Privacy-Preserving Image Retrieval Based on Additive Secret Sharing

Zhihua Xia, Member, IEEE, Qi Gu, Lizhi Xiong, Wenhao Zhou, Jian Weng, Member, IEEE

Abstract—The rapid growth of digital images motivates individuals and organizations to upload their images to the cloud server. To preserve the privacy, image owners would prefer to encrypt the images before uploading, but it would strongly limit the efficient usage of images. Plenty of existing schemes on privacy-preserving Content-Based Image Retrieval (PPCBIR) try to seek the balance between security and retrieval ability. However, compared to the advanced technologies in CBIR like Convolutional Neural Network (CNN), the existing PPCBIR schemes are far deficient in both accuracy and efficiency. With more cloud service providers, the collaborative secure image retrieval service provided by multiple cloud servers becomes possible. In this paper, inspired by additive secret sharing technology, we propose a series of additive secure computing protocols on numbers and matrices with better efficiency, and then show their application in PPCBIR. Specifically, we extract CNN features, decrease the dimension of features and build the index securely with the help of our protocols, which include the full process of image retrieval in plaintext domain. The experiments and security analysis demonstrate the efficiency, accuracy and security of our scheme.

Index Terms—Privacy-preserving Image Retrieval, Additive Secret Sharing, Pre-trained CNN, Secure PCA, Secure Index Building

1 INTRODUCTION

Recent years have witnessed the explosive growth of personal image, and services like CBIR are utilized by more and more people with the help of cloud computing. Accordingly, the CBIR technologies attract plenty of researchers. The schemes on feature extraction [1] from images and index building [2] for large-scale vectors have been researched in detail and these excellent schemes have been put into practice like Google search by image.

It should be noticed that images contain rich sensitive information in many cases. For instance, patients would not like to expose their medical images. As the Cloud Server (CS) is always not fully trusty, it is unsuitable to upload the plaintext images to CS directly. Therefore, more and more researchers pay their attention to PPCBIR. The PPCBIR schemes should not only keep the accuracy and efficiency of image retrieval, but also ensure the images and their features will not be exposed under the reasonable security assumption.

However, due to the security restrictions, the accuracy and efficiency of existing PPCBIR schemes are much lower than works [3] in CBIR. For instance, the features used in PPCBIR either put a heavy burden [4] on the image owner or show inferiority [5] during the retrieval. The main reason for the inferiority is the plenty of non-linear computation appeared in the state-of-the-art CBIR schemes. It should be noticed that the traditional functional encryption tools are quite low efficient when facing these computations.

With more and more different cloud server manufacturers are willing to provide cloud computing services, the service provided by two non-collusion CS attracts many researchers in recent years. The attendance of two non-collusion CS can avoid the utilization of traditional high-complexity functional encryption methods, which gives a new way to decrease the consumption of secure retrieval.

Additive secret sharing is a typical technology which widely used in Secure Multi-party Computation (SMC). Recently, more researchers utilize this technology to execute their tasks [6]-[8] under the environment of cloud computing. Especially, recent work [9] constructs protocols that can support the secure computation on almost all basic elementary functions in constant-rounds interaction based on additive secret sharing. However, the existing schemes always ignore secure matrix computation and cause unnecessary communication complexity for ensuring the information-theoretic security of input.

Accordingly, in this work, we aim to bridge the gap between PPCBIR and CBIR based on additive secret sharing technology. We first construct some basic secure computation protocols based on two non-collusion servers, then show how to utilize them in PPCBIR. In summary, we mainly make the following contributions:

1) We construct a series of novel protocols based on additive secret sharing. Especially, besides a more efficient secure comparison method, the computation protocol on matrices like matrix inversion is also designed to support complex computation.

2) Secure CNN features are extracted as visual descriptors for image retrieval from a pre-trained VGG16 by using the proposed protocols. Due to the efficiency of our protocols, our scheme consumes much less time than a recent CNN feature extraction method [6].

3) Based on the secure matrix inversion protocol, we
2 RELATED WORK

We introduce the existing work related to this paper from the perspective of CBIR and PPCBIR.

2.1 Content-Based Image Retrieval

In the early stage of CBIR, the global features \(^{10}\), such as color and texture, are firstly extracted by researchers. However, as the global feature is fragile to the influence like illumination or rotation. Benefiting from the scale-space theory, local features \(^{11}\) (e.g., SIFT) are proposed and show a great advantage to global features. Some following studies firstly use typical -means to divide all the data into \(N\) categories. Then, for each category, if the vectors in the current category are too many, the server will use \(k\)-means to split this category again. The index building process will be executed until the number of vectors in each category is less than a fixed number.

Hash-based schemes follow the idea that the neighbor points in high dimension will still be neighbors when they are mapped into the low dimension. They build a reasonable mapping method called Locality Sensitive Hash (LSH) \(^{15}\) for common distance measurements (e.g., Euclidean distance). One typical LSH method called C2LSH (Count Collusion LSH) \(^{19}\) proposed the concept of dynamic collision and virtual rehash, which got significant storage and accuracy advantage. Briefly speaking, they construct a series of LSH mapping functions in the same LSH family, then use these functions to map all vectors in the database. The vectors which have higher collision frequency will have the chance to participate in actual distance computation. We will describe more calculation details of HKM and C2LSH in subsection 3.3.

In recent years, great progress has been made in the PQ-based and graph-based schemes, which make billion-scale vector retrieval practical. Although the motivation behind these methods is far different from the former two categories, however, the actual calculation is similar (e.g., multiplication, comparison, etc.). For simplicity and generality, in this paper, we choose two typical methods HKM and C2LSH as examples and apply them in encrypted image retrieval.
feature vectors index building. It will be easy to notice that other ANNS schemes are also able to incorporate into our work.

2.2 Privacy-preserving Content-Based Image Retrieval

For security considerations, the PPCBIR task has attracted more and more researchers in this decade. Different from CBIR, the schemes in PPCBIR need to protect the image and its feature for the owner of the image and provide the CBIR service to authorized users with the help of the cloud server.

Existing schemes on PPCBIR can be classified into two categories: feature-encryption based schemes and image-encryption based schemes. In the first category [1, 23, 24], the image owner needs to extract the features from plaintext images and builds an index for them. Then the owner uploads encrypted images and a corresponding index to the cloud server. Since the feature extraction and index building are always resource-consuming tasks, it is not appropriate for leaving them to the image owner.

In this case, schemes [5, 25] in the second category, which outsources these tasks, attract more researchers in recent years. In this category, the image owner only needs to encrypt his images and outsources them to CS. The key problem here is how to extract the valid feature from an encrypted image. In this case, schemes in the second category can be classified into two classes based on the type of encrypted features that CS extracts from the encrypted images. The detailed information is described as follows.

2.2.1 schemes based on statistic feature

The schemes in the first class extract statistic histograms as the encrypted image feature. The main method here is protecting the images and their features by permutation and keeping valid distance between encrypted images. Ferreira et al. [26] proposed a tailor-made encryption scheme for images. In this scheme, the image color is protected by value substitution and image texture is shuffled randomly by rows and columns. After uploading, the HSV (Hue-Saturation-Value) color histograms are extracted from the encrypted images at the CS side. The index building will be further built by CS. It should be noted that both features and the index are in the encrypted domain, however, the Manhattan distance between features is still same as the distance in plaintext state. Since the global feature is too weak for the retrieval, Xia et al. [5] further propose a scheme called BOEW which considers the encryption in each block and ensure that the encrypted local features are valid for retrieval. The server further aggregates the encrypted local features with the typical BOW model which greatly improves retrieval efficiency.

As the encryption in spatial domain malfunctions the compression, that is, it will greatly increase the size of encrypted images. Therefore, some schemes [25, 27] try to execute PPCBIR in JPEG-domain. These methods encrypt the image in JPEG-format and ensure the JPEG-format be kept after the encryption. The encryption methods used are almost similar (e.g., permutation) to the schemes in spatial domain. It should be noted that the above schemes are all based on probabilistic encryption, which makes the security of their schemes depends on the size of image content, and the statistic feature shows a mediocre performance for retrieval.

2.2.2 schemes based on typical feature

The second class tries to extract typical features (e.g., SIFT) from the encrypted images. Different from the statistic feature, plenty of linear, nonlinear, and comparison computations appear during the extraction process of typical features. To our knowledge, Hsu et al. [28] first investigate privacy-preserving SIFT extraction in the encrypted domain based on the Homomorphic Encryption (HE) [29] tool which supports direct computation on the ciphertext. However, their scheme suffers from high interaction rounds and potential security risks. The following schemes [30] notice that one single server is hard to cope with these problems. They further try to use multiple servers (e.g., two servers) to collaboratively execute the encrypted SIFT feature extraction. For example, Hu et al. [31] combine the Somewhat HE and parallel technology, propose batched secure multiplication and comparison protocol between two servers. However, these schemes are still too time-consuming rather than practice. In recent work, Liu et al. [33] try to extract encrypted features based on a pre-trained CNN model (e.g., VGG16). The better feature extractor brings much higher accuracy.

It should be noted that the encrypted feature extraction by pre-trained CNN is equivalent to the following problem: how to execute the inference process of CNN with known parameters in safety. There are many schemes [6] try to handle this fundamental problem in recent years. The HE tool is also considered first. However, due to the nonlinear computation or comparison in the activation layer and pooling layer, the basic HE can not satisfy demands. There are two main strategies to cope with this problem: one type of schemes [34] tries to use SMC technology, such as garbled circuit [38], to assist nonlinear operations; the others [56] attempt to seek approximate algorithms to replace them. Actually [33] can be seen as the utilization of [34]. However, these schemes not only cause plenty of time consumption but also need to convert ciphertext to an integer (or bits), which will cause extra precision loss.

To alleviate the problem, Huang et al. [6] proposed a scheme completely based on additive secret sharing which naturally supports linear computation on the ciphertext. They further construct a secure comparison protocol based on Beaver’s triples [37]. The avoidance of HE makes the scheme obtain a great advantage in time consumption, however, the protocols they proposed still have lots of room for improvement.

In this paper, we propose a novel PPCBIR scheme based on pre-trained CNN. Different from the previous low-efficiency schemes, we first propose enough novel computation protocols based on additive secret sharing, then simulate the feature extraction in state-of-the-art CBIR schemes, and further propose secure feature compression and index building technologies to decrease the consumption. The utilization of these strategies makes our scheme both efficient and accurate.
3 Preliminaries

3.1 Image feature extracted by pre-trained CNN

In this work, the feature is extracted from a typical CNN model called VGG16, and the parameters pre-trained by a large image dataset [1] are utilized. CNN is a sequence of layers that compute and transform the input into output. Each layer is made up of a set of neurons. Like most of the other CNN models, VGG16 is composed of Fully Connected layer (FC), Convolutional layer (Conv), Activation Layer (AL), and Pooling layer (Pool). In detail, each layer contains the following computation:

- **FC and Conv.** FC and Conv layer only involve the linear computation. For example, the neurons in Conv layer share the same weights $W$ and biases $b$, which are often called filters. Consider a filter sizes $n \times n$, then each neuron in the current layer is transformed from an $n \times n$ region of neurons in the previous layer. In detail, the $(j, k)^{th}$ neuron is obtained from $y_{j,k} = \sum_{i=0}^{n-1} \sum_{m=0}^{n-1} w_{i,j}x_{i+k,j+m} + b$. Here, $w_{i,j}$ means the corresponding value in weights matrix $W$ and $x$ represents the neuron in the previous layer. It should be noted that the $W$ is fixed when we use pre-trained CNN. As we disuse the FC layer in the VGG16, our scheme can deal with images in any size.

- **AL.** Activation layer provides the non-linear computation in the neural networks, and Rectified Linear Unit (ReLU) is the unique AL in VGG16. For the input matrix $X$, the ReLU layer returns the result $ReLU(X)$ composed by $max(x, 0)$, where $x$ is the member value in $X$. It is easy to notice that ReLU only involves comparison computations.

- **Pool.** The pooling layer partitions the input layer into a set of non-overlapping rectangles, then a down-sampling operation is executed on each rectangle to get the value in the current layer. In VGG16, max-pooling sizes $2 \times 2$ is utilized, which means the maximum number will be saved in each $2 \times 2$ block and transmitted to the later layer. Max-pooling also only contains the comparison computations.

3.2 Additive Secret Sharing

Secret sharing is one of the most important technologies in SMC. The secret information will be randomly split into multi-shares, and each participant only has one share. The secret can be reconstructed only when a sufficient number of shares are combined. Additive secret sharing can be seen as an $(n, n)$-threshold secret sharing technology. It means that the secret $x$ can be randomly split into $n$ shares, for example, $x = x_1 + x_2 + \ldots + x_n$, and recover the secret needs all $n$ share, here $n \geq 2$. In this paper, we only focus on the situation that $n = 2$.

In recent years, more researchers try to use the additive secret sharing technologies to outsource computation tasks to CS. For example, the image owner can outsource his image by randomly splitting the image into two additive shares, and each cloud server $S_i$ ($i = 1, 2$) owns one share. For simplicity, the following $S_i$ are all represents servers $S_1$ and $S_2$, similarly, the following values, with $i$ in subscript, represent the additive share of corresponding value if $i$ is undefined. It should be noted that the additive secret sharing has an important property: each server $S_i$ can execute linear computation without interaction. For example, each server owns $u_i, v_i$, and they want to compute $u \pm v$. Due to $(u_1 \pm v_1) + (u_2 \pm v_2) = u \pm v$, only needs to compute $u_i \pm v_i$, and the result is still the share of $u \pm v$. Similarly, each server can execute the secure multiplication by a known number (e.g., pre-trained parameters).

*Beaver's triples* is widely used in additive secret sharing to execute the secure multiplication called $\text{SecMult}$. During the offline phase (i.e., before computing secure multiplication), a pre-computed triple of the form $(a_i, b_i, c_i) = (ab = c)$ is generated and $(a_i, b_i, c_i)$ is sent to server $S_i$. Here, $(a_i, b_i, c_i)$ is the additive share of $(a, b, c)$. In online phase, each server computes $e_i = x_i - a_i$ and $f_i = y_i - b_i$. Then two server exchange the information of $e_i$ and $f_i$. Finally, $S_i$ can compute $z_i = f_i + c_i + (i - 1)ef$. It is easy to notice that $z_1 + z_2 = xy$. The existing schemes [38] further expand the secure scalar multiplication to secure matrix dot product as shown in algorithm 1.

Although the fixed size vectors need to be computed during offline phase in algorithm 1, we would show that it is feasible when facing a specific task in subsection 3.3.2. In recent work, Xiong et al. [9] further expands *Beaver’s triples*, and constructs protocols which can support almost all basic elementary function and four fundamental operations with only constant-rounds interaction rounds. In the rest of the paper, an additive share is called a share for simplicity.

3.3 HKM and C2LSH

Excessive comparison operations will lead to low retrieval efficiency, in this case, a reasonable index is indispensable. In PBCIR schemes based on typical features, these methods can not only accelerate the retrieval, but also significantly reduce the communication consumption that will be shown in subsection 8.3.1. For simplicity, the HKM and C2LSH are chosen to represent tree-based and hash-based index building schemes. As we focus on the way which employs these technologies when feature vectors are stored as a share in two servers, only the numerical calculation process on feature vectors is given below. The readers can refer to [18] and [19] for more details (e.g., virtual rehash) about these schemes.

3.3.1 Index building and query of HKM

As described in subsection 2.7, there are mainly two steps in the index building process of HKM:

### Algorithm 1 Secure matrix multiplication $\text{SecMatMul}$

**Input:** $S_i$ has $X_i, Y_i$.

**Output:** $S_i$ gets $(X \cdot Y)_i$.

**Offline Phase**:

1. $T$ generates random matrix $A, B$ and computes $C = A \cdot B$.
2. $T$ randomly splits $(A, B, C)$ into two additive share $(A_i, B_i, C_i)$ and sends the share to corresponding server $S_i$.

**Online Phase**:

3. $S_i$ computes $E_i = X_i - A_i$ and $F_i = Y_i - B_i$.
4. $S_1$ and $S_2$ exchange the share $E_i$ and $F_i$.
5. $S_i$ computes $X_i \cdot F + E \cdot Y_i + C_i - (i - 1)E \cdot F$. 


As described in subsection 2.1, there are mainly two steps during the retrieval based on C2LSH:

(i) **Index vectors by k-means.** The server divides all the vectors into k different categories by k-means first. There are three main steps during executing k-means: firstly, k random vectors are chosen as the centroid vectors; secondly, each vector computes their distance to centroid vectors and joins in the nearest category; finally, the centroid vectors will be updated as the mean value of new vectors in the current category. The second and third steps will be executed iteratively until a fixed number of times or the centroid vectors becomes stable.

(ii) **Repeat k-means partition.** After initial partitioning, the vectors in each subspace may still be too many. In this case, to ensure the number of vectors in each subspace less than a fixed number, the server will execute k-means on the corresponding subspace where excessive vectors exist. The partitioning will not end until each subspace meets the requirements.

After the index building, each vector will become one leaf node of the HKM tree like Fig. 1. Here, non-leaf nodes (except the root node) are the centroids of corresponding space.

During the retrieval, to achieve high efficiency and accuracy, the server will choose enough suitable vectors to compose candidates. The true distances between query and candidates will be computed and m-nearest vectors will be selected from them. Generally speaking, the number of vectors in candidates is usually over m (e.g., 3 times of m), in this case, the core challenge is how to seek enough and high-quality candidates. There are mainly two steps in HKM after the server gets the query vector:

(i) **Seek the nearest vectors.** The server computes distances between query and centroids in the first layer. Then the server enters the subspace nearest node represents and adds all other brother nodes into the candidate list. The above process will be repeated until the server finds leaf nodes.

(ii) **Get enough candidate vectors.** In HKM, the feature vectors that the leaf nodes in the above step represent will be all added into candidates. If not enough, the server will set the nearest node in the candidate list as the root node and repeat the above step to get new leaf nodes.

### 3.3.2 Index building and query of C2LSH

As described in subsection 2.1, there are mainly two steps in the index building process of C2LSH:

(i) **Generate hash functions.** The LSH family in C2LSH is shown as follows:

\[
h_{\vec{a},\vec{b}}(n) = \lfloor \frac{h_{\vec{a},\vec{b}}(\vec{q})}{w} \rfloor,
\]

\[
h_{\vec{a}}(n) = \vec{a} \cdot \vec{o} + bwx,
\]

where \(\vec{o}\) is a feature vector to be mapped, \(\vec{a}\) is the same dimension vector where each entry is drawn independently from the standard normal distribution \(N(0, 1)\), \(w\) is a user-specified constant (e.g., 1), \(b\) is a real number uniformly random drawn from \([0, w]\) and \(x\) is a positive random integer. It is easy to notice that the hash function in C2LSH is composed of \((\vec{a}, b, w, x)\). The server will generate a series of LSH functions in this family.

(ii) **Map vectors into LSH tables.** After generating the LSH functions, the server will map all the vectors in the database to a series of LSH tables. Each LSH function will generate a corresponding LSH table as shown in Table 1. If the \(\{h_{\vec{a},\vec{b}}(n)\}\) of feature vector \(\vec{o}\) equal to bid, then the corresponding image identity will be added into bucket bid in the j-th LSH table, here \(j \in [1, m]\), and \(m\) is the number of generated LSH functions.

| Bucket identity | Image identities in current bucket |
|-----------------|-----------------------------------|
| bid₁            | ID₁₁, ID₁₃₅, ID₁₄₃, ID₁₄₆       |
| bid₂            | ID₃₁, ID₁₃₁, ID₁₄₂               |
| \(\ldots\)      | \(\ldots\)                       |
| bidₙ            | ID₁₂₃, ID₂₃₁, ID₁₃₈, ID₂₂₄, ID₂₃₅ |

There are mainly two steps during retrieval based on C2LSH:

(i) **Map query into LSH tables.** Same as the vectors in the database, server will use all LSH functions to map the query vector \(\vec{q}\) into corresponding LSH tables. Then the vectors in the same bid in any LSH table will be seen as one collision with the query. The collision information will be stored, and the vector will be selected into candidates if it reaches a certain number of collisions.

(ii) **Get enough candidate vectors.** Similarly, the vectors in current bid may be insufficient. C2LSH further propose a scheme called virtual rehash to seek new bids for further collision count and the calculation during virtual rehash only needs \(h_{\vec{a},\vec{b}}(\vec{q})\) in each LSH table. \(\textbf{[19]}\) gives two ways to stop seeking candidates: the first one stops when \(m\) or more vectors whose distance from the query vector is less than a specific value are found; the second one stops when enough (e.g., \(3 \times m\)) vectors reach a certain number (related to the number of vectors in the database) of collisions, where \(m\) is the number of vectors returned.

### 4 Proposed System Architecture

#### 4.1 System model

Similar to \(\textbf{[31]}\) and \(\textbf{[33]}\), the proposed scheme involves five entities, i.e., the image owner, the cloud server \(S₁\), cloud server \(S₂\), a trusted party \(T\) and authorized users. The system model is shown in Fig. 2.

**Image owner** owns a large-scale database \(D = \{D₁\}^n_{i=1}\) with the corresponding identity set \(ID = \{ID₁\}^n_{i=1}\) to be
outsourced, here \( n \) represents the number of images. To preserve privacy, the images are all randomly split into two parts \( C_1 = \{ C_1^i \}_{i=1}^n \) and \( C_2 = \{ C_2^i \}_{i=1}^n \) based on additive secret sharing. The encrypted image databases are separately sent to \( S_1 \) and \( S_2 \).

**Cloud server** \( S_1 \) and \( S_2 \) undertake the task of feature extraction, feature compression, index building, and secure retrieval. Two cloud servers need to interact with each other and collaboratively complete the above tasks.

**Trusty third party** called \( T \) takes the charge of generating random numbers or random vectors which will be used during the secure computation. It should be noted that the generation of numbers is offline and lightweight work. The \( T \) can be undertaken by the government or the owner of images, we will further discuss this problem in subsection 8.3.2.

**Authorized user** authorized by the image owner has the authorization to retrieve the corresponding images. During the retrieval, user sends the trapdoor to two servers and gets the encrypted images from the servers. The plaintext results can be obtained by simply adding the corresponding encrypted versions.

### 4.2 Security Model

Similar to many previous works [31] in PPCBIR based on typical features, the semi-honest but non-collusion CS is considered in our scheme. It means each server will execute the protocol as asked but will try to analyze the information from data, but due to their reputation and financial interests, they will not collude with each other. As described above, the third party \( T \) only needs to generate random numbers in the offline phase, which means it can be undertaken by a low-computation device. Therefore, it is reasonable to assume that \( T \) is honest and non-colluding.

## 5 Basic secure computation based on two server

### 5.1 Secure Comparison

The typical comparison schemes [6] based on additive secret sharing try to use the most significant bit, however, it will lead to \( l \) rounds interaction, where \( l \) is the bit-length of ciphertext. Inspired by multiplicative secret sharing, recent work [9] proposed a novel comparison protocol that only needs three rounds of interaction. These schemes try to ensure the information-theoretic security of input, however, in the computation process of a practical task, plenty of intermediate values are practically meaningless. It means even though we do not protect some insignificant information (e.g., the range of value or the ratio between values), a specific task will not suffer actual damage.

The following phenomenon should be noted: if a number is multiplied by a random number (except 0), the result is also random. This means that it is impossible to infer the input number from the result. At the same time, multiplying a random positive number will not change the sign of the original data. Following this observation, a simple but valid secure comparison process for numbers can be shown in algorithm 2.

Each server firstly computes the share of \( u - v \), then the servers compute the multiplication of \( u - v \) and a random positive number which is generated by servers independently. If the result \( f = f_1 + f_2 \) is positive, it implies that \( u - v \) is also positive, and vice versa. We will analyze the security and potential information leakage of this protocol in section 7. It is easy to expand \( \text{SecCmp} \) to compute ReLU function, which is shown in algorithm 3.

### 5.2 Secure Matrix Inverse

The motivation of \( \text{SecCmp} \) is transforming the number, which needs protection, to another random number in safety and then re-share the new number. At the same time, the complex computation we execute on the new number can be reflected in the original number after a reasonable transform. In this way, the complexity of operations like...
comparison can be avoided. This motivation can be further applied to the matrix computation. Inspired by [39], we propose the secure matrix inverse protocol as algorithm 4.

In SecMatInv, each server first generates a random square matrix, then servers compute its dot product with the input. As the property that \((Z \cdot X)^{-1} \cdot (Z_1 + Z_2) = X^{-1}\), each server can finally get the share of \(X^{-1}\). As the sum of the random matrix generated by two servers is almost certainly (i.e., the probability is 1) reversible, for simplicity, we here ignore the potential risk brought by the random matrix. It should be noted that this situation can be detected by calculating the rank of \(W\). Especially when the matrix is simplified as a number, the secure division computation SecDiv is shown as algorithm 5.

6 PRIVACY-PRESERVING CBIR BASED ON TWO SERVERS

In this section, we firstly give the feature extraction process based on additive secret sharing, then further show the PCA compression method on encrypted features. Finally, the secure index building methods based on HKM and C2LSH are given for better retrieval efficiency. An overview of the process is shown in Fig. 3.

6.1 Secure Pre-trained CNN Feature extraction and aggregation

In this paper, following the suggestion in [14], the last activation layer before FC in pre-trained VGG16 is utilized as a feature extractor. As subsection 4.1 shows, the CNN model this paper involves mainly contains three different types of layers called \(\text{Conv}, \text{AL},\) and \(\text{Pool}\).

As the parameters or hyper-parameters are fixed and known by both two servers, which means all the weights are just constant in the CNN. For \(\text{Conv}\), it is just the secure constant multiplication which means no interactions are necessary during the inference process.

ReLU is unique \(\text{AL}\) in VGG16, as algorithm 3 shows, two rounds of interaction are needed between the servers. The values in feature vectors larger than 0 are unchanged, and the others will be reset as 0 by both \(S_i\).

The max-pooling needs to seek the position of the maximum value in each \(2 \times 2\) block, due to the comparison of different values can be executed simultaneously, only two rounds of interaction will be necessary for finding the position. Similar to ReLU, the number in the corresponding position is kept the same, and other numbers will be reset to 0 by \(S_i\).

After the extraction, following the conclusion in [17], the average aggregation is used to aggregate all the numbers gotten by each filter. Obviously, no interaction is needed during the aggregation. Here, each server gets a 512-dim encrypted feature vector for each image.

6.2 Secure Feature compression

PCA is the most commonly used feature compression method [3] for accelerating the retrieval. Here, we try to propose secure PCA computation on the encrypted features collaboratively stored in two servers. Note that the compression can significantly reduce communication consumption during secure retrieval.

To our knowledge, no existing schemes try to execute the secure PCA based on additive secret sharing, the essential problem here is the way in computing eigenvectors of the matrix in safety. The following Lemmas should be noticed in this case:

\[\text{Lemma 1. If } A \text{ and } B \text{ are similar matrices (i.e., } A \sim B), \text{ and } B = P^{-1}AP, \text{ then } A \text{ and } B \text{ have the same eigenvalues; If there is an eigenvector } \vec{x} \text{ under eigenvalue } \lambda \text{ of matrix } A, \text{ then } P^{-1}\vec{x} \text{ is an eigenvector of } B \text{ under eigenvalue } \lambda.\]

\[\text{Lemma 2. If } \lambda \text{ is an eigenvalue of matrix } A, \text{ then } k \times \lambda \text{ is an eigenvalue of matrix } k \times A; \text{ If there is an eigenvector } \vec{x} \text{ under eigenvalue } \lambda \text{ of matrix } A, \text{ then the eigenvector } k \vec{x} \text{ is also under eigenvalue } k \times \lambda \text{ of matrix } k \times A.\]

Based on Lemma 1 and 2, algorithm 6 shows the scheme of PCA on an encrypted matrix. Each server first executes zero-centering for each column of the matrix \(X_i^{n \times d}\), the \(\vec{m}_i\) composed by mean values will be stored in the server.

Then, each server will randomly generate a matrix \(P_i\) and a random positive number \(t_i\), and the matrix \((P_i^{-1})_i\) will be collaboratively computed based on SecMatInv. Here the \(P\) and \((P^{-1})_i\) are generated to transform the original matrix, and \(t\) is used to protect the equivalent eigenvalues. The \(Y\) can be computed with the above information as shown in line number 5. To simplify the calculation, here we only let \(S_1\) undertake the task of computing eigenvectors. Due to the high concurrency during the actual task, it is easy to balance the workload on two servers.

Since \(Y\) shares the same eigenvalues with \(t \times X\) and \(t\) is a positive number, the eigenvectors \(T\) under the \(s\) maximal eigenvalues of \(Y\) are related to those of \(X\). In this case, based on Lemma 1, the \(P \cdot T\) is the corresponding eigenvectors of

---

**Algorithm 4 Secure matrix inversion protocol SecMatInv**

**Input:** \(S_i\) has \(X_i\).

**Output:** \(S_i\) gets \(Y_i = (X^{-1})_i\).

**Offline Phase**:
1. \(T\) generates enough random matrices the sub-protocol uses and sends to \(S_i\).

**Online Phase**:
2. \(S_i\) generates a random square matrix \(Z_i\).
3. \(S_1\) and \(S_2\) collaboratively compute \(W_i = \text{SecMatMul}(Z_i, X_i)\).
4. \(S_1\) and \(S_2\) re-share and get \(W\), then \(S_i\) computes \(W^{-1}\).
5. \(S_i\) computes \(Y_i = W^{-1} \cdot Z_i\).

**Algorithm 5 Secure division protocol SecDiv**

**Input:** \(S_i\) has \(u_i\) and \(r_i\).

**Output:** \(S_i\) gets \((\frac{u_i}{g})_i\).

**Offline Phase**:
1. \(T\) generates enough random numbers the sub-protocol uses and sends to \(S_i\).

**Online Phase**:
2. \(S_i\) generates random number \(r_i\).
3. \(S_1\) and \(S_2\) collaboratively compute \(f_i = \text{SecMul}(u_i, r_i)\) and \(g_i = \text{SecMul}(u_i, r_i)\).
4. \(S_1\) and \(S_2\) re-share and get \(g_i\), then \(S_i\) computes \(\frac{u_i}{g_i}\).
Algorithm 6 Secure PCA protocol

Input: $S_i$ has $X_i^{n \times d} (n >> d)$.
Output: $S_i$ gets $P_i^{n \times s}$.

Offline Phase:
1: $T$ generates enough random numbers and matrices the sub-protocols use and sends to $S_i$.

Online Phase:
2: $S_i$ computes the mean value for each column of $X_i$ and composes $d$-dim vector $\bar{m}_i$, then computes $\bar{X}_i = X_i - \bar{m}_i$.
3: $S_i$ generates a random square matrix $P_i^{d \times d}$ and a random number $t_i$.
4: $S_1$ & $S_2$ collaboratively compute $P_i^{1 \times 1} = \text{SecMatInv}(P_i)$.
5: $S_1$ & $S_2$ collaboratively compute $Y_i = \text{SecMul}(t_i, \text{SecMatMul} (P_i^{1 \times 1}, \text{SecMatMul} (\text{SecMatMul} (X_i^T, X_i), P_i)))$.
6: $S_2$ sends $Y_2$ to $S_1$, then $S_1$ computes the eigenvectors of $Y_i$, then pick the $T^{d \times s}$ composed by eigenvectors under the $s$ maximal eigenvalues and sends them to $S_2$.
7: $S_i$ computes $V_i^{d \times s} = P_i \cdot T$.
8: $S_1$ & $S_2$ collaboratively compute $U_i^{d \times s}$ which is composed by $\text{SecMul}(v_i^{jk}, v_i^{jk})$ for all $v_i^{jk}$ in $V_i$, here $j \in [1, d], k \in [1, s]$.
9: $S_i$ computes the sum value for each column in $U_i^{d \times s}$, and compose a $s$-dim vector $\vec{r}_i$, then re-share it.
10: $S_i$ gets $\vec{r}$ and computes $v_i^{jk} = \sqrt{r_k}$ for all $v_i^{jk}$ in $V_i$, here $r_k$ means the $k$-th element in $\vec{r}$.
11: $S_1$ & $S_2$ collaboratively compute $F_i^{n \times s} = \text{SecMatMul} (X_i, V_i)$.

Algorithm 7 Secure sorting SecSort

Input: $S_i$ has numbers $\{u_i^l\}, l \in [1, n], n$ is the amount of numbers.
Output: $S_i$ gets the sorted numbers $\{v_i^l\}$.

Offline Phase:
1: $T$ generates enough random numbers and matrices the sub-protocols use and sends to $S_i$.

Online Phase:
2: $S_i$ randomly generates a positive number $t_i$.
3: $S_1$ & $S_2$ collaboratively compute $f_i^l = \text{SecMul}(t_i, u_i^l)$ for each $l$ in range $[1, n]$.
4: $S_1$ & $S_2$ re-share the $\{f_i^l\}$, then sort $\{f_i^l\}$.
5: $S_i$ gets $\{v_i^l\}$ by permuting $\{u_i^l\}$ based on the size relation of numbers in $\{f_i^l\}$.

Finally, the server can get the share of compressed data based on the share of original data and compression matrix $V$. Each server will save the mean vector $\bar{m}_i$ and the compression matrix $V_i$ for the retrieval process.

6.3 Secure Index Building

To accelerate the retrieval process, the index building schemes are considered on encrypted feature vectors stored in two servers. The typical HKM and C2LSH are used as examples, and it will be easy to notice that other schemes can be combined with our scheme too.

The comparison of distance is indispensable during the index building or query process. However, if we use the SecCmp to compare plenty of distances and try to find the minimum one, it will lead to too high interaction rounds or too much parallel calculation. For example, to seek the minimum value in 10 numbers, it will lead to 55 times of comparison in parallel. Note that the size relation of distances between features is insensitive, we propose the algorithm[7] to execute secure sorting effectively.

In algorithm[7], each server generates a random positive number and gets $\{f_i^k\}$ by executing SecMul on the number and share of true distances, then re-share them. In this case, the true distances will be protected by a random multiplication, but the size relation will be kept. The server can get the right size relationship of the share based on $\{f_i^k\}$. For efficiency, we also use the SecSort protocol in
Algorithm 8 Secure k-means protocol

Input:  \( S_i \) has \( k \), one share of vectors \( \{x_l^i\} \) and the ID each vector corresponds, here \( l \in [1,n] \), \( n \) is the number of vectors.

Output:  \( S_i \) gets one share of cluster centers, and the information the vectors in each cluster.

Offline Phase:
1:  \( T \) generates enough random numbers and matrices the sub-protocols use and sends to \( S_i \).

Online Phase:
2:  \( S_1 \) randomly picks \( k \) vectors as the centroid \( \{C_L^i\} \) and sends the corresponding ID to \( S_2 \).
3:  \textbf{for} \( l = 1 : n \) \textbf{do}
4:      \( S_1 & S_2 \) collaboratively compute the squared Euclidean distances between \( x_l^i \) and centroid vectors based on SecMatMul.
5:      \( S_1 & S_2 \) collaboratively seek the nearest centroid vector of \( x_l^i \) based on SecSort.
6:      \( S_i \) puts the \( x_l^i \) in the category which nearest centroid represents.
7:  \textbf{end for}
8:  \( S_i \) computes the mean value of vectors in each category, which is actually the share of new centroid vectors \( C_L^i \).
9:  Repeat line 3 to line 8 until a certain rounds or \( \{C_L^i\} \) unchanged.

the pooling layer for each 2×2 block. After constructing the secure sorting protocol, the secure index building schemes are shown as follows.

6.3.1 secure index building of HKM
As described in subsection 3.3, there are mainly two steps in the secure index building of HKM:
(i) Index encrypted vectors by secure k-means. The key problem in this step is the way of executing secure k-means based on the share stored in two servers. It should be noted that the identities of images are owned by both servers, which means they can execute the calculation on the corresponding vectors independently and synchronously by a determined algorithm. At the same time, some hyperparameters, like the number of images or the value of \( k \) that HKM uses, are also known by both servers. In this case, since the operations on vectors during k-means are only addition and comparison, and those are easy to execute as shown in Algorithm 8.

In algorithm 8 \( S_1 \) first randomly choose initial centroids and share it to \( S_2 \) for synchronization. Then, each server collaboratively executes clustering. The squared Euclidean distances are computed and SecSort is utilized for seeking the nearest centroid. Since the vectors in each category are known by both servers, the new mean value (i.e., centroid vector) can be easily computed without interaction.

(ii) Repeat secure k-means partition. The situation is the same as that in plaintext, as ID is known by both servers, which means each server can judge and repeat partition independently and synchronously. After the index building, each server can generate the same tree structure which is also same as that in plaintext state, however, the non-leaf nodes (except root node) and leaf nodes are only the share of true values.

Since the k-means in each layer can be executed in parallel, the complexity of interaction rounds during HKM index building is \( O(\log_k N) \), and the complexity of communication sizes is \( O(Nd log_k N) \), where \( k \) is the hyper-parameter in HKM, \( N \) and \( d \) are the number and dimension of vectors in the database.

6.3.2 secure index building of C2LSH
As described in subsection 3.3, there are mainly two steps during the secure index building process:
(i) Generate secure hash functions. It should be noted that the generator of LSH function is generating random numbers that are subject to a specific distribution. In this case, the following lemma should be noted.

Lemma 3: If two independent random variables \( X, Y \) satisfy normal distribution \( N(a_1, b_1) \) and \( N(a_2, b_2) \), then \( X+Y \) satisfy normal distribution \( N(a_1 + a_2, b_1 + b_2) \).

As C2LSH needs the \( \vec{a} \) whose elements obey \( N(0,1) \), in this case, each server can generate the elements in \( \vec{a} \), which obey \( N(0,1/2) \) independently. \( w \) is a constant value known by both two servers. As \( b \) is a random number in \([0,w]\), therefore, it can be generated by any server. For simplicity, let us assume that this task is undertaken by \( S_1 \), which means \( b_2 = 0 \).

(ii) Mapping encrypted vectors into LSH tables. Similar to formula 2, the servers first compute the share of \( h'(\vec{o}) \) collaboratively as follows,

\[
    h'_{\vec{a},b}(\vec{o}) = \text{SecMatMul}(\vec{a}, \vec{o}) + b_i w x.
\]

Here \( \vec{a}, \vec{o} \) and \( b_i \) are the share of \( \vec{a}, \vec{o} \) and \( b \), \( \vec{o}, \vec{b}, w \) and \( x \) are the same definition as formula 2. After computation, the servers will re-share them and get the \( \{h'_{\vec{a},b_j}(\vec{o})\} \), where \( j \in [1,m] \) and \( k \in [1,n] \), \( m \) is the number of generated LSH functions, \( n \) is the number of vectors in the database. Then each server can further compute bid each share of vectors belong based on formula 1. Finally, each server can add all IDs into the corresponding bucket in LSH tables. After the index building, two servers will generate the same LSH tables that are also same as that in plaintext state, however, the vectors corresponding to the IDs are only the share of true values.

Since the computation and share of \( \{h'_{\vec{a},b_j}(\vec{o})\} \) can be executed in parallel, the interaction rounds during C2LSH index building is 2, and the size relation is about \( O(Nd) \), here \( N \) and \( d \) are the number and dimension of vectors in the database.

6.4 Secure Retrieval
This section introduces the actual retrieval process and that is the only online phase from the respective of the secure retrieval task. The authorized user splits the query image into two shares and sends it as the trapdoor to \( S_i \).

After verifying the authorization of the user, servers will extract the aggregation feature of the query as subsection 6.1. Then servers collaboratively compress the extracted feature \( \vec{q}_i \), with the \( \vec{m}_i \) and \( V_i \) stored during compression, by computing \( \vec{q}_i = \text{SecMatMul}(\vec{q}_i, \vec{m}_i, V_i) \). Finally, according
to the method of index building, servers collaboratively search the m nearest images of the query. The retrieval process of HKM and C2LSH is shown as follows.

6.4.1 secure retrieval based on HKM
As described in subsection 3.3, there are mainly two steps during the secure retrieval based on HKM:

(i) Seek the nearest encrypted vectors. The servers will collaboratively compare the distances between query and centroids in each layer to find the nearest vectors and maintain the candidate list. The above operation only involves multiplication and sorting, which means the servers can easily finish the above task based on SecMatMul and SecSort. The above process can be executed iteratively until servers find the leaf nodes.

(ii) Get enough encrypted candidate vectors. The server will add the corresponding encrypted vectors in the current subspace into candidates. Based on the candidate list and protocols, the servers can execute the above step until they find enough candidates independently and synchronously. Finally, the squared Euclidean distances will be computed during the secure retrieval based on HKM:

\[ \text{dist} = (\langle q, a \rangle - \langle v, a \rangle)^2 \]

As described in subsection 3.3, there are mainly two steps during the secure retrieval based on HKM:

6.4.2 secure retrieval based on C2LSH
As described in subsection 3.3, there are mainly two steps during the secure retrieval based on C2LSH:

(i) Map encrypted query into LSH tables. Similar to the calculation during the index building, the \( \{ h'_{a,b,i}(q_i) \} \) will be computed based on encrypted query feature \( \tilde{q}_i \) and pre-generated encrypted LSH functions, here \( j \) is the same definition as that in subsection 6.3.2. Then the \( \{ h'_{a,b,i}(q_i) \} \) values are shared and the bid of query in each LSH table can be computed. In this case, since ID is known by both servers, the collision information can be counted by each server independently and synchronously.

(ii) Get enough encrypted candidate vectors. Also, \( \{ h'_{a,b,i}(q_i) \} \) is shared by both servers, the virtual rehash in each LSH table can be completed by each server without interaction. Like HKM, after getting enough candidates, the SecMatMul and SecSort will be finally used to get m-nearest images. To avoid the extra interaction, only the second completion condition in [19] is used in our experiment.

Since the servers need to map the query and compute some true distances, the interaction rounds during C2LSH index building is 5, and the complexity of communication sizes is \( O(md) \), where \( d \) is the dimension of the query vector, \( m \) is the number of vectors in candidates.

7 SECURITY ANALYSIS
The security analysis of our scheme is under the typical universal composability framework [40]. The execution of our scheme mainly involves the interaction between two cloud servers, and the process of the interaction is defined as the real experiment. In the proposed scheme, two cloud servers are potential adversaries. To prove the security of the protocol, it suffices to show that the view of the corrupted party (i.e., \( S_1 \) or \( S_2 \)) is simulatable given its input and output.

In the ideal experiment, the simulator \( S \) is defined as the one that can simulate the view of the corrupted party by using functionality \( F \). In this paper, we define the \( F \) as follows: \( F \) owns the input information and generates the random numbers or matrices locally, then it completes the calculation as subprotocols, the corresponding view of \( S \) will be filled by the calculation results. In the following, we will prove that the view is indistinguishable to that in the real world, and the view will not expose the detailed input information. To simplify the proofs, the following Lemmas will be used.

**Lemma 4.** A protocol is perfectly simulatable if all its subprotocols are perfectly simulatable.

**Lemma 5.** The protocol SecMul and SecMatMul are secure in the honest-but-curious model.

The readers can refer [41], [6], and [38] for the proofs of Lemma 4 and 5.

**Theorem 1.** The protocol SecCmp is secure in the honest-but-curious model.

**Proof.** For \( S_1 \), the view \( v_1 \) during executing protocol will be \( (t_1, a_1, b_1, c_1, \alpha_2, \beta_2, f_1, f) \). Here \( t_1 \) is randomly chosen from the range \([0, +\infty)\), which is indistinguishable. Here \( (a_1, b_1, c_1, \alpha_2, \beta_2, f_1) \) is simulatable based on Lemma 4 and 5. For simplicity, we assume \( f \) is positive. Due to \( f \) is the result of \( (u_1 - v_1) + (u_2 - v_2) \times (t_1 + t_2) \) computed by \( F \). Due to \( t_1 \) and \( t_2 \) only have the limitation that they should be positive, which means that \( f \) is randomly chosen from the range \([0, +\infty)\). In this case, \( f \) is indistinguishable from the real world. However, due to the \( S \) has detailed information of \( t_1 \), the range of \( (u - v) \) (e.g., \([0, \frac{1}{\ell})\)) will be exposed, however, it is still an infinite field. The view for \( S_2 \) will get a similar conclusion. In this case, the simulator can get an indistinguishable view and the view will not expose the detailed information of \( u, v \) or \( u - v \).

**Theorem 2.** The protocol SecMatInv is secure in the honest-but-curious model.

**Proof.** For \( S_i \), except those brought by SecMul, the view \( v_i = (Z_i, W) \). Here \( Z_i \) is composed of random numbers that are indistinguishable. \( W \) is the result of \((Z_1 + Z_2) \cdot (X_1 + X_2)\) computed by \( F \). Due to the randomization of \( Z \), any non-singular matrix in the corresponding size is the potential result, which means \( W \) is indistinguishable.

**Theorem 3.** The protocol SecDiv is secure in the honest-but-curious model.

**Proof.** For \( S_i \), besides those brought by SecMul, the view \( v_i = (t_i, u_i) \). Here \( t_i \) is randomly chosen from \( \mathbb{R} \) which is indistinguishable. \( y \) is the result of \((v_1 + v_2) \cdot (k_1 + k_2)\) computed by \( F \). Due to the randomization of \( k \), any non-zero number is the potential result, which means \( y \) is indistinguishable.

**Theorem 4.** The protocol SecSort is secure in the honest-but-curious model.
Proof. For $S_l$, besides those brought by SecMul, the view \( v_i = (t, \{f^{l}\}) \), where \( l \in [1, n] \) and \( n \) is the amount of input numbers. Here \( t_i \) is randomly chosen from \( \mathbb{R} \) which is indistinguishable. \( f^l \) is the result of \( (u_1^l + u_2^l) \times (t_1 + t_2) \) computed by \( F \). Due to the randomization of \( t \), any non-zero number is the potential result \( f^l \), which means \( \{f^l\} \) is indistinguishable. However, since \( S \) knows \( t \) is invariant, the ratio of numbers in \( \{u^l\} \) will be exposed.

**Theorem 5.** The CNN feature extraction process is secure in the honest-but-curious model.

Proof. All views in \( S_l \) are brought by SecCmp, and SecSort, therefore, the CNN feature extraction process is secure based on Lemma 4.

**Theorem 6.** The secure PCA protocol is secure in the honest-but-curious model.

Proof. For \( S_l \), besides those brought by SecMul, SecMatMult, and SecMatInv, the view \( v_i = (P_1, t, Y, \vec{r}) \). Here \( P_1 \) and \( k_1 \) are uniform random numbers that are obviously indistinguishable. \( Y \) is the result of \( t \times P^{-1} \cdot X^T \). \( X \cdot P \) computed by \( F \). Since \( t \) and \( P \) are random, therefore, any matrix, which is similar to the input matrix, with value scaling is the potential result \( Y \), which means that \( Y \) is indistinguishable. However, due to the above limitation, a similar matrix of eigenvectors and the ratio between eigenvalues is leaked. \( \vec{r} \) is composed of the sum of compressed eigenvectors, which is fixed with the same input, in this case, we only need to prove that exposure of \( \vec{r} \) will not cause actual damage to input. It should be noted that \( r_i \) is the square sum of corresponding \( d \)-dim eigenvectors, it is impossible to infect the detail value in the vector from only the square sum of them. This is essentially the problem in which there are more unknown values than effective equations on them.

**Theorem 7.** The secure k-means, HKM index building and query process are secure in the honest-but-curious model.

Proof. All views in \( S_l \) during calculating the above process are brought by SecMul, SecMatMult, and SecSort, in this case, the above process is secure based on Lemma 4. However, due to the share of \( ID \) and the leakage in SecSort, it is inevitable to leak the similarity between vectors or images.

**Theorem 8.** The C2LSH index building and query process are secure in the honest-but-curious model.

Proof. For \( S_l \), during executing the index building process of C2LSH, besides those brought by SecMatMult, the view \( v_i = \{h^l_d, b, (\alpha)\} \). Here \( h^l_d, b, (\alpha) \) is the result of \( \vec{d} \cdot \vec{a} + b \cdot \vec{x}, j \) \( \vec{a} \) means the \( (\vec{a}, b, j, x) \) is the parameter in \( j \)-th LSH table, \( \vec{a} \) means the \( k \)-th feature vector stored in the database, other definitions here are same as formula (2).

As numbers in \( \vec{a} \) are randomly chosen from distribution \( N(0, 1) \), although not uniformly, each \( h^l_d, b, (\alpha) \) is randomly chosen from an infinite field that is indistinguishable. It should be noted that although plenty of LSH functions map the same \( \vec{a} \) into different values, which constructs plenty of effective equations. However, each equation on \( \vec{a} \) will introduce more unknown random numbers (e.g., \( \vec{d} \) ). In this case, the view will not expose any information on detailed values in \( \vec{a} \). The situation is similar during the query process. It should be noted that although the detailed value information will not be exposed during the index building process, it is inevitable to leak the similarity information between images.

8 Experiment results

The section evaluates the performance of the proposed scheme in terms of encryption effectiveness, retrieval accuracy and retrieval efficiency. We implement the proposed scheme on Windows 10 operating system. All the user side experiments (i.e., image owner, authorized user, and trusty third party) are executed in a machine with Intel Core i5-8250u CPU @ 1.6GHZ and 16 GB memory. The cloud side experiments are executed on two servers running Windows 10 with the LAN setting. Each server is equipped with Intel Core i7-9700 CPU @ 3GHZ and 16 GB memory. We use Corel-1k and Corel-10k image dataset as the experimental dataset. The images in Corel-1k size either 384×256 or 256×384. Corel-1k includes 10 categories of images and each category contains 100 similar images. The size of images in Corel-10k is either 187×126 or 126×187. Corel-10k includes 100 categories of images and each category contains 100 similar images.

As described in subsection 2.2 the schemes in the first category let the image owner undertake the feature extraction task, which is obviously inferior to our scheme. Therefore, we focus on the comparison of schemes in the second category. We first give the comparison of the methods which execute the privacy-preserving CNN inference.

8.1 Performance of Privacy-Preserving CNN Inference Protocol

Due to the advantage of our schemes benefit from the designed protocols, we first show the comparison results of the complexity of protocols which are shown in Table 2. Table 2 compares the sign of numbers based on the most significant bit, therefore, their protocol needs \( O(l) \) rounds interaction, here \( l \) means the bit-length of ciphertext, which is generally set as 32. [9] gives their scheme based on additive resharing and multiplicative resharing, however, the transformation between them always leads to three or more interaction rounds.

To show the advantage in the scene of privacy-preserving CNN better, the real-time comparison is given in Table 3. Same as previous works [34], the MNIST dataset which contains plenty of 28×28 gray-scale images is used. The structure of used CNN model comes from the previous work [42], which is quite similar to VGG16: 2-Conv and 2-FC with ReLU and MaxPool.

Since our scheme outsources a part of random number generation tasks to CS, the offline consumption also significantly decreases. As a baseline, the actual efficiency of generating random numbers or matrices is shown as follow: In one second, \( T \) can generate \( 1.7 \times 10^7 \) random share of Beaver’s triples; or \( 5.4 \times 10^5 \) random share of matrices \( (A_i, B_i, (A \cdot B)_i) \), where \( A_i \) sizes \( 1 \times 8 \) and \( B_i \) sizes \( 8 \times 1 \); or \( 6.4 \times 10^5 \) random share of matrices \( (A_i, B_i, (A \cdot B)_i) \), where \( A_i \) sizes \( 512 \times 8 \) and \( B_i \) sizes \( 8 \times 1 \). The baseline will be used in the subsection 8.3.1.

The above results show the advantage of our schemes compared to previous CNN inference protocols in time or...
TABLE 3: Inference consumption of simple CNN

| Model      | Runtime(s) | Message sizes(MB) |
|------------|------------|-------------------|
|            | Offline    | Online            | Offline | Online |
| MiniONN [42] | 4.72       | 72                | 3046    | 6226   |
| Gazelle [34]  | 9.34       | 3.56              | 940     | 296    |
| Huang [6]    | 0.09       | 0.21              | 1.52    | 0.99   |
| Ours        | 0.002      | 0.05              | 1.41    | 0.24   |

TABLE 2: Comparison of the protocol complexities

|                  | SecCmp Rounds | Comm(bits)       | SecDiv Rounds | Comm(bits)       | SecMatInv Rounds | Comm(bits)       |
|------------------|---------------|------------------|---------------|------------------|------------------|------------------|
| Huang [6]        | $l + 1$       | $10l - 2$        | -             | -                | -                | -                |
| Xiong [25]       | 3             | $2l + 2$         | 3             | 6l               | -                | -                |
| Ours             | 2             | 6l               | 2             | 10l              | 2                | 6n*$l$           |

storage consumption. Besides, our scheme does not need scaling to ensure that the protocol works correctly, accordingly, no loss in accuracy will be caused in theory. Therefore, to our knowledge, the most efficient and accurate results in PPCBIR schemes based on typical features are introduced in this work. In the following, we only compare our work to the schemes based on the statistic feature.

8.2 Consumption before retrieval

In this section, we focus on the process from the step that image owner encrypts his images to the step that the cloud servers finish the index building. Two typical schemes [5], [25] based on statistics are chosen as the comparative experiments, we also give the results in plaintext state as the baseline and comparison.

8.2.1 Image outsourcing

In our scheme, the image owner only needs to split the image into two different encrypted images and send them to two servers. The schemes based on statistics need permutations on the image, which leads to plenty of time consumption. The time consumption of outsourcing whole Corel-1k or Corel-10k is shown in Table 4.

TABLE 4: Time consumption of image outsourcing

|                  | BOEW [5] | Cheng [25] | Ours |
|------------------|----------|------------|------|
| Corel-1k         | 281.86s  | 79.07s     | 16.59s |
| Corel-10k        | 1118.33s | 407.75s    | 84.46s |

8.2.2 Feature extraction and aggregation

As described in subsection 6.1, the encrypted feature will be collaboratively extracted by two servers. Due to plenty of non-linear operations, the consumption exceeds the schemes based on statistics. However, on the one hand, the aggregation consumption is much lower in our scheme; on the other hand, the time consumption of extraction can be reduced if simpler CNN models (e.g., Inception-V4 [43]) are used. The total time consumption of the two datasets is shown in Table 5.

The communication size during feature extraction is given in Table 6. The tasks in this subsection are all executed in the offline phase, which means $T$ has sufficient time for the generation of random numbers or matrices. Therefore, in this subsection, only the communication consumption during executing the task is given.

8.2.3 Feature Compression

In this work, the dimension of the compressed feature is set as 8 for both Corel-1k and Corel-10k. Since the previous schemes in PPCBIR ignore the process of compression, we only report the experiment results of our scheme which are shown in Table 7. It can be seen that the compression in consumption is negligible compared to the above step, however, we will show that it will highly decrease the communication consumption in the following.

8.2.4 Index Building

In our experiments, the hyper-parameter $k$ in HKM is set as 4, and the minimum number limitation of vectors in candidates is set as the $3 \times m$ in both HKM and C2LSH, where $m$ is the number of returned images (e.g., 50). The other parameters of C2LSH are set as suggested in [19].

Table 8 gives the time consumption of two index building schemes described in subsection 6.3. Here the results on both uncompressed and compressed features are given, and it is easy to notice that the communication consumption gain significantly decreases. For example, the communication consumption during HKM building on the compressed feature is only about 1/38 of the uncompressed version in Corel-1k.

8.3 Retrieval Consumption and Precision

8.3.1 Retrieval Consumption

Before the authorized user gets the plaintext images he needs, there are mainly three steps as described in subsection 6.4: Trapdoor generation, similarity computation in CS, and decryption.

The time consumption comparison results on retrieval (return Top-50 similar images) from Corel-1k and Corel-10k are shown in Table 9 and Table 10. From the tables, it can be seen that although the feature extraction of our scheme needs more time compared to statistics based schemes. However, the advantage of encryption and decryption will make our total consumption smaller. The non-index (i.e., linear index) situation is also given for the comparison. It is easy to notice that the utilization of index greatly decreases communication costs, and makes all kinds of consumption not substantially increase as the size of the dataset increase.

8.3.2 Discussion on trusty third party

From the above experiments, it is easy to see that only two types of matrices needed to be pre-generated for retrieval. The size of the matrix is determined by the dimension of the uncompressed and compressed features, which can be seen
TABLE 5: Time consumption of feature extraction and aggregation

|               | Xia [25]       | Cheng [25]     | Ours           | Plaintext   |
|---------------|----------------|----------------|----------------|-------------|
|               | Corel-1k       | Corel-10k      | Corel-1k       | Corel-10k   |
| Feature extraction | 190.99s       | 1495.07s      | 109.79s       | 426.33s     |
| Feature aggregation | 620.81s       | 860.73s       | 428.49GB      | 5455.11s    |

TABLE 6: Communication size of feature extraction

|               | Corel-1k | Corel-10k |
|---------------|----------|-----------|
| Feature extraction | 428.49GB | 1012.45GB |

TABLE 7: Consumption of feature compression

|               | Runtime | Message size | Plaintext |
|---------------|---------|--------------|-----------|
| Corel-1k      | 1.719s  | 31.83MB      | 0.279s    |
| Corel-10k     | 3.592s  | 137.31MB     | 0.375s    |

Table: As hyper-parameters shared by all entities before retrieval. Therefore, $\mathcal{T}$ can easily generate enough random matrices before retrieval.

After giving the actual time consumption of retrieval, we briefly analyze the choice of trusty third party $\mathcal{T}$. It is fairly good that the government or other credible agencies which can be trusty to all participants are willing to undertake this role; however, it is often difficult to seek in the real world. In this case, a reasonable assumption is letting the owner of images play the role instead.

In detail, besides outsourcing images, the image owner also spends a small amount of computing resources to generate random numbers and matrices, and the operations during offline phase are based upon them. Before the query, the authorized user also costs some time for generation, and servers will utilize them during the feature extraction of the query and secure compression process.

When facing actual distance computation, it is inevitable that the features of both image owner and authorized user will be involved. As the feature comes from authorized user is always meaningless to image owner, it is more suitable for image owner to undertake the generation task for the process. Since the random data needed during distance computation is about only 1ms as shown in Table 9 and 10 which means about one thousand queries can be supported at the cost of only one second during the offline phase. To sum up, we believe that the owner of images is suitable and capable of playing the role of $\mathcal{T}$.

8.3.3 Retrieval accuracy

In our experiments, the "precision" of a query is defined as that in [14]: $P_m = m'/m$ where $m'$ is the number of real similar images in the $m$ retrieved images. We choose all the images in Corel-1k and Corel-10k, and the retrieval accuracy comparison in these two datasets is shown as Fig. 4(a).

From the first and second sub-figures, due to the better feature extractor, it can be seen that our scheme shows great advantage when compared with the schemes based on statistics. As the third and fourth sub-figures show, we may observe that high dimension vectors show better accuracy. However, when the number of images is small (e.g., Corel-1k), the compressed feature shows better accuracy and the utilization of index (e.g., HKM) can even improve the accuracy by excluding the wrong candidate vectors brought by the feature extractor. Furthermore, the accuracy of our scheme is same as the corresponding plaintext state, in other words, the lossless property is confirmed by both theory and experiments.

9 Conclusion

In this paper, to bridge the gap between CBIR and PPCBIR, we utilize typical additive secret sharing and propose a series of novel protocols to execute secure computation efficiently. We further simulate the inference process of VGG16, the typical compression and index building schemes for better accuracy and efficiency. In future work, we will consider the construction of better protocols and other applications based on additive secret sharing.

Acknowledgements

This work is supported in part by the Jiangsu Basic Research Programs-Natural Science Foundation under grant numbers BK20181407, in part by the National Natural Science Foundation of China under grant numbers U1936118, 61672294, in part by Six peak talent project of Jiangsu Province (R2016L13), Qinglan Project of Jiangsu Province, and 333 project of Jiangsu Province, in part by the National Natural Science Foundation of China under grant numbers U1836208, 61702276, 61772283, 61602253, and 61601236, in part by National Key R&D Program of China under grant...
TABLE 8: Consumption of Index Building

| | Corel-1k | Corel-10k |
|---|---|---|
| | Ours | Plaintext | Ours | Plaintext |
| | Runtime | Message sizes | Runtime | Message sizes |
| HKM 512-dim | 10.13s | 781.69MB | 1.05s | 275.63s | 25.74GB | 12.97s |
| 8-dim | 2.34s | 20.87MB | 0.72s | 569.19MB | 7.72s |
| 512-dim | 1.43s | 6.81MB | 0.52s | 15.99s | 65.61MB | 6.18s |
| 8-dim | 1.17s | 1.99MB | 0.39s | 12.98s | 25.88MB | 5.28s |

TABLE 9: Retrieval Consumption in Corel-1k (Top-50)

| | BOEW [5] | Cheng [25] | Ours | Plaintext |
|---|---|---|---|---|
| | Offline | Online | Runtime | Message sizes | Runtime |
| | Trapdoor generation | feature extraction | 0.28s | 0.182s |
| | feature aggregation | 0.01s | 0.08s |
| | feature compression | 0.14s | 0.051s |
| | HKM retrieval | - | 0.2ms |
| | C2LSH retrieval | - | 0.76ms |
| | Linear retrieval | - | 54.33s |
| Decryption | 14.09s | 3.92s |
| Total | 14.57s | 58.52s |

TABLE 10: Retrieval Consumption in Corel-10k (Top-50)

| | BOEW [5] | Cheng [25] | Ours | Plaintext |
|---|---|---|---|---|
| | Offline | Online | Runtime | Message sizes | Runtime |
| | Trapdoor generation | feature extraction | 0.172s | 0.149s |
| | feature aggregation | 0.01s | 0.075s |
| | feature compression | 0.14s | 0.051s |
| | HKM retrieval | - | 0.2ms |
| | C2LSH retrieval | - | 0.76ms |
| | Linear retrieval | - | 54.33s |
| Decryption | 5.99s | 2.03s |
| Total | 5.99s | 34.19s |

2018YFB1003205, in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund, in part by the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAET) fund, China. Zhihua Xia is supported by BK21+ program from the Ministry of Education of Korea.

REFERENCES

[1] L. Zheng, Y. Yang, and Q. Tian, “Sift meets cnn: A decade survey of instance retrieval,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 5, pp. 1224–1244, 2017.
[2] C. Fu, C. Wang, and D. Cai, “Satellite system graph: Towards the efficiency up-boundary of graph-based approximate nearest neighbor search,” arXiv preprint arXiv:1907.06146, 2019.
[3] X.-S. Wei, J.-H. Luo, J. Wu, and Z.-H. Zhou, “Selective convolutional descriptor aggregation for fine-grained image retrieval,” IEEE Transactions on Image Processing, vol. 26, no. 6, pp. 2868–2881, 2017.
[4] 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.
[5] Z. Xia, L. Jiang, D. Liu, L. Lu, and B. Jeon, “Boev: a content-based image retrieval scheme using bag-of-encrypted-words in cloud computing,” IEEE Transactions on Services Computing, 2019.
[6] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, “A lightweight privacy-preserving cnn feature extraction framework for mobile sensing,” IEEE Transactions on Dependable and Secure Computing, 2019.
[7] L. Liu, J. Su, X. Liu, R. Chen, K. Huang, R. H. Deng, and X. Wang, “Toward highly secure yet efficient kmn classification scheme on outsourced cloud data,” IEEE Internet of Things Journal, vol. 6, no. 6, pp. 9841–9852, 2019.
[8] J. Chen, L. Liu, R. Chen, and W. Peng, “Shosvd: Secure outsourcing of high-order singular value decomposition,” in Australasian Conference on Information Security and Privacy. Springer, 2020, pp. 309–329.
[9] L. Xiong, W. Zhou, Z. Xia, Q. Gu, and J. Weng, “Efficient privacy-preserving computation based on additive secret sharing,” arXiv preprintarXiv:2009.05356, 2020.
[10] A. K. Jain and A. Vailaya, “Image retrieval using color and shape,” Pattern recognition, vol. 29, no. 8, pp. 1233–1244, 1996.
[11] D. G. Lowe, “Object recognition from local scale-invariant features,” in Proceedings of the seventh IEEE international conference on computer vision, vol. 2. Ieee, 1999, pp. 1150–1157.
[12] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in European conference on computer vision. Springer, 2006, pp. 404–417.
[13] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in null. IEEE, 2003, p. 1470.
[14] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” International journal of computer vision, vol. 104, no. 2, pp. 154–171, 2013.
[15] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, “Neural codes for image retrieval,” in European conference on computer vision. Springer, 2014, pp. 584–599.
REFERENCES

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “ImageNet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.

[17] A. Babenko and V. Lempitsky, “Aggregating local deep features for image retrieval,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1269–1277.

[18] M. Muja and D. G. Lowe, “Scaleable nearest neighbor algorithms for high dimensional data,” IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 11, pp. 2227–2240, 2014.

[19] J. Gan, J. Feng, Q. Fang, and W. Ng, “Locality-sensitive hashing scheme based on dynamic collision counting,” in Proceedings of the 20th ACM SIGMOD International Conference on Management of Data, 2012, pp. 541–552.

[20] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 1, pp. 117–128, 2010.

[21] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), vol. 2. IEEE, 2006, pp. 2161–2168.

[22] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in Proceedings of the twentieth annual symposium on Computational geometry, 2004, pp. 253–262.

[23] 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.

[24] J. Yuan, S. Yu, and L. Guo, “Seisa: Secure and efficient encrypted image search with access control,” in 2015 IEEE conference on computer communications (INFOCOM). IEEE, 2015, pp. 2083–2091.

[25] H. Cheng, X. Zhang, J. Yu, and Y. Zhang, “Encrypted jpeg image retrieval using block-wise feature comparison,” Journal of Visual Communication and Image Representation, vol. 40, pp. 111–117, 2016.

[26] B. Ferreira, J. Rodrigues, J. Leitao, and H. Domingos, “Practical privacy-preserving content-based retrieval in cloud image repositories,” IEEE Transactions on Cloud Computing, 2017.

[27] H. Liang, X. Zhang, and H. Cheng, “Huffman-code based retrieval for encrypted jpeg images,” Journal of Visual Communication and Image Representation, vol. 61, pp. 149–156, 2019.

[28] 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.

[29] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 169–178.

[30] S. Wang, M. Nasser, M. Atallah, and Q. Malluhi, “Secure and private outsourcing of shape-based feature extraction,” in International conference on information and communications security. Springer, 2013, pp. 90–99.

[31] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, “Securing sift: Privacy-preserving outsourcing computation of feature extractions over encrypted image data,” IEEE Transactions on Image Processing, vol. 25, no. 7, pp. 3411–3425, 2016.

[32] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in International conference on the theory and applications of cryptographic techniques. Springer, 1999, pp. 223–238.

[33] F. Liu, Y. Wang, J. Wang, Y.-Z. Zhang, and J. Lin, “Intelligent and secure content-based image retrieval for mobile users,” IEEE Access, vol. 7, pp. 119209–119222, 2019.

[34] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, “{GAZELLE}: A low latency framework for secure neural network inference,” in 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018, pp. 1651–1669.

[35] A. C. Yao, “Protocols for secure computations,” in 23rd annual ACM symposium on foundations of computer science (sfcs 1982). IEEE, 1982, pp. 160–164.

[36] R. Gilad-Bachrach, N. Dowlon, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in International Conference on Machine Learning, 2016, pp. 201–210.

[37] D. Beaver, “Efficient multiparty protocols using circuit randomization,” in Annual International Cryptology Conference. Springer, 1991, pp. 420–432.

[38] S. Wagh, D. Gupta, and N. Chandran, “Securenn: Efficient and private neural network training,” IACR Cryptol. ePrint Arch., vol. 2018, p. 442, 2018.

[39] J. Bar-Ilan and D. Beaver, “Non-cryptographic fault-tolerant computing in constant number of rounds of interaction,” in Proceedings of the eighth annual ACM Symposium on Principles of distributed computing, 1989, pp. 201–209.

[40] R. Canetti, “Universally composable security: A new paradigm for cryptographic protocols,” in Proceedings 42nd IEEE Symposium on Foundations of Computer Science. IEEE, 2001, pp. 136–145.

[41] D. Bogdanov, S. Laur, and J. Willemson, “Sharemind: A framework for fast privacy-preserving computations,” in European Symposium on Research in Computer Security. Springer, 2008, pp. 192–206.

[42] J. Liu, M. Juuti, Y. Lu, and N. Asokan, “Oblivious neural network predictions via minioonn transforms,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 619–631.

[43] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” arXiv preprint arXiv:1602.07261, 2016.

[44] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun, “Performance evaluation in content-based image retrieval: overview and proposals,” Pattern recognition letters, vol. 22, no. 5, pp. 593–601, 2001.
Wenhao Zhou is currently pursuing his master degree in the School of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include secure multiparty computation and privacy-preserving computation.

Jian Weng received the B.S. and M.S. degrees in computer science and engineering from South China University of Technology, Guangzhou, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2008. He is currently a Professor and the Dean with the College of Information Science and Technology, Jinan University, Guangzhou, China. He has authored or coauthored more than 100 papers in cryptography and security conferences and journals, such as CRYPTO, EUROCRYPT, ASIACRYPT, TCC, PKC, TPAMI, TIFS, and TDSC. His research interests include public key cryptography, cloud security, and blockchain. He was the PC Co-Chairs or PC Member for more than 30 international conferences. He also serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.