IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Toward Efficient Encrypted Image Scrambling.

[1] Miss. Kajal M. Dhodare, [2] Miss. Dipali. G. Mandare , [3] Miss. Vaishnavi P. Gaddekar [4] Miss. Ishika R. Waghmare, [5] Miss. Mehnaz sheikh

[1] B.Tech. Student, [2] Assistant Professor Computer Science & Engineering, Ballarpur Institute of Technology, India

ABSTRACT : Outsourcing image search services to public clouds is an ever-increasing trend. However, directly outsourcing image datasets to untrusted clouds introduces privacy concerns. Several secure image retrieval schemes have been proposed recently. However, most of them require participation from image owners when building secure indexes, which wastes many computational resources of the image owners. Several schemes are proposed to solve this problem, but they suffer from low search accuracy on large datasets. In this paper, we propose the first secure image retrieval scheme that simultaneously solves these two problems. To obtain higher search accuracy, we extract image features via fine-tuned convolutional neural networks. Then, the image features are encrypted by using the secure k-Nearest Neighbor algorithm. To improve search speed and reduce the cost of image owners, we let cloud servers locally build a secure hierarchical index graph by using the encrypted image features. Besides, the secure index can be built and updated in parallel. We provide security analysis for the proposed scheme. Performance evaluations on the CIFAR-10 dataset show that the proposed scheme is practical. Moreover, compared with a recent scheme, our scheme can save more index construction time and cost of image owners when building secure indexes.

Key Word : Content-based image retrieval, encrypted image retrieval, navigable small world graph, secure index.

I. INTRODUCTION

With the rapid development of multimedia devices, a great many images are created every day. Image retrieval is a promising technology that helps us quickly find the images we are interested in. Content-based image retrieval (CBIR) plays a vital role in image retrieval. CBIR aims to use the visual content of a query image to search similar images from an image database, which is useful in many fields, such as remote diagnosis [1], face recognition [2], and online shopping [3]. Concerning the huge storage and the complicated maintenance cost, more and more image owners prefer to outsource huge image datasets and image search services to public clouds. Then, authorized users can retrieve similar images from the clouds. Unfortunately, although the public clouds reduce the cost of the image owners, they also introduce new security threats [4]. Since the outsourced image datasets may contain sensitive information, directly outsourcing the image datasets to untrusted cloud servers may cause the sensitive images to be stolen by commercial

The associate editor coordinating the review of this manuscript and approving it for publication was Maurizio Tucci.

opponents, hackers, and cloud providers. Using traditional cryptosystems to protect the images can avoid the disclosure of the sensitive information, but the public clouds can no longer provide image search services.

Researchers have made great efforts to achieve secure image retrieval. The authors of [5]–[9] propose several schemes based on feature encryption. Their image owners first extract image features and generate an encrypted searchable index. The encrypted index is then outsourced to clouds, such that the clouds can search similar images for users. Although these schemes achieve good search efficiency and search accuracy, their image owners have to undertake index generation and encryption tasks on their own, which consume a lot of computational resources. To combat this problem, Yuan *et al.* [6] propose a scheme called SEISA, in which they build a secure hierarchical index tree by recursively employing a secure k-means outsourcing algorithm. This algorithm allows image owners to delegate many computational tasks to clouds. However, the algorithm requires the owners to re-encrypt the centroid vectors of new clusters. Thus their image owners still consume many computational and communicational resources when the clouds build a secure index.

To tackle this problem, the authors of [10]–[17] introduce several schemes based on image encryption. Their image owners encrypt images using some encryption techniques and outsourcethemtoclouds.Then,thecloudslocallyextractfeatures from the encrypted images to build a searchable index. Theauthorsof[10]–[12]usehomomorphicencryption[18]to encrypt images. Though they achieve good search accuracy, the methods are impractical due to high time and space cost. Other works [13]–[17] use other special encryption techniques to encrypt images. Despite their success, the algorithms suffer low search accuracy since the cloud servers can only extract low-level image features (e.g., color histograms). Thus, the above schemes based on image encryption are not suitable for large-scale image retrieval.

In order to realize large-scale secure image retrieval, we focus on the research line based on feature encryption. However, as stated above, participation from image owners is still needed during the index construction process. To overcome this obstacle, we propose a secure image search scheme which allows image owners to fully delegate all computational tasks to clouds when building secure indexes. Our idea is to create a proximity graph as our index, which can be built by using encrypted features to find similar encrypted images and establishing connections between them. The contributions of our work can be summarized as follows:

1) We propose a simple yet effective secure image search scheme which can greatly reduce the cost of image owners when building secure indexes.
2) To obtain higher search accuracy, we extract image features via fine-tuned Convolutional Neural Networks (CNNs). Then, the image features are protected by using the secure k-Nearest Neighbor (kNN) algorithm.
3) To improve search speed, we let clouds employ the Hierarchical Navigable Small World (HNSW) graph algorithm to build a secure index without participation from image owners. The construction and update algorithms of the secure index can be run in parallel.
4) We provide security analysis for the proposed scheme. Performance evaluations on the CIFAR-10 dataset demonstrate that the proposed scheme is efficient.

The rest of this paper is organized as follows. Section II reviews the related work. Section III describes the system model, threat model, design goals, and preliminaries. Section IV elaborates our scheme. Section V provides security analysis. Section VI presents the performance evaluations. Finally, we conclude the whole paper in Section VII.

II. RELATED WORK

Searchable encryption (SE) allows query users to securely retrieve specific information on encrypted datasets. Early SE schemes focus on private single keyword retrieval over encrypted documents [19]. Afterwards, many SE schemes were proposed to support various search scenarios including multiple keywords search [20], conjunctive keywords search [21], distributed search [22], [23] and similarity search [24]–[26]. However, most of them are not suitable for encrypted image retrieval tasks.

Several secure image retrieval schemes [5]–[17] have been proposed in the past few years. These schemes can be divided into two main categories, schemes based on feature encryption and schemes based on image encryption.

In the schemes [5]–[9] based on feature encryption, the image owners first extract image features, generate searchable index, encrypt the index and outsource the secure index to cloud servers. Then, the cloud servers realize secure image search via the encrypted index. Lu *et al.* [5] design the first secure image retrieval scheme by using order preserving encoding and Min-Hash. In their scheme, the image owners establish a secure inverted index to improve search speed. In [6]–[9], the authors propose secure image search schemes by using the secure kNN algorithm [24]. Specifically, Yuan *et al.* [6] extract Fisher vectors as image features and use the k-means algorithm to build a hierarchical index tree to boost search speed. Li *et al.* [7] extract CNN features from images and take a category-based hierarchical index tree as their index structure. In [8], [9], the authors employ Locality-Sensitive Hashing (LSH) [27] to reduce search time. Though the above schemes give good performance on the task of secure image search, they are not practical enough since their image owners have to consume a lot of time and computational resources when building secure indexes.

Several schemes [10]–[17] based on image encryption have been proposed to solve the above problem. In these schemes, the image owners employ special encryption algorithms to encrypt images and upload them to cloud servers. Then, the cloud servers locally deal with feature extraction and index generation. In [10]–[12], the authors propose secure image search schemes by using homomorphic encryption [18]. However, these schemes introduce huge time and space cost, and require extensive user participation during the search phase. Ferreira *et al.* [13] propose a secure image search scheme which uses probabilistic encryption to protect texture information and deterministic encryption to protect color information. In [14]–[16], the authors introduce encrypted image search schemes by jointly using stream cipher and permutation cipher. Xia *et al.* [17] propose a secure image retrieval scheme with the help of permutation encryption and image segmentation. The above schemes can only extract low-level image representations (e.g. color histograms) from encrypted images. Because of the ‘‘semantic gap’’, image retrieval systems using low-level features yield low search accuracy [28]. Thus the above schemes are not practical for large-scale image retrieval.

Therefore, despite the success of existing schemes, they either require image owners to undertake index generation, or yield low search accuracy. To overcome these problems, we propose a secure image retrieval scheme which employs fine-tuned CNNs to extract semantic features and allows clouds to build a secure index graph locally to improve search speed.

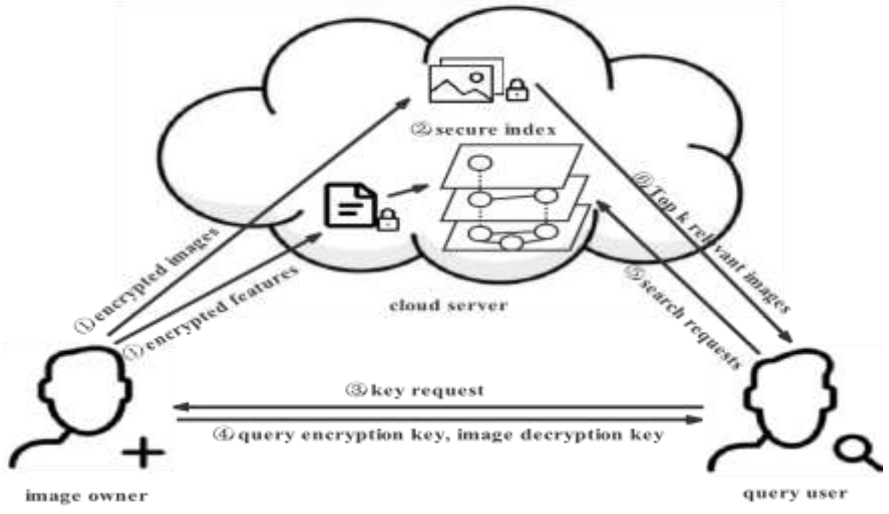

FIGURE 1. System model.

III. PROBLEM FORMULATION

A. *SYSTEM MODEL*

Our system model consists of three types of participants: image owners, query users and a cloud server, as illustrated in Figure 1. The image owners extract feature vectors from their images. To ensure data privacy, the image owners encrypt their images and feature vectors. Then they outsource all the encrypted data to the cloud server. After receiving all the encrypted data, the cloud server uses the encrypted feature vectors to build a secure index locally to boost the search speed. Authorized query users can request an image decryption key and a query encryption key from the image owners. Then,theygenerateencryptedsearchrequestsusingthequery encryption key to send to the cloud server. After receiving the encrypted search requests, the cloud server retrieves and returns relevant encrypted images to the query users. Finally, the query users decrypt all the encrypted images using the image decryption key.

B. *THREAT MODEL*

The cloud server is assumed to be "honest-but-curious" in our system. In other words, the cloud server correctly performs our protocol, but tries to learn some private information. In addition, we assume that the image owners and the query users are fully trusty. Therefore, the image owners and the query users do not collude with the cloud server. Finally, according to the available data to the cloud server, we consider the known ciphertext model, also known as the ciphertext-only model. That is, the cloud server can only obtain all the encrypted images, the corresponding encrypted feature vectors, the encrypted search requests and the secure index structure.

C. *DESIGN GOALS*

The goals of our secure image retrieval scheme are described as follows:

1) Security Guarantee: The private information of all the encrypted data cannot be learned by the cloud server.

1) Fine-tuned Convolutional Neural Network (CNN) Feature Extraction: CNN models have achieved much better results than traditional methods (e.g., SIFT) in many image-related tasks [29]. It has been shown that the fully connected layers of AlexNet [30] provide high-level visual feature representations of images. However, the high-dimensional feature vectors are inefficient for image search. In recent years, in order to reduce the storage and the computational cost, a series of dimension reduction methods [3], [31]–[33] are proposed to generate more compact feature representations. Lin *et al.* [3] propose to embed a new latent layer with less neurons between the F7 and the F8 layers of the pre-trained AlexNet, and then fine-tune the new CNN model. Finally, the feature representations of images are extracted from the latent

layer, and then transformed to binary hash codes. The experimental results in [3] show that their method outperforms several state-of-the-art methods, e.g., LSH [27], ITQ [34], CNNH [31].

2) Hierarchical Navigable Small World (HNSW) Graph: A navigable small world (NSW) graph [35] is an approximate Delaunay graph [36] with nodes corresponding to the stored objects and edges connecting each node with several nodes who are most similar to it in a given metric space. By starting from an entry point and searching the NSW graph greedily, we can obtain approximate kNN results. A HNSW graph [37] is a mutil-layer NSW graph similar to the skip list structure. We can regard the upper layers of the HNSW graph as some express lanes which can speed up searches. That is, an upper layer randomly selects nodes from its lower layer, and takes them as the stations of an express lane, therefore, we can skip some nodes in the upper layer to make searches faster. The authors in [37] demonstrate the search complexity of the HNSW graph is $O(\log(N))$. More specific algorithm description will be presented in Section IV-C.

IV. SECURE IMAGE SEARCH SCHEME

A. IMAGE FEATURE EXTRACTION

A new fully connected layer with a custom size is embedded between the F7 and the F8 layers of the pre-trained AlexNet [30]. The image owners fine-tune the new CNN model using their training image set to learn domain specific image representations. Then, the parameters of all layers of this model are updated using the backpropagation algorithm. The image owners feed their images to the fine-tuned CNN model to extract image features from the new layer. Thus, the CNN model becomes our image feature extractor. For retrieval accuracy, we do not transform the feature vectors to binary hash codes like [3]. The image owners also share the image feature extractor to authorized query users. Finally, any authorized query user can extract feature vectors from his/her query images using the same fine-tuned CNN model.

In fact, our feature extraction algorithm is not limited to AlexNet and can also employ other advanced CNN models such as VGGNet [38]. It is noteworthy that multi-label image retrieval is a more practical and challenging task in the CBIR field.Inordertosolvemulti-labelimageretrievaltasks,recent studies [39], [40] propose some CNN models for learning hash functions that preserve multi-layer semantic similarity between multi-label images. By inserting a new fully connected layer to these models and using these models as our feature extractors, our scheme can achieve secure multi-label image retrieval.

Fine-tuning neural networks and extracting features may consume a lot of computing resources. In recent years, researchers have proposed to use fully homomorphic encryption to offload expensive neural network computing on public clouds while protecting privacy [41]–[43]. Thus, fine-tuning models and extracting features are not longer big burdens to image owners.

B. FEATURE ENCRYPTION METHOD

In this subsection, we will introduce the encryption technology of feature vectors. We protect the feature vectors by using the Asymmetric Scalar-product Preserving Encryption(ASPE)algorithm[24]whichsupportskNNcomputation on encrypted feature vectors. We describe the core content of the ASPE algorithm as follows.

1) Key: Two $(n + 1)\times(n + 1)$ invertible random matrices $\{M_1,M_2\}$ and an $(n + 1)$-bit binary random vector S.

2) Data encryption: For an n-dimension database point p, the data owner first extends it to $P = -0.5\sum_{i=1}^{n} p^2_i ,p_1,p_2,\ldots,p_n$, and then splits P into two random vectors P_1 and P_2 according to S as: If $S[i] = 0$, $P_1[i]$ and $P_2[i]$ are set randomly such that $P_1[i] + P_2[i] = P[i]$; otherwise, $P_1[i]$ and $P_2[i]$ are set equal to $P[i]$. Finally, p is encrypted as $P' = (M_1^T P_1, M_2^T P_2)^T$.

3) Query encryption: For an n-dimension query point q, the query user first randomly selects a positive number r, and then extends q to $Q = (r,rq_1,rq_2,\ldots,rq_n)$. Then Q is split into two random vectors Q_1

and Q_2 according to S as: If $S[i] = 0$, $Q_1[i]$ and $Q_2[i]$ are set equal to $Q[i]$; otherwise, $Q_1[i]$ and $Q_2[i]$ are set randomly such that $Q_1[i] + Q_2[i] = Q[i]$. Finally, q is encrypted as $Q' = \left(M_1^{-1}Q_1, M_2^{-1}Q_2\right)$.

4) Euclidean distance (ED) comparison: Let $P^0{}_a$ and $P^0{}_b$ be the encrypted values of the data points p_a and p_b. We can determine whether p_a is closer to a query point q than p_b by checking whether $P^0{}_a - P'_b) \cdot Q' > 0$, where Q^0 is the encrypted value of the query point q. The computation is given as follows.

$P0_a - P'_b) \cdot Q0$

$$= \left[\left(M_1^T P_{a1}, M_2^T P_{a2}\right) - \left(M_1^T P_{b1}, M_2^T P_{b2}\right)\right]^T \cdot Q0$$

$$= \left[M_1^T (P_{a1} - P_{b1}), M_2^T (P_{a2} - P_{b2})\right]$$
$$\cdot \left(M_1^{-1}Q_1, M_2^{-1}Q_2\right)$$
$$= (P_{a1} - P_{b1}) \cdot Q_1 + (P_{a2} - P_{b2}) \cdot Q_2T$$

$$= (P_a - P_b) \cdot Q$$

$$= 0.5r\,[ED(P_b,Q) - ED(P_a,Q)]. \qquad (1)$$

C. SECURITY DESIGN

In this subsection, we elaborate our encrypted image retrieval scheme. It is worth noting that images can be encrypted by using the Advanced Encryption System (AES) before outsourced to the cloud server. Our scheme consists of 6 blocks:GenKey,EncData,EncQuery,BuildIndex,Searchand Update. We present the detailed description of each algorithm as follows.

GenKey: The image owner creates a secret key $SK = k_{img},S,M_1 \cdot M_2\}$, where k_{img} is used for image encryption, S is an $(n + 1)$-bit random vector, M_1 and M_2 are $(n + 1)\times(n + 1)$invertiblerandommatrices.Then,theimage owner shares the secret key to authorized query users.

EncData: The image owner extracts feature vectors from his/her images via an image feature extractor. To ensure data privacy, the image owner: (1) encrypts the images using the AES algorithm; (2) encrypts the feature vectors using the data encryption algorithm of the ASPE algorithm. The encrypted feature vectors, which we refer to as the encrypted index vectors, will be associated with the nodes of our secure index.

EncQuery: The query users extract feature vectors from their query images via the same image feature extractor. Then, the query users generate encrypted query vectors by using the query encryption algorithm of the ASPE algorithm. Encrypted query vectors are also named as encrypted search requests. In order to build the secure index, the image owner also needs to generate encrypted query vectors for searching the nearest nodes in the secure index.

BuildIndex:Theimageownersendstheencryptedimages, the corresponding encrypted index vectors and the encrypted query vectors to the cloud server. After receiving all the encrypted data, the cloud server builds a secure HNSW index graph locally to increase search speed. As illustrated in Figure 2, each layer of the secure HNSW index graph is a secure NSW graph, where nodes are coresponding to encrypted index vectors and edges are formed between

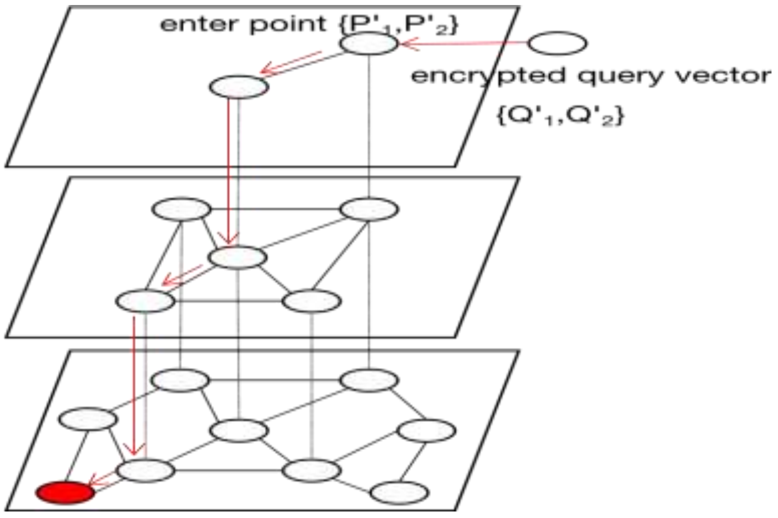

FIGURE 2. A search example of our secure HNSW graph.

similar nodes. The nodes of higher layer graphs are the subsets of lower ones, such that traversing the higher layer graphs are much faster than traversing the lower ones.

In this paragraph, we introduce some notations. We use $N_i = \{MaxL_i, P^0{}_i, Q^0{}_i, Neighbors_i[0...MaxL_i], EImgId_i$ to denote a node, where *i* is its index. Each node contains five values: (1) the maximum layer of the node, denoted as $MaxL_i$, (2) an encrypted index vector, denoted as $P^0{}_i$, (3) an encrypted query vector, denoted as $Q^0{}_i$, (4) the neighbor nodes of it at each layer from the 0th layer to its maximum layer, denoted as $Neighbors_i[0...MaxL_i]$ (initially filled with $MaxL_i + 1$ empty arrays), specifically, its neighbor nodes at layer *h* is denoted by $Neighbors_i[h]$. (5) the id of its corresponding encrypted image, denoted as $EImgId_i$.

Here, we present the search algorithm of a secure NSW graph, which is an important building block of our secure index construction algorithm. For an encrypted query vector Q^0, the search algorithm starts from an entry node to employ Breadth-First Traversal to find *k* nearest nodes to Q^0. The pseudo-code for searching a secure NSW graph is shown in Algorithm 1. The search algorithm is a combination of Breadth-First Traversal and greedy algorithm. Specifically, as shown in Eq. 1, given two nodes *a* and *b* and the encrypted query vector Q^0, we can determine which node is closer to Q^0. Then we create a result set and a candidate set, noting that the size of the result set is fixed to *k*. Each time the algorithm selects the nearest node N_C from the candidate set to Q^0. Then, for every neighbor node N_i of N_C, the algorithm selects the furthest node N_F from the result set to Q^0, and adds N_i to the result set and the candidate set if N_i is closer to Q^0 than N_F. The process is repeated until the candidate set is empty or N_F is closer to Q^0 than N_C. Note that the algorithm may reach a local minimum rather than a global minimum, thus the algorithm may not return the true *k* nearest nodes. Increasing the number of returned nodes *k* leads to higher search accuracy at the cost of longer search time.

The construction process of the secure HNSW index graph is quite straightforward. At the initial stage of the process,

Algorithm 1 SecureGraphSearch

Input: layer of secure HNSW graph *h*, entry point N_{EP}, encrypted query vector Q^0, number of nodes to return *k*

Output: *k* nodes nearest to Q^0

/* *CandidateHeap* is a min heap, in which the root element is the node nearest to Q^0. *ResultHeap* is a max heap, in which the root element is the node furthest to Q^0. An element of *CandidateHeap* and *ResultHeap* is a tuple that can be compared based on its second attribute value. */

```
1     $VisitedList \leftarrow N_{EP}$;
2     $dist_{EP} \leftarrow -N_{EP}.P^0 \cdot Q^0$;
3     $Element \leftarrow (N_{EP}, dist_{EP})$;
4     $CandidateHeap.push(Element)$;
5     $ResultHeap.push(Element)$; 6 while $|CandidateHeap| > 0$ do
7         $(N_C, dist_C) \leftarrow CandidateHeap.GetRoot()$;
8         $(N_F, dist_F) \leftarrow ResultHeap.GetRoot()$;
9         if $dist_C > dist_F$ then
10            breakwhile;
11        for $N_i$ ⍰ $N_C.Neighbors[h]$ do
12            if $N_i$ ⍰ $VisitedList$ then
13                $VisitedList \leftarrow VisitedList$ ⍰ $N_i$;
14                $(N_F, dist_F) \leftarrow ResultHeap.GetRoot()$;
15                $dist_i \leftarrow -N_i.P^0 \cdot Q^0$;
16                if $dist_F > dist_i$ or $|ResultHeap| < k$ then
17                    $element \leftarrow (N_i, dist_i)$;
18                    $CandidateHeap.push(element)$;
19                    $ResultHeap.push(element)$;
20                    if $|ResultHeap| > k$ then
21                        $ResultHeap.pop()$;
22 return $ResultHeap$;
```

the cloud server just creates a node for the first encrypted index vector and the corresponding encrypted query vector, and takes the node as the entry point of the secure HNSW index graph. Then, the cloud server continuously inserts new nodes to the secure index. That is, every new node is connected to its nearest nodes in the secure index graph. The pseudo-code for the insertion algorithm is presented in Algorithm 2. Specifically, for every encrypted index vector and the corresponding encrypted query vector, the cloud server creates a new node and randomly selects a maximum layer *MaxLayer* with an exponentially decaying probability distribution. Then, the insertion process for the node is divided into two phases: accelerated search phase and insertion phase.

During the accelerated search phase, the cloud server starts from the entry point of the secure HNSW index graph and

Algorithm2 InsertNode

Input: secure HNSW index graph *SIGraph*, encrypted index vector P^0, encrypted query vector Q^0, encrypted image id *EImgId*, maximum number of established connections *M*, maximum number of established connections at the 0th layer M_{layer0}, candidate number *ef* , normalization factor m_L **Output**: update *SIGraph* by inserting new node

```
1     $N_{EP} \leftarrow SIGraph.Entrypoint$;
2     $L \leftarrow N_{EP}.MaxL$;
3     $MaxLayer \leftarrow$ b$-\ln(random(0,1)) \times m_L$c; 4 $Neighbors \leftarrow MaxLayer + 1$ empty arrays;
5     $N_{new} \leftarrow MaxLayer, P^0, Q^0, Neighbors, EImgId$;
6     for $l \leftarrow L$ downto $MaxLayer + 1$ do
7     $N_{EP} \leftarrow SecureGraphSearch(l, N_{EP}, Q^0, 1)$;
8     for $l \leftarrow \min(L, MaxLayer)$ downto 0 do
```

```
9        $ResultHeap \leftarrow SecureGraphSearch(l, N_{EP}, Q^0, ef)$;
10       $friends \leftarrow$ top $M$ nearest nodes from $ResultHeap$ to $Q0$;
11       for $N_i \in friends$ do
12           $N_{new}.Neighbors[l] \leftarrow N_{new}.Neighbors[l] \cup N_i$;
13           $N_i.Neighbors[l] \leftarrow N_i.Neighbors[l] \cup N_{new}$;
14           $neighbors \leftarrow N_i.Neighbors[l]$;
15           if $l == 0$ then
16               if $|neighbors| > M_{layer0}$ then
17                   $N_i.Neighbors[l] \leftarrow ShrinkNeighbors(N_i, neighbors)$;
18           else
19               if $|neighbors| > M$ then
20                   $N_i.Neighbors[l] \leftarrow ShrinkNeighbors(N_i, neighbors)$;
21       $N_{EP} \leftarrow$ get the nearest node from $friends$ to $Q^0$;
22   if $MaxLayer > L$ then
23       $SIGraph.entrypoint \leftarrow N_{new}$;
24   Function ShrinkNeighbors($N_{current}, Neighbors$):
     /* $ResultHeap$ is a max heap, in which the root element is the node furthest to $N_{current}$. An element of $ResultHeap$ is a tuple that can be compared based on its second
        attribute value.                                        */
25       $ResultHeap \leftarrow \emptyset$;
26       for $N_i \in Neighbors$ do
27           $Element \leftarrow (N_i, -N_i.P^0 \cdot N_{current}.Q')$;
28           $ResultHeap.push(Element)$;
29       $ResultHeap.pop()$;
30       return $ResultHeap$;
```

uses the secure NSW graph search algorithm to search the nearest node to the encrypted query vector in the top layer. Then, the cloud server takes it as the entry point of the next layer and follows the same process. The process is repeated until reaching *MaxLayer*.

During the insertion phase, the cloud server starts from the entry point of *MaxLayer* and searches the *ef* nearest nodes to the encrypted query vector. Then, the cloud server adds connections between the top *M* nearest nodes and the inserted node at current layer. Then, the cloud server takes the nearest node as the entry point of the next layer and follows the same process. The process is repeated until reaching the 0th layer. Note that during the insertion phase, we set a threshold *M* for the number of neighbors of every node at current layer to prevent search slowdowns. Specifically, when the number of neighbors of a node is greater than *M*, we use its encrypted query vector to get the top *M* nearest nodes as its neighbors. In particular, we set a special threshold $M_{layer0} > M$ for the 0th layer to improve search accuracy.

Search: When receiving an encrypted search request Q^0, the cloud server retrieves k similar encrypted images via the secure HNSW index graph. Figure 2 shows an example of a search using our secure index. The search algorithm also contains an accelerated search phase which is similar to that of the insertion algorithm. The only difference is that the accelerated search phase of the search algorithm stops when reaching the first layer. Then, the cloud server starts from the entry point found in the first layer and employs Algorithm 1 to search the *k* nearest nodes to Q^0 in the 0th layer. Finally, the cloud server will return the corresponding encrypted images of the *k* nodes to the query user.

Update: Our scheme supports two update operations: insertion and deletion. To insert a new image, the image owner encrypts the image, generates its encrypted index vector and encrypted query vector, and then uploads them to the cloud server. Finally, the cloud server inserts the encrypted data to the secure index by using Algorithm 2. Note that the image owner can also use the new image to fine-tune his/her CNN models as needed. To delete an image, the cloud server first finds the node that belongs to the image, and then deletes the connections between the node and its neighbors at each layer from its maximum layer to the 0th layer. Finally, the cloud deletes the node from the index. Unlike previous scheme [5]–[9], the update algorithms of our secure index not only allow image owners to be offline, but also can be done in parallel, since the update processes of different nodes rarely access the same neighbor nodes. The parallel updates can make better use of cloud resources to improve update efficiency.

V. SECURITY ANALYSIS

In this section, we analyze the security of the proposed scheme under the threat model described in Section III-B.

1) *Data Privacy:* Our scheme employs the AES algorithm to encrypt all the outsourced images. This makes it almost impossible for an attacker to obtain the contents of the outsourced images. The image features also need to be protected, since they may reveal some image contents. Our scheme employs the ASPE algorithm to encrypt every feature vector. Under the known ciphertext model, the cloud server or other attackers cannot learn any private information of the encrypted feature vectors as long as the secret key of the ASPE algorithm iskept confidential,as provedin [24].Besides, byusing the randomly splitting procedure, the ASPE algorithm also provides data unlinkability such that the encrypted results are different even for the same feature vector. Data unlinkability effectively prevents the cloud server or other attackers from learning some useful statistical information.

2) *Index Privacy:* Our secure index graph consists of a series of nodes. Each node need to permanently preserve an encrypted index vector and a corresponding encrypted query vector. The privacy of the encrypted feature vectors is well protected by the ASPE algorithm. However, the cloud server knows the neighbor relationships in our secure index graph. Almost all secure indexes leak the similar relationships between the encrypted images. The leakage of the similar relationships is a compromise for search efficiency.

VI. PERFORMANCE EVALUATION

In this section, we present the performance evaluations of our scheme. We investigate the performance of the proposed scheme on feature extraction, feature encryption, secure index outsourcing, search efficiency, update efficiency, and storage consumption. To demonstrate the performance of our scheme, we implement the feature extraction algorithm using pytorch [44] and the rest algorithms using C++. In previous schemes [5]–[9] based on feature encryption, SEISA [6] is the only one that provides the secure index outsourcing algorithm. We implement the secure index outsourcing, search, and update algorithms used in SEISA, and compare ours with them. To fairly compare the search accuracy of these two secure indexes, we use the same image features generated by our feature extraction algorithm to build these two secure indexes, respectively. We run the experiments on a computer running Windows 10 with AMD Ryzen5 1600X Six-Core Processor CPU @ 3.6 GHz, NVIDIA Geforce GTX 1070Ti GPU and 8 GB of RAM.

A. EVALUATION METRIC, DATASETS AND PARAMETERS

We use Precision at k (P@k) as the performance metric. We compute the P@k for each query as

$$P@k = \frac{\sum_{i=1}^{k} rel(i)}{k}, \qquad (2)$$

where k is the number of returned images; $rel(i)$ is an indicator function returning 1 if the image at rank i is a relevant image, and 0 otherwise.
We evaluate the proposed scheme on the CIFAR-10 dataset [45]. The CIFAR-10 dataset has a total of 60000 images, categorized into 10 distinct object categories. Each class consists of 5000 training images and 1000 testing images. In our experiments, the 50000 training images are used to fine-tune the CNN models to get feature extractors, and the 10000 testing images are used to evaluate the performance of our search and update algorithms.
Here, we give the parameter settings for our experiments. We investigate the performance of our schemes for different feature dimensions by setting the size h of the newly added latent layer of our CNN model to $\{64,128,256,512,1024\}$, respectively. For the HNSW graph algorithm, Malkov *et al.* [37] suggest that the reasonable range of the number of established connections M is 5-100. Increasing M leads to longer indexing time but better search accuracy. We choose M and M_{layer0} to be 10 and 20, respectively. The parameter *ef* of the insertion algorithm is set to 150 to get nicer search quality. The normalization factor m_L is set to $1/\ln(M)$ to achieve the optimum overlap between the neighbor nodes of the same node on different layers. For the secure hierarchical k-means index tree, we use the same settings as Yuan *et al.* [6]. Specifically, we set the parameter T to 100. That is, each time we can use the secure k-means outsourcing algorithm to cluster images into 100 groups.

B. SETUP EVALUATION

1) FEATURE EXTRACTION

We fine-tune the CNN models with different h on the training images. Each model is trained for 50,000 iterations. On the average, the process of fine-tuning a model takes 2 hours. The memory costs of the fine-tuned models are 218MB, 219MB, 221MB, 225MB, and 233MB, respectively as h varies. Increasing feature dimensions leads to little growth of memory cost. When the fine-tuned models work as feature extractors, on the average, the feature extraction process of one image takes 5ms.

2) FEATURE ENCRYPTION

The image owners and the query users employ the ASPE algorithm to encrypt their image features. The computation of generating an encrypted index vector or an encrypted search request is mainly two multiplications of an $(n+1)\times(n+1)$ matrix and an $(n+1)$-dimension vector. Figure 3 shows the encryption time w.r.t. different feature dimensions. As can be seen, increasing the feature dimensions results in more encryption time.

3) SECURE INDEX CONSTRUCTION

As stated in Section I, SEISA builds a hierarchical index tree by recursively using the secure k-means outsourcing algorithm. Thus it still requires participation from the image owners. On the other hand, by using the secure HNSW graph algorithm, our secure index construction algorithm allows the cloud server to build a secure index locally without requiring participation from the image owners.
Figure 3 shows the comparison of the construction cost of our scheme with that of SEISA on the CIFAR-10 dataset w.r.t. different feature dimensions. As can be seen, our scheme requires less construction time than SEISA. In particular,

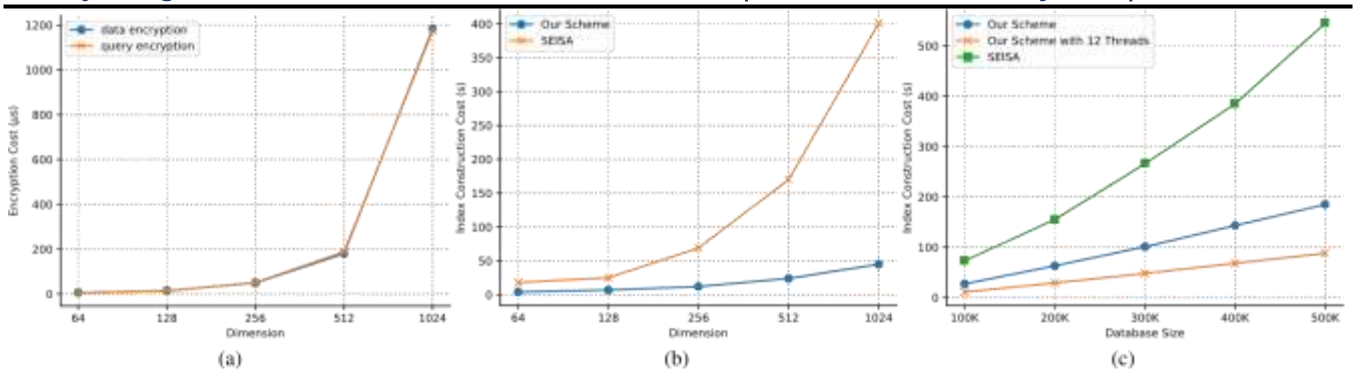

FIGURE 3. System setup cost: (a) Encryption cost of a feature vector of different dimensions; (b) Secure index construction cost on the CIFAR-10 dataset w.r.t. different dimensions; (c) Secure index construction cost on extended datasets with different sizes (the dimensions of feature vectors are fixed to 128).

when the image features have 1024 dimensions, the index construction time of SEISA is 8 times as long as ours. This is because their index outsourcing algorithm requires the image owners to re-encrypt the centroid vectors of clusters, and the encryption and decryption cost of high-dimensional feature vectors is expensive.

To validate the scalability of the proposed scheme, we expand the CIFAR-10 dataset by using Augmentor [46] which provides different image transformation operations, e.g., flipping, rotating and zooming. Specifically, we specify a probability for each operation we need and add these operations one by one to create an image generation pipeline. By executing the pipeline, we expand the dataset to different sizes ranging from 100K to 500K. We then feed the extended datasets to the fine-tuned CNN model to extract 128-dimensional feature vectors to evaluate the performance of our secure index construction algorithm. Figure 3 shows the comparison of the construction time of our scheme with that of SEISA on the five datasets. As can be seen, the construction time for the two schemes is proportionaltothesizeofthedatasets.Atthesametime,itisobvious that our scheme is faster than SEISA on the five datasets. And as the size of the datasets increases, the gap between the two schemes increases. This implies the potential of our scheme on even larger datasets. Besides, the performance of our index construction algorithm can be further improved by utilizing parallel computing as shown in Figure 3.

C. SEARCH EVALUATION

We evaluate the search cost of our secure index graph w.r.t. feature dimensions and the sizes of datasets. Figure 4 shows the comparison of the search cost of our scheme with that of SEISA on the CIFAR-10 dataset w.r.t. different feature dimensions when top 10 encrypted images are retrieved. And Figure 4 shows the search cost of the two schemes on the extended datasets when the image features have 128 dimensions and top 10 encrypted images are retrieved. From the figures, we can know that our scheme is a little slower than SEISA. This is because a secure hierarchical k-means tree can be regard as the optimal version of our secure index graph.

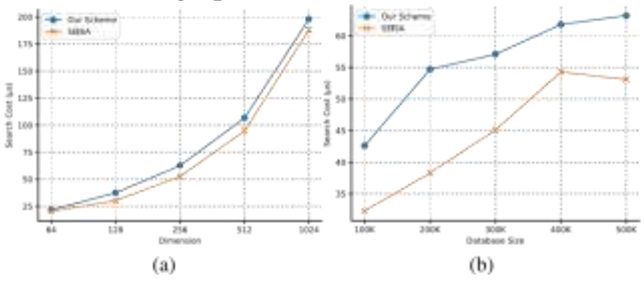

FIGURE 4. Search cost: (a) Search time on the CIFAR-10 dataset w.r.t. different dimensions; (b) Search time on extended datasets with different sizes (the dimensions of feature vectors are fixed to 128).

TABLE 1. Retrieval precision on the CIFAR-10 dataset when feature vectors have 128 dimensions and top k images are retrieved.

Scheme	k				
	200	400	600	800	1000
brute-force kNN	0.926	0.926	0.926	0.926	0.926
Our scheme ($M = 10$)	0.919	0.918	0.918	0.917	0.917
Our scheme ($M = 20$)	0.924	0.924	0.923	0.923	0.923
SEISA	0.926	0.926	0.926	0.926	0.926

In thehierarchical k-means tree, thenodes of higherlayersare obtained by calculating the mean vectors of their leaf nodes. However, in the HNSW graph, the nodes of higher layers are selectedrandomlyfromthelowerones.Thusoursecureindex graph may not be optimal. However, our scheme can still be practical on large datasets. In particular, when the dataset has 500K images, our scheme takes 63.2 μs to finish a 10-NN search and SEISA takes 53.1 μs.

To evaluate the accuracy of our scheme, we fix the dimensions of the image feature vectors to 128 and range the number of retrieved images from 200 to 1000. The search accuracy of the two schemes are reported in Table 1. As we can see, the search accuracy of our scheme is a little lower than SEISA. We argue that the small value of the parameter M leads to a high possibility of the search process falling into a local minimum and thus resulting in lower accuracy. Table 1 also shows that the search accuracy can be improved by increasing the value of M.

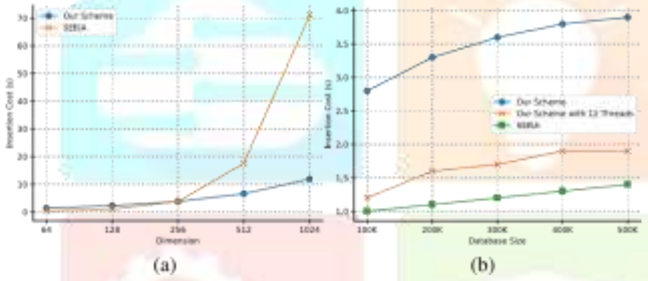

FIGURE 5. Insertion cost: (a) Insertion cost on the CIFAR-10 dataset w.r.t. different dimensions; (b) Insertion cost on extended datasets with different sizes (the dimensions of feature vectors are fixed to 128).

TABLE 2. Storage consumption on extended datasets with different sizes (the dimensions of feature vectors are fixed to 128).

secure index	Storage consumption (MB)				
	100K	200K	300K	400K	500K
Our scheme	426	852	1278	1705	2130
SEISA	223	431	664	903	1190

D. *UPDATE EVALUATION*

As described in [6], the update algorithms of SEISA require the image owners to re-encrypt $2(\log_T(N) - 2)$ feature vectors. On the other hand, our scheme allows the cloud server to update the secure HNSW index graph locally without any involvement from the image owners. Thus our scheme saves more computational and communicational cost for the image owners.

Figure 5 shows the comparison of the cost for our secure index to add 10,000 new images with that of SEISA on the CIFAR-10 dataset w.r.t. different feature dimensions. It can be seen that when the image features are high-dimensional, our scheme is much faster than SEISA. This is because the image owners in SEISA need to re-encrypt $2(\log_T(N) - 2)$ image features, and the encryption and decryption of the high-dimensional feature vectors are time-consuming.

Figure 5 shows the comparison of the cost for our secure index to add 10,000 new images with that of SEISA on the extended datasets when the features have 128 dimensions. As can be seen, our scheme is slower than SEISA. This is because the insertion algorithms of the two schemes both use their search algorithms, and our search algorithm is slower than their (as can be seen in Figure 4). Besides, the performance of our insertion algorithm can be greatly improved by using parallel computing as shown in Figure 5.

E. *STORAGE CONSUMPTION OF INDEX*

Table 2 shows the storage consumption of our scheme and that of SEISA on the extended datasets when the image features have 128 dimensions. Compared with SEISA, our scheme takes about 2× storage consumption. The main storage consumption of SEISA comes from the encrypted index vectors of the nodes of their index. However, each node in our scheme also needs to save an encrypted query vector. As described in our insertion algorithm, the encrypted query vector of a node is used to make the number of neighbors of the node not exceed M and M_{layer0}.

VII. CONCLUSION

In this paper, we present an encrypted image retrieval scheme that significantly reduces the computational and communicational cost of the image owners when building the secure indexes. We use short CNN feature vectors to obtain higher search accuracy and lower storage. To ensure high search speed and reduce the cost of the image owners, a secure HNSW index graph is built locally by the cloud server. In the research line based on feature encryption, we believe that this is the first scheme that enables cloud servers to build and update secure indexes locally without participation from image owners. Our secure index can also be used as a building block in other secure retrieval fields. We have confirmed that the proposed scheme is secure against the known ciphertext model. The experimental results on the CIFAR-10 dataset show that our scheme is efficient. In the future, we intend to further improve the efficiency of our scheme. Transforming high-dimensional CNN features into short binary codes is the trend of image retrieval. Thus, efficient encryption schemes that support computation of Hamming distance on encrypted data will be our focus.

REFERENCES

[1] V. V. Estrela and A. E. Herrmann, "Content-based image retrieval (CBIR) in remote clinical diagnosis and healthcare," in *Encyclopedia E-Health Telemedicine*. Hershey, PA, USA: IGI Global, 2016, pp. 495–520.

[2] Y. Ma, L. Wu, X. Gu, J. He, and Z. Yang, "A secure face-verification scheme based on homomorphic encryption and deep neural networks," *IEEE Access*, vol. 5, pp. 16532–16538, 2017.

[3] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2015, pp. 27–35.

[4] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *J. Netw. Comput. Appl.*, vol. 79, pp. 88–115, Feb. 2017.

[5] W. Lu, A. Swaminathan, A. L. Varna, and M. Wu, "Enabling search over encrypted multimedia databases," *Proc. SPIE*, vol. 7254, Feb. 2009, Art. no. 725418.

[6] J. Yuan, S. Yu, and L. Guo, "SEISA: Secure and efficient encrypted image search with access control," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 2083–2091.

[7] X. Li, Q. Xue, and M. C. Chuah, "CASHEIRS: Cloud assisted scalable hierarchical encrypted based image retrieval system," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[8] 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," *Inf. Sci.*, vol. 387, pp. 195–204, Dec. 2017.

[9] J. Qin, H. Li, X. Xiang, Y. Tan, W. Pan, W. Pan, W. Ma, and N. N. Xiong, "An encrypted image retrieval method based on Harris corner optimization and LSH in cloud computing," *IEEE Access*, vol. 7, pp. 24626–24633, 2019.

[10] C. Y. Hsu, C. S. Lu, and S. C. Pei, "Image feature extraction in encrypted domain with privacy-preserving SIFT," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4593–4607, Nov. 2012.

[11] Y. Bai, L. Zhuo, B. Cheng, and Y. F. Peng, "Surf feature extraction in encrypted domain," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2014, pp. 1–6.

[12] Q. Wang, L. Gao, H. Wang, and X. Wei, "Face detection for privacy protected images," *IEEE Access*, vol. 7, pp. 3918–3927, 2019.

[13] B. Ferreira, J. Rodrigues, J. Leitão, and H. Domingos, "Practical privacypreserving content-based retrieval in cloud image repositories," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 784–798, Jul./Sep. 2019.

[14] H. Cheng, X. Zhang, J. Yu, and F. Li, "Markov process-based retrieval for encrypted jpeg images," *EURASIP J. Inf. Secur.*, vol. 2016, no. 1, 2016, Art. no. 1.

[15] H. Cheng, X. P. Zhang, J. Yu, and Y. Zhang, "Encrypted JPEG image retrieval using block-wise feature comparison," *J. Vis. Commun. Image Represent.*, vol. 40, pp. 111–117, Oct. 2016.

[16] Z. Xia, L. Lu, T. Qiu, H. J. Shim, X. Chen, and B. Jeon, "A privacypreserving image retrieval based on AC-coefficients and color histograms in cloud environment," *Comput., Mater. Continua*, vol. 58, no. 1, pp. 27–44, 2019.

[17] Z. Xia, X. Ma, Z. Shen, X. Sun, N. N. Xiong, and B. Jeon, "Secure image LBP feature extraction in cloud-based smart campus," *IEEE Access*, vol. 6, pp. 30392–30401, 2018.

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1999, pp. 223–238.

[19] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.

[20] Y. Wu, J. Hou, J. Liu, W. Zhou, and S. Yao, "Novel multi-keyword search on encrypted data in the cloud," *IEEE Access*, vol. 7, pp. 31984–31996, 2019.

[21] C. Hu, X. Song, P. Liu, Y. Xin, Y. Xu, Y. Duan, and R. Hao, "Forward secure conjunctive-keyword searchable encryption," *IEEE Access*, vol. 7, pp. 35035–35048, 2019.

[22] C. Cai, X. Yuan, and C. Wang, "Hardening distributed and encrypted keyword search via blockchain," in *Proc. IEEE Symp. Privacy-Aware Comput. (PAC)*, Aug. 2017, pp. 119–128.

[23] A. Kelarev, X. Yi, S. Badsha, X. Yang, L. Rylands, and J. Seberry, "A multistage protocol for aggregated queries in distributed cloud databases with privacy protection," *Future Gener. Comput. Syst.*, vol. 90, pp. 368–380, Jan. 2019.

[24] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.

[25] Q. Wang, M. He, M. Du, S. S. M. Chow, R. W. F. Lai, and Q. Zou,
"Searchable encryption over feature-rich data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 496–510, May/Jun. 2018.

[26] M. He, J. Zhang, G. Zeng, and S. M. Yiu, "A privacy-preserving multipattern matching scheme for searching strings in cloud database," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 293–302.

[27] A.Gionis,P.Indyk,andR.Motwani,"Similaritysearchinhighdimensions via hashing," *VLDB*, vol. 99, no. 6, pp. 518–529, 1999.

[28] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "Deepcrack: Learning hierarchical convolutional features for crack detection," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1498–1512, Mar. 2019.

[29] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1224–1244, May 2018.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[31] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrievalviaimagerepresentationlearning,"in*Proc.28thAAAIConf.Artif. Intell.*, Jun. 2014, pp. 2156–2162.

[32] A. Krizhevsky and G. E. Hinton, "Using very deep auto encoders for content-based image retrieval," in *Proc. ESANN*, vol. 1, 2011, p. 1–7.

[33] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 584–599.

[34] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[35] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Approximate nearest neighbor algorithm based on navigable small world graphs," *Inf. Syst.*, vol. 45, pp. 61–68, Sep. 2014.

[36] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces," in *Proc. Int. Conf. Similarity Search Appl.* Berlin, Germany: Springer, 2012, pp. 132–147.

[37] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[39] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1556–1564.

[40] D. Wu, Z. Lin, B. Li, M. Ye, and W. Wang, "Deep supervised hashing for multi-label and large-scale image retrieval," in *Proc. ACM Int. Conf. Multimedia Retr.*, Jun. 2017, pp. 150–158.

[41] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1351–1362, May 2016.

[42] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 201–210.

[43] O.-A. Kwabena, Z. Qin, T. Zhuang, and Z. Qin, "MSCryptoNet: Multischeme privacy-preserving deep learning in cloud computing," *IEEE Access*, vol. 7, pp. 29344–29354, 2019.

[44] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS*, Oct. 2017, pp. 1–4.

[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, USA, Tech. Rep., 2009, vol. 1, no. 4.

[46] M. D. Bloice, C. Stocker, and A. Holzinger, "Augmentor: An image augmentation library for machine learning," Aug. 2017, *arXiv:1708.04680*. [Online]. Available: https://arxiv.org/abs/1708.04680