy from being known by the third parties, but also ensures that the recipient can obtain valid information. Therefore, we design the K–anonymity search principle to hide the image user's query purpose. The following is the definition of K–anonymity search principle.

Definition 1. (K–anonymity search principle) A trapdoor provides K–anonymity protection, if in the trapdoor set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{\mathrm{K}}\}$, any two $\mathcal{T}_i$ and $\mathcal{T}_j$ are indistinguishable.

During the **GenTrap** stage, the image user introduces K–1 confusing trapdoors generated from other images into $\mathcal{T}$ and submits $\mathcal{T}$ to the cloud server. Assuming that the trapdoors submitted to the cloud server is $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{\mathrm{K}}\}$, and the target that the image user wants to query is $\mathcal{T}_1$. Our SEI guarantees that these trapdoors are indistinguishable, so that the cloud server cannot distinguish $\mathcal{T}_1$ from $\mathcal{T}$. In prior encrypted image retrieval schemes [3], [10], [13], the cloud server is able to absolutely confirm the purpose of the image user based on the unique query trapdoor submitted by the image user. But in our scheme, if the cloud server wants to distinguish the image user's target from K trapdoors, the probability of success drops to $1/\mathrm{K}$. The greater the value of K, the higher secure SEI is. However, the more trapdoors are mixed, the lower the efficiency of the cloud server search, and the greater the burden on the image user to filter the images that match the target query from the search results. SEI can achieve a trade-off between security and efficiency.

Remarks. In SEI, we consider both semi-trusted cloud server and untrusted image users at the same time. Compared with previous schemes that leaked all searchable keys, SEI reduces the key leakage and achieves higher security with only one interaction between image owner and image user. Also, it is worth noticing that SEI can significantly improve the search accuracy and efficiency with the help of pre-trained CNN model and index tree. Furthermore, SEI achieves the hiding of search results by employing the K–anonymity technology, which hinders the cloud server from stealing information from the search results. Thus, SEI is feasible in practice, which can be proved in the following performance analysis.

6 Security and Performance Analysis

To illustrate that SEI is secure and feasible in practice, we present the detailed security and performance analysis separately.

6.1 Security Analysis

In this subsection, we will show the process of proof in detail for the `known ciphertext model` and `known background model`. Before presenting, we will define some notations that will be used.

Definition 2. (History) $H = \{\mathcal{M}, \mathcal{I}, \mathcal{T}\}$ where $\mathcal{M}$ is an image database, $\mathcal{I}$ is an index and a set of queries $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_t\}$ is submitted by image users.

Definition 3. (View) The view of cloud server is $V(H) = \{Enc_{sk_o}(\mathcal{M}), Enc_{sk_o}(\mathcal{I}), Enc_{sk_u}(\mathcal{T})\}$. As the encrypted form of a H, $V(H)$ including encrypted images $Enc_{sk_o}(\mathcal{M})$, encrypted index $Enc_{sk_o}(\mathcal{I})$, and encrypted trapdoors $Enc_{sk_u}(\mathcal{T}) = \{Enc_{sk_u}(\mathcal{T}_1), Enc_{sk_u}(\mathcal{T}_2), \cdots, Enc_{sk_u}(\mathcal{T}_t)\}$. Note that the cloud server only observes the views.

Definition 4. (Trace) The trace of a H is $Tr(H) = \{Tr(\mathcal{T}_1), Tr(\mathcal{T}_2), \cdots, Tr(\mathcal{T}_t)\}$, consisting of a series of traces of queries. It captures the information available to the cloud server for each query $\mathcal{T}_i$, including the length of the encrypted query vectors, the search pattern $PA_{\mathcal{T}_i}$ and the search results $\mathcal{R}_i$. Specifically, $Tr(\mathcal{T}_i) = \mathcal{R}_i$, where $\mathcal{R}_i$ is the search results matching the query $\mathcal{T}_i$.

Lemma 1. Our scheme ensures the index confidentiality and indistinguishability under `known ciphertext model` and `known background model`.

Proof. In SEI, the encryption of index is essential to encrypt feature vectors and cluster center vectors. All the cluster center vectors are treated as feature vectors. The encryption method of feature vectors in SEI is more secure than the traditional ASPE. As proved in existing secure kNN algorithm [11], the index is secure under `known ciphertext model`.

In the `known background model`, even if the adversary obtains some plaintext images, it is impossible to extract the underlying feature vectors as long as Principal Component Analysis (PCA) [32] parameters are protected, let alone without ciphertext form corresponding to the plaintext. Although the image user can get A from $\bar{A}$, he still cannot recover M since A only contains partial information of M. Moreover, θ is well protected by the image owner, which further protects the image owner's feature vectors. Thus, the sensitive information about vectors in index will not be leaked under `known background model`. □

Lemma 2. Our scheme ensures the trapdoor confidentiality and indistinguishability under `known ciphertext model` and `known background model`.

Proof. Given two feature vectors f_{q_1} and f_{q_2} from the same query images m_q, the cloud server constructs

$$\begin{cases} f'_{q_1} = \gamma_1 \cdot (f_{q_1} \cdot W^\top \cdot \bar{A}^\top + U), \\ f'_{q_2} = \gamma_2 \cdot (f_{q_2} \cdot W^\top \cdot \bar{A}^\top + U). \end{cases} \tag{6}$$

Since different image users have different $\bar{A}$, W and U, the trapdoor confidentiality and indistinguishability are guaranteed. Even if f_{q_1} and f_{q_2} are from a certain image user in different queries, the cloud server cannot obtain f_{q_1} and f_{q_2} due to different γ, neither distinguish the difference between the two trapdoors. Hence, SEI achieves the trapdoor confidentiality and indistinguishability under `known ciphertext model`.

As for the `known background model`, the cloud server obtains $\bar{A}$, U from other untrusted users. However, the cloud server is still unable to work out f_{q_1} or f_{q_2} because the random matrix W and random number γ are privately held by the image owner and image user, respectively. Similarly, the cloud server cannot distinguish the difference between $\mathcal{T}_{q_1}$ and $\mathcal{T}_{q_2}$ because γ_1 and γ_2 are well protected by the image user. As a result, SEI guarantees the trapdoor confidentiality and indistinguishability under `known background model`. □

6.1.1 In the known ciphertext model

In the `known ciphertext model`, given two histories that have the same trace, if the cloud server cannot distinguish which history from a simulator, he will not learn more knowledge about the index and image database beyond the search and the access pattern.

Theorem 1. Our scheme is secure under the `known ciphertext model`.

Proof. We denote *Sim* as a simulator which can simulate a view V that is indistinguishable from the view of the cloud server. We construct the specific simulation process as follows:

- *Sim* selects random numbers as the feature vectors and outputs $\mathcal{M}' = \{m'_i, 1 \le i \le |\mathcal{M}'|\}$.
- *Sim* randomly picks up vectors r', S', X', a pseudo-random permutation $\boldsymbol{\pi}'$ and an invertible matrix M'. Let $sk'_o = \{r', S', X', \boldsymbol{\pi}', M'\}$. At the same time, *Sim* generates A', U' to the image user and sets $sk'_u = (A', U')$.
- *Sim* selects random numbers as the query vectors and encrypts the vectors to $Enc(\mathcal{T}') = \{Enc_{sk'_u}(\mathcal{T}'_1), Enc_{sk'_u}(\mathcal{T}'_2), \cdots, Enc_{sk'_u}(\mathcal{T}'_t)\}$.
- Based on the search pattern $PA_{\mathcal{T}_i}$, *Sim* can generate $Enc_{sk'_o}(\mathcal{I}')$ as follows:
  1) Assume $PA_{\mathcal{T}_i}$ goes through node j at level h of the index tree. Let $V'_{h,j}$ be the stored vector at the corresponding node in the simulated index tree. $V'_{h,j}$ is initially a null vector.
  2) *Sim* sets $V'_{h,j} = V'_{h,j} + \mathcal{T}'_i$.
  3) After all queries are proceed, *Sim* encrypts $Enc_{sk'_o}(V'_{h,j})$.
- Based on these views, *Sim* outputs $V(H') = (Enc_{sk'_o}(\mathcal{M}'), Enc_{sk'_o}(\mathcal{I}'), Enc_{sk'_u}(\mathcal{T}'))$.

In summary, the search results on $Enc_{sk'_o}(\mathcal{I}')$ with $Enc_{sk'_u}(\mathcal{T}')$ are the same as the trace known by the cloud server. Thus, we claim that no probabilistic polynomial-time (P.P.T.) adversary with more than 1/2 probability can distinguish between the view $V(H)$ and $V(H')$. Especially, since the symmetric encryption has semantic security, no P.P.T adversary can distinguish between $Enc_{sk_o}(\mathcal{M})$ and $Enc_{sk'_o}(\mathcal{M}')$. And the indistinguishability of index and trapdoors is based on the indistinguishability of the secure kNN encryption, which is proved by Lemma 1 and 2. In a word, our scheme is secure under `known ciphertext model`. □

6.1.2 In the known background model

Under the `known background model`, we assume that the cloud server can deduce data information according to past queries and obtain selected background information (e.g., $\bar{A}$, U etc.) from untrusted image users.

Theorem 2. Our scheme is secure under the `known background model`.

Proof. Unlike the proof of Theorem 2, the cloud server has already known $\bar{A}$ and U under the `known background model`. So the simulator generates sk''_o based on the real $\bar{A}$ and U. It has been shown in Lemma 1 and 2 that the index and trapdoors remain indistinguishable, which indicates that a P.P.T. adversary cannot distinguish between $V(H)$ and $V(H'')$. Therefore, our scheme is secure against the adversary in the `known background model`. □

6.2 Performance Analysis

In this part, we analyze the theoretical and actual performance of SEI by comparing with the traditional typical scheme using kNN algorithm to encrypt features and linear search (expressed by kNN-CBIR[2]). For the theoretical performance, we mainly focus on the computation and storage overhead. After that, we perform empirical experiments with a real-world dataset to demonstrate the efficiency and feasibility of SEI scheme.

6.2.1 Theoretical performance

In TABLE 3, we compare the computation complexity of SEI with kNN-CBIR other than **EncImage** and **DecImage** steps. The complexity of generating an image encryption key is also omitted because it is same in both schemes. It can be seen that the computation complexity of SEI and kNN-CBIR in the **GenKey** and **GenTrap** phases is $\mathcal{O}(d^2)$. Moreover, only one more interaction in the **GenKey** phase, will we solve the key leakage problem of untrusted users. Although the computation complexity of **GenIndex** in SEI is higher than that of kNN-CBIR, it is acceptable to improve the efficiency of **Search**. In contrast to kNN-CBIR which uses the linear search algorithm, SEI constructs a hierarchical index tree in **GenIndex** stage which greatly reduces the computation complexity of **Search** from $\mathcal{O}(nlogk)$ to $\mathcal{O}(\frac{n}{K}logK \cdot logk)$. In TABLE 4, we compare the storage and communication overhead of SEI with kNN-CBIR. In the whole process, the image owner only needs to store the image encryption key and the feature vector encryption key, the image user stores the feature vector encryption key, and the cloud server stores the encrypted image database and index. Considering the image encryption key and encrypted image database are the same in these schemes, we evaluate the storage overhead from the remaining two aspects: the feature vector encryption key (denoted as key) and the encrypted index (denoted as index). Although SEI has more storage overhead than kNN-CBIR to achieve high accuracy and efficiency, it is insignificant and acceptable compared to the storage overhead of encrypted images.

TABLE 3: Computation complexity of various schemes

Schemes	GenKey	GenIndex	GenTrap	Search
SEI	$\mathcal{O}(d^2)$	$\mathcal{O}((2^K-1+n)d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(\frac{n}{K}logK \cdot logk)$
kNN-CBIR	$\mathcal{O}(d^2)$	$\mathcal{O}(nd^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(nlogk)$

Notes. "d": the dimension of feature vector; "n": the number of images; "K": the number of cluster centers; "k": the number of search results.

For communication overhead in SEI, the image owner and the image user transfer the keys to each other during the **GenKey** stage. Besides, the image owner sends the index to the cloud server during the **GenIndex** stage, while the image user submits the trapdoor to the cloud server during the **GenTrap** stage. Finally, the cloud server returns the top-k search results to the image user during the **Search** stage. Since SEI allows the image user to generate trapdoor locally, there is one interaction during the **GenKey** stage. In the **GenIndex** stage, although the delivery of the index tree causes a large communication overhead, it is only a one-time operation and is worthwhile to improve search efficiency. Moreover, the communication overhead is mainly consumed by transmitting encrypted images for the image owner. Therefore, the communication overhead of the index tree is tolerable and acceptable. In the **Search** stage, SEI and kNN-CBIR both cost $k\varrho$ bits to return the search results.

2. Randomly selecting two matrices M_1, $M_2 \in R^{d\times d}$ and a Boolean vector $S \in \{0,1\}^n$ to encrypt the feature vectors and finding out the similar images in a linear search manner.

6.2.2 Actual performance

We implemented the proposed scheme in a virtual cloud environment with a KVM processor (2.1 GHZ) and 4GB memory by using Python programming language based on Numpy Library[3] under Ubuntu 18.04 LTS operating system. We use the well-known Caltech256 [33] image database consisting of 256 categories to conduct various experiments. We randomly select 50 categories from 256 categories and each category with at least 80 images. The performance of SEI is evaluated in terms of the search accuracy and efficiency.

Accuracy. For evaluating the search accuracy of the experiment, we use Precision at top-k (P@k) as the performance metrics. P@k is defined as

$$\mathrm{P@}k = \frac{num\_positive}{k},$$

where $num\_positive$ represents the number of positive images in the top-k returned images. For comparison, we vary the value of k from 5 to 40. The search accuracy mainly depends on the feature vector extraction and feature vector dimension, as well as encryption method. TABLE 5 shows the comparison of search accuracy among the MPEG-7 (EHD) and CNN feature descriptors, we sets n=4000 and finds that the CNN features are superior to EHD. Then, based on the 512-dimensional CNN feature, we use the Principal Component Analysis (PCA) method to reduce the dimension to 512, 256, 128, 64, 32 dimensions to show the effect of the dimension on the search accuracy. It can be seen that the 128-dimensional performance is optimal and can reach 66% for k=5. Considering that the current mainstream feature encryption method ASPE, the gap between SEI and traditional ASPE scheme is small. Moreover, the accuracy of kNN-CBIR decreases rapidly than that of SEI as k increases. Additionally, we show the search accuracy under the wrong key. The accuracy of the obtained search results is as low as 1% when the image user encrypts the feature vector with the wrong key, which illustrates that the unauthorized image user cannot get any information from SEI. In TABLE 6, we vary the n from 2000 to 4000 by selecting 40, 50, 60, 70, 80 from each category and find that the search accuracy increases as the number of images in per category increases.

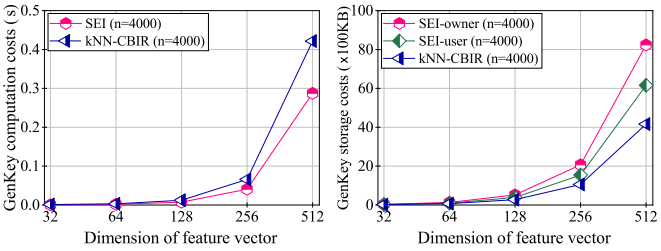
Efficiency. In Fig. 8(a), 8(b), we compare the actual performance of **GenKey** algorithm for SEI and kNN-CBIR schemes. As for the computation costs of key generation, SEI has more computation burden than the kNN-CBIR. This is the sacrifice made by SEI in the key generation phase to avoid sharing the encryption key with the image users. Similarly, more keys than kNN-CBIR take up more storage overhead, such as $M, r, S, \boldsymbol{\pi}, \theta, X, k_{ie}$ for image owner and $W, U, \bar{A}, k_{ie}$ for image user, but the overhead is acceptable for the sake of avoiding the risk of key exposure.

3. http://www.numpy.org/.

TABLE 4: Storage and communication overhead of various schemes

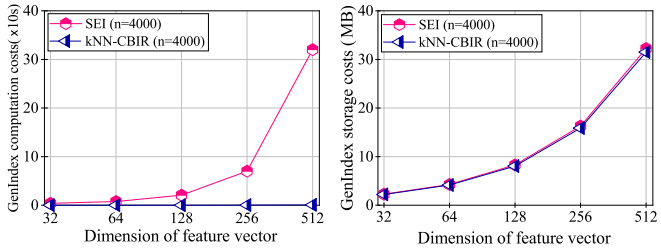
Schemes	Storage overhead		Communication overhead			
	key	Index	**GenKey**	**GenIndex**	**GenTrap**	**Search**
SEI	$(9d^2+17d+9)\omega$	$n'\iota+2n(d+1)\omega$	$(5d^2+6d+2)\omega+\|\lambda\|$	$n'\iota+2n(d+1)\omega$	$2(d+1)\omega$	$k\varrho$
kNN-CBIR	$(2d^2+5d+3)\omega$	$2n(d+1)\omega$	$(2d^2+5d+3)\omega+\|\lambda\|$	$2n(d+1)\omega$	$2(d+1)\omega$	$k\varrho$

Notes. "d": the dimension of feature vector; "n": the number of images; "k": the number of search results; "n'": the number of nodes in index tree; "ι": the bit length of each node; "ω": the bit length of each dimension of vector; "$|\lambda|$": the bit length of image encryption key; "ϱ": the bit length of each encrypted image.



(a) **GenKey** computation costs. (b) **GenKey** storage costs.

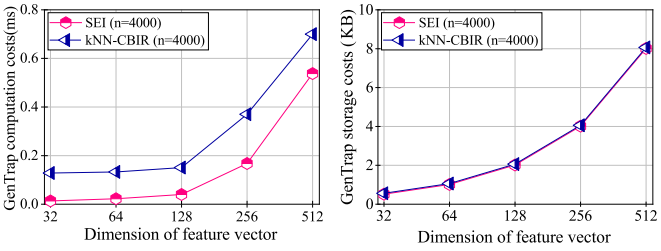
Fig. 8: **GenKey** costs of various schemes.



(a) **GenIndex** computation costs. (b) **GenIndex** storage costs.

(c) **GenIndex** computation costs. (d) **GenIndex** storage costs.

Fig. 9: **GenIndex** costs of various schemes.



(a) **GenTrap** computation costs. (b) **GenTrap** storage costs.

Fig. 10: **GenTrap** costs of various schemes.

TABLE 5: P@k under different dimension of feature vector

k	5	10	20	30	40
32	0.597	0.569	0.542	0.525	0.514
64	0.652	0.625	0.599	0.582	0.569
128	0.660	0.643	0.626	0.612	0.604
256	0.577	0.560	0.543	0.533	0.526
512	0.417	0.390	0.363	0.341	0.324
EHD	0.318	0.225	0.166	0.140	0.124
kNN-CBIR	0.770	0.699	0.623	0.573	0.532
Wrong key	0.011	0.008	0.010	0.008	0.011

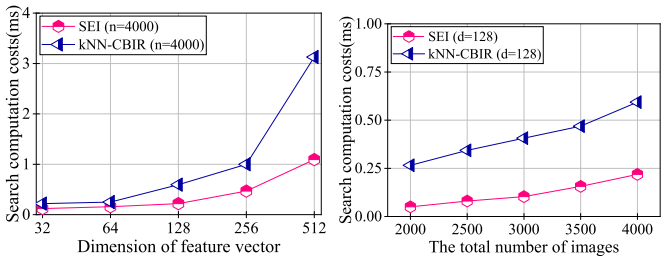
TABLE 6: P@k under different total numbers of images

k	5	10	20	30	40
2000	0.585	0.568	0.548	0.538	0.531
2500	0.508	0.484	0.465	0.456	0.449
3000	0.500	0.475	0.454	0.441	0.434
3500	0.501	0.473	0.451	0.442	0.435
4000	0.660	0.643	0.626	0.612	0.604

In Fig. 9(a), 9(b), 9(c), 9(d), we plot the computation and storage costs in **GenIndex** algorithm. The index construction process mainly consists of building an index tree and encrypting feature vectors, and its overhead is affected by the dimension of feature vector and the total number of images. The computation costs of **GenIndex** increase as the dimension of feature vector and the total number of images increase. It is pointed out that the construction time of the index tree is relatively large due to the application of the clustering algorithm. At the same time, it must be said that the **GenIndex** algorithm is just a one-time operation and the efficiency improvement effect in the **Search** stage brought about by the index tree is remarkable. The storage costs of SEI also are more than kNN-CBIR since we build a tree to speed up the search process. However, we can see that their storage costs are very close from Fig. 9(b), 9(d), which clearly speaks that the storage costs of the index tree make up a small part.

In Fig. 10(a), 10(b), we show the computation and storage costs of trapdoor generation in **GenTrap** algorithm. We notice that the computation costs of the **GenTrap** algorithm in SEI mainly depend on the feature vector dimension d of the query image. When d varies between 32 and 512, the computation costs increase with d in kNN-CBIR and SEI. For example, when setting d=128, the trapdoor generation time required by kNN-CBIR and SEI is 0.15ms and 0.04ms, respectively. Besides, the storage costs of SEI are close to that of kNN-CBIR. For example, the storage overhead of kNN-CBIR is 2.06KB, and that of SEI is 2.01KB. Therefore, SEI can achieve the image user to generate the trapdoor locally with lower computation and storage overhead.

In Fig. 11(a), 11(b), we display the performance of en-



(a) **Search** computation costs. (b) **Search** computation costs.

Fig. 11: **Search** costs of various schemes.

crypted image search in **Search** algorithm under different dimension of feature vector and the total number of images. When n=4000, computation costs of **Search** have been going on up as dimension increasing. However, the growth of SEI is less than that of kNN-CBIR. By varying the total number of images from 2000 to 4000, the computation costs of **Search** in SEI are significantly lower than that of kNN-CBIR because we built the index tree in the **GenIndex** phase to improve efficiency. For example, with n=4000, d=128, the kNN-CBIR required 0.6ms while our scheme only needed 0.22ms. In the face of a large-scale image database, SEI will have significant advantages compared to the traditional kNN-CBIR scheme. We omit the analysis of storage costs in **Search** algorithm since the storage costs of SEI are the same as that of kNN-CBIR when returning the top k similar images. However, it should be mentioned that in an extended version of SEI, mixing K–1 other images into the target query will result in a K–1 increase in search results and a K–1 increase in search time, which is a trade-off between efficiency and security.

Above all, we draw that SEI can completely realize the design goals in terms of the actual performance using a real-world dataset. In comparison with the prevalent kNN-CBIR scheme, we can verify that our scheme is feasible in practice.

7 Conclusion

In this paper, we investigate similarity search for encrypted images in secure cloud computing. Concretely, we introduce a clustering improvement method and give the design method of the hierarchical index tree. With these two techniques, SEI can efficiently perform the retrieval process and achieve high accuracy based on features extracted by the CNN model. Further, we consider untrusted image users in SEI and hence propose a similarity calculation method with limited key-leakage. We also give strict security analysis and conduct experiments on a real-world dataset, which indicate that SEI is secure and feasible in practice.

Our future works can be summarized as follows:

- *Supporting dynamic update*: We will develop a way to build an efficient index and enable support dynamic update of image databases.
- *Supporting multi-owner scenario*: The management issues for different image encryption keys will be resolved to adapt to the multi-owner scenario.
- *Supporting verificatable*: The image user who receives the search results in the image search can verify the results.

Acknowledgments

This work was supported in part by the Key Program of NSFC Grant (U1405255), National Natural Science Foundation of China (No. 61702404, No. 61972094), the Fundamental Research Funds for the Central Universities (No. JB191506), Shaanxi Science and Technology Coordination and Innovation Project (2016KTZDGY05-06), the National Natural Science Foundation of Shaanxi Province (No. 2019JQ-005), China Postdoctoral Science Foundation Funded Project(No. 2017M613080).

References

[1] C.-Y. Hsu, C.-S. Lu, and S.-C. Pei, "Image feature extraction in encrypted domain with privacy-preserving sift," *IEEE transactions on image processing*, vol. 21, no. 11, pp. 4593–4607, 2012.

[2] M. Li, M. Zhang, Q. Wang, S. S. Chow, M. Du, Y. Chen, and C. Lit, "Instantcryptogram: Secure image retrieval service," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2222–2230.

[3] Z. Xia, N. N. Xiong, A. V. Vasilakos, and X. Sun, "Epcbir: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing," *Information Sciences*, vol. 387, pp. 195–204, 2017.

[4] M. Shen, G. Cheng, L. Zhu, X. Du, and J. Hu, "Content-based multi-source encrypted image retrieval in clouds with privacy preservation," *Future Generation Computer Systems*, 2018.

[5] C. Guo, S. Su, K.-K. R. Choo, and X. Tang, "A fast nearest neighbor search scheme over outsourced encrypted medical images," *IEEE Transactions on Industrial Informatics*, 2018.

[6] X. Wang, J. Ma, X. Liu, and Y. Miao, "Search in my way: Practical outsourced image retrieval framework supporting unshared key," in *Proc. IEEE Conference on Computer Communications (INFOCOM'19)*. IEEE, 2019, pp. 2485–2493.

[7] H. Liang, X. Zhang, Q. Wei, and H. Cheng, "Secure image retrieval with multiple keys," *Journal of Electronic Imaging*, vol. 27, no. 2, p. 023032, 2018.

[8] Z. Xia, Y. Zhu, X. Sun, Z. Qin, and K. Ren, "Towards privacy-preserving content-based image retrieval in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 276–286, 2015.

[9] B. Cheng, L. Zhuo, Y. Bai, Y. Peng, and J. Zhang, "Secure index construction for privacy-preserving large-scale image retrieval," in *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*. IEEE, 2014, pp. 116–120.

[10] X. Li, Q. Xue, and M. C. Chuah, "Casheirs: Cloud assisted scalable hierarchical encrypted based image retrieval system," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[11] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.

[12] L. Zhang, T. Jung, C. Liu, X. Ding, X.-Y. Li, and Y. Liu, "Pop: Privacy-preserving outsourced photo sharing and searching for mobile devices," in *2015 IEEE 35th International Conference on Distributed Computing Systems*. IEEE, 2015, pp. 308–317.

[13] J. Yuan, S. Yu, and L. Guo, "Seisa: Secure and efficient encrypted image search with access control," in *Proc. IEEE conference on computer communications (INFOCOM'15)*. IEEE, 2015, pp. 2083–2091.

[14] Y. Zhu, Z. Wang, and Y. Zhang, "Secure k-nn query on encrypted cloud data with limited key-disclosure and offline data owner," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2016, pp. 401–414.

[15] L. Zhang, T. Jung, K. Liu, X.-Y. Li, X. Ding, J. Gu, and Y. Liu, "Pic: Enable large-scale privacy preserving content-based image search on cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3258–3271, 2017.

[16] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. IEEE, 2000, pp. 44–55.

[17] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[18] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE transactions on parallel and distributed systems*, vol. 27, no. 2, pp. 340–352, 2015.

[19] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3008–3018, 2017.

[20] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[21] W. Lu, A. Swaminathan, A. L. Varna, and M. Wu, "Enabling search over encrypted multimedia databases," in *Media Forensics and Security*, vol. 7254. International Society for Optics and Photonics, 2009, p. 725418.

[22] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'10)*. IEEE, 2010, pp. 3384–3391.

[23] Y. Huang, J. Zhang, L. Pan, and Y. Xiang, "Privacy protection in interactive content based image retrieval," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[24] T. K. Hazra, S. R. Chowdhury, and A. K. Chakraborty, "Encrypted image retrieval system: a machine learning approach," in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2016, pp. 1–6.

[25] Z. Xia, L. Jiang, D. Liu, L. Lu, and B. Jeon, "Boew: A content-based image retrieval scheme using bag-of-encrypted-words in cloud computing," *IEEE Transactions on Services Computing*, 2019.

[26] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 601–612.

[27] Z. A. Abduljabbar, H. Jin, A. Ibrahim, Z. A. Hussien, M. A. Hussain, S. H. Abbdal, and D. Zou, "Privacy-preserving image retrieval in iot-cloud," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 799–806.

[28] Y. Zhu, Z. Huang, and T. Takagi, "Secure and controllable k-nn query over encrypted cloud data with key confidentiality," *Journal of Parallel and Distributed Computing*, vol. 89, pp. 1–12, 2016.

[29] M. S. Nair and M. Rajasree, "Fine-grained search and access control in multi-user searchable encryption without shared keys," *Journal of Information Security and Applications*, vol. 41, pp. 124–133, 2018.

[30] Y. Zhu, J. Yu, and C. Jia, "Initializing k-means clustering using affinity propagation," in *2009 Ninth International Conference on Hybrid Intelligent Systems*, vol. 1. IEEE, 2009, pp. 338–343.

[31] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.

[32] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[33] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

Jianfeng Ma (M'16) received the Ph.D. degree in computer software and telecommunication engineering from Xidian University, Xi'an, China, in 1995. He was a Research Fellow with Nanyang Technological University, Singapore, from 1999 to 2001. He is currently a Professor and a Ph.D. Supervisor with the Department of Cyber Engineering, Xidian University. His current research interests include information and network security, wireless and mobile computing systems, and computer networks.

Yinbin Miao (M'18) received the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He was a Post-Doctoral Fellow with Nanyang Technological University, Singapore, from 2018 to 2019. He is currently a Lecturer with the Department of Cyber Engineering, Xidian University. His current research interests include information security and applied cryptography.

Yue Wang received the B.E. degree in Communication Engineering from Xidian University, Xi'an, China, in 2018. He is currently a Ph.D. candidate in School of Cyber Engineering, Xidian University. His current research interests include trusted computing, intelligent system security and CPS security.

Ximeng Liu (M'16) received the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2015. He is a Professor with the Key Laboratory of Information Security of Network Systems, College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His current research interests include applied cryptography and big data security.

Yingying Li received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 2017. She is currently a Ph.D. candidate in School of Cyber Engineering, Xidian University. Her current research interests include cloud computing security and multimedia security.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. degree from Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship at the University of Texas at San Antonio (UTSA). He is the recipient of various awards including the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, ESORICS 2015 Best Paper Award. He is an Australian ComputerSociety Fellow, and an IEEE Senior Member.