IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 22, 2019, accepted August 2, 2019, date of publication August 14, 2019, date of current version September 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935222

Intelligent and Secure Content-Based Image Retrieval for Mobile Users

FEI LIU[1], YONG WANG[1], FAN-CHUAN WANG[1], YONG-ZHENG ZHANG[2], AND JIE LIN[1]
[1]School of Computer Science and Engineering, Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Corresponding author: Yong Wang (cla@uestc.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2018YFB0804702.

ABSTRACT With the tremendous growth of smart mobile devices, the Content-Based Image Retrieval (CBIR) becomes popular and has great market potentials. Secure image retrieval has attracted considerable interests recently due to users' security concerns. However, it still suffers from the challenges of relieving mobile devices of excessive computation burdens, such as data encryption, feature extraction, and image similarity scoring. In this paper, we propose and implement an IND-CPA secure CBIR framework that performs image retrieval on the cloud without the user's constant interaction. A pre-trained deep CNN model, i.e., VGG-16, is used to extract the deep features of an image. The information about the neural network is strictly concealed by utilizing the lattice-based homomorphic scheme. We implement a real number computation mechanism and a divide-and-conquer CNN evaluation protocol to enable our framework to securely and efficiently evaluate the deep CNN with a large number of inputs. We further propose a secure image similarity scoring protocol, which enables the cloud servers to compare two images without knowing any information about their deep features. The comprehensive experimental results show that our framework is efficient and accurate.

INDEX TERMS Content-based image retrieval, convolutional neural network (CNN), lattice-based homomorphic scheme.

I. INTRODUCTION

It might be a cost-effective way to provide efficient and intelligent Content-Based Image Retrieval (CBIR) services that smart mobile users outsource their images onto cloud servers. This is due to its tremendous advantages, such as on-demand self-service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage-based pricing, and transference of risks. Despite the fact that the cloud-based CBIR has tremendous business and technical advantages to handle large-scale image repositories, new challenges regarding image data security have also arisen. These commercially operated cloud services are still struggling to handle the issue of efficient image retrieval with user's security concerns. For example, automatic face and object recognition functionality in Facebook photo management system has brought users excellent experience in the year 2011. Unfortunately, the community worried about their personal privacy when they knew that anyone could easily stalk and track anyone else by utilizing Facebook and various image search engines such as Google and Bing. This functionality had to be removed from their system after the prolonged controversy of one year. Awkwardly, Facebook recovered this functionality once again due to the requirement of intelligent image searching despite significant disapproval of community. Similar example came from Google glasses.

The primary reason for such dilemma is that the community is afraid of being stalked and illegally searched by malicious hackers from anywhere, especially if the image retrieval is performed by the system automatically. However, the system can generate image searching results more intelligently with the help of the modern machine-learning-based features, such as the automatic face recognition service can retrieve a list of images being captured with a specific friend. By modifying only the access control mechanism from public to private does not guarantee that the uploaded image is totally safe on a cloud platform. Furthermore, disabling

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

the automatic object recognition or encrypting the sensitive contents of images are not proper solutions, because both of them decrease the usability and efficiency of image searching.

Although users prefer systems to offer both functionalities, i.e. intelligent and secure image retrieval, the challenging task is to outsource the image retrieval onto a cloud without letting the cloud know anything about the image contents during the processing phase. In recent years, Convolutional Neural Networks (CNN), as a class of complex and powerful machine learning models, have demonstrated better-than-human accuracy across a variety of image retrieval tasks such as Approximate Nearest Neighbor (ANN) search [1]. A pre-trained CNN can be used as a feature extractor for intelligent and secure image retrieval, which requires both secure addition and multiplication operations. Fully homomorphic encryption techniques such as Lattice-based schemes [2] can potentially handle such issue, but are not easily adoptable due to their large computational complexity. Another approach is Secure Multi-party Computation (SMC) that supports secure image similarity calculation. However, it requires both parties i.e., the cloud and the client to interact constantly, which is not always possible for the mobile user.

In this paper, we study the problem of intelligent and secure CBIR for mobile users. The main contributions are summarized as follows:

- We propose and implement a CBIR framework that shifts excessive computations onto the cloud servers, such as IND-CPA secure image re-encryption, deep feature extraction, and image similarity scoring. In this way, a mobile user only needs to encrypt his/her image with a lightweight encryption algorithm and upload the encryption onto the cloud. The latter performs ANN image retrieval without the user's anymore interaction. Our framework supports dynamic updating of image databases and indexes.
- We use a pre-trained deep CNN model, i.e., VGG-16, to extract the deep features of an image. The information about the neural network is strictly concealed by utilizing lattice-based homomorphic scheme. We implement a real number computation mechanism to achieve better accuracy than previous works such as Gazelle framework without loss of its efficiency.
- We propose and implement a divide-and-conquer CNN evaluation protocol to deal with the problem of noise growth in the homomorphic scheme. Compared with previous works such as Gazelle, the protocol makes it possible to homomorphically evaluate very deep CNN with a large number of inputs.
- We further propose a secure image similarity scoring protocol, which enables the cloud servers to compare two images without knowing any information about their deep features. We apply our framework into three public image datasets. The experimental results show that our framework is efficient and accurate.

The rest of the article is organized as follows. We review the related works in Section II. Section III explains the preliminaries. In Section IV and Section V, we described our framework and protocols in detail. The experimental evaluations are shown in Section VI. Finally, we draw some brief conclusions in Section VII.

II. RELATED WORK

Homomorphic Encryption (HE) allows anyone to compute an arbitrary or a specific function (e.g., addition) f on an encryption of x, without decrypting it [3]–[5]. Homomorphic encryptions allowing only one type of operations (addition or multiplication) are called partially homomorphic encryptions such as Paillier cryptosystem [1] for additive homomorphic encryption (AHE), and ElGamal cryptosystem [6] for multiplicative homomorphic encryption.

Fully homomorphic encryption (FHE) performs both addition and multiplication, which was first introduced by Craig Gentry [4]. This scheme is based on ideal lattices and its construction contains two steps. It starts with a somewhat homomorphic encryption (SWHE) scheme which performs a fixed number of additive and multiplicative operations in the encrypted domain. A bootstrapping operation is added to the SWHE which results in fully homomorphic encryption, i.e., the number of operations is unlimited. Unfortunately, this scheme is inefficient for practical applications due to its high computation and memory cost.

In recent years, numerous lattice-based FHE schemes have been introduced in order to make it practical, such as Brakerski and Vaikuntanathans schemes [7], [8] and their famous optimization BrakerskiGentry-Vaikuntanathan (BGV) [9]. Their corresponding efficient implementations include Seal [10], TFHE [11], and HElib [12]. They are dramatically more efficient than conventional Paillier AHE. These libs enable us to design an efficient solution for secure CBIR on deep neural networks. The BGV encryption scheme consists in hiding the plaintext message with noise in order to create the ciphertext message. The decryption consists in removing the noise from the ciphertext message. The noise level increases with each homomorphic operation. If the noise level exceeds a certain threshold, it is no longer possible to correctly decrypt the message. The noise growth is much more important with multiplication as opposed to addition. This limits their applicability because deep neural networks need more operations.

In HE-based CBIR schemes, users encrypt images pixel by pixel by utilizing a homomorphic cryptosystem (e.g., Paillier [13], ElGamal [6], or Lattice-based AHE [11]), which allows the cloud to index and process their images in the encrypted domain. Hsu *et al.* [14] proposed a high-precision CBIR algorithm by adopting Paillier cryptosystem to encrypt images. This approach is suffered from significative ciphertext expansion, which leads to slow encryption and decryption and scalability issues. Hu *et al.* [15] further proposed an efficient scheme for SIFT feature extraction by utilizing the ring-Learn-With-Error (r-LWE) homomorphic cryptosystem [8]. Different from their previous scheme proposed in [14], their batched secure multiplication protocol is built on Some-What

Homomorphic Encryption (SWHE) scheme that enables the two parties to securely compute the products of multiple pairs of integers simultaneously, with computation and communication costs greatly reduced. Zheng and Huang [16] replaced Paillier ciphertexts with pointers to a ciphertext table. It reduced the number of encryption operations and minimized ciphertext expansion. Li *et al.* [17] proposed a double-decryption SIFT feature extraction scheme based on the BCP cryptosystem, which is an additively homomorphic scheme with two independent decryption algorithms. Although HE-based schemes allow the cloud server to process and index their encrypted images, which is semantically secure. Unfortunately, they present much higher time and space complexity [18]. More importantly, these schemes naturally are facing with ciphertext expansion and noise growth problems [7], [11], [19]–[22]. These have potentially negative effects on the scalability and accuracy. For example, schemes in [15]–[17] can only deal with the integer values of SIFT vectors and accept limited additive homomorphic operations. It is hardly applicable when considering CBIR with deep features, such as features extracted by convolutional neural networks (CNNs), because these schemes perform very poorly due to the large multiplicative depth in a CNN.

Others aim to improve image search efficiency and reduce storage requirement for massive image data retrieval. Qin Zou *et al.* [23] used Locality Sensitive Hashing (LSH) and constructed an index for SIFT feature vectors, which greatly reduced computational overhead on the client side because of avoiding the usage of the homomorphic encryption. Xia *et al.* [24] proposed a two-stage CBIR scheme that achieved constant search time by utilizing LSH. The scheme supports SIFT feature CBIR with the earth mover's distance (MED) as similarity metric.

We discuss the closely related works in detail here. Juvekar *et al.* [25] designed Gazelle, a secure neural network inference, using a combination of fully homomorphic encryption (FHE) and traditional two-party computation techniques. In their scheme, the user can acquire the classification results without revealing their input to the server, while guaranteeing the security of the server's neural network. Due to Gazelle is based on a two-party secure computation (2PC) scheme, the generated feature vectors are carefully distributed onto the two PCs. Specifically, the user keeps the noise-added feature vector, and the server keeps the corresponding noise. Gazelle provides scalable and efficient homomorphic operations for secure evaluation of convolutional neural networks. However, it requires constant communication between the user and the server, which is hardly acceptable by mobile users. On the other hand, Gazelle uses 64-bit word (a single machine word) to represent an integer, which improves the efficiency of homomorphic operations. However, CNN models are parameterized by real numbers in reality. In addition, when the number of homomorphic operations becomes large, Gazelle consumes huge memory spaces. These disable Gazelle be applied into the homomorphic evaluation of deep CNN models such as VGG-16, which are often used in CBIR due to their high accuracy. To address these problems, we propose a CBIR scheme that supports efficient homomorphic evaluation of the deep CNN model and releases mobile users from heavy computation and communication burdens.

The scheme proposed in [26] also supports secure CBIR with deep convolutional neural networks. In their scheme, the deep features are extracted by a VGG-16 model, which are then transformed into compact hash codes by a deep auto-encoder. Although it has very high CBIR accuracy, the scheme assumes the communication channels are not secure but the server is trusted. Hence, the query is processed on the server side in plaintext, and the user's input (user's image) is revealed to the server. In our scheme, we consider the security issues on the server side, which assumes that the server follows Honest-But-Curious (HBC) threat model (see Section IV).

In this work, we do not consider the problem of secure data mining, intended as training a neural network over encrypted data, which can be addressed, e.g., with the approach of [20]. Instead, we assume that the neural network is trained with data in the clear and we focus on the evaluation part.

III. PRELIMINARIES

A. CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional Neural Network (CNN) [27] is a deep artificial neural network, which has been proven very effective in areas such as image classification and objects recognition. A CNN is usually composed of linear layers each of which can be a convolutional (*Conv*) layer, or a fully-connected (*FC*) layer, and non-linear layers each of which applies a non-linear function, a *activation* function, that acts on each element of the input, or a *pooling* function that reduces the output size. Typical non-linear functions can be one of several types: the most common in the convolutional setting are *MaxPool* function and *ReLU* function. A CNN has several layers of non-linearities, which allows extracting increasingly complex features of the input and can lead to a better ability to generalize.

The Visual Geometry Group network (VGG-16) [28] can serve as a high accurate feature extractor. Its architecture is shown in Figure.1. The input image to the VGG-16 network is of fixed size, i.e., $3 \times 224 \times 224$. It is passed through a stack of various convolutional layers of different receptive fields. The stride rate for convolutional layers and pooling layers remains the same throughout the VGG-16 network which is 3×3 with stride 1 in convolutional layer and 2×2 with stride 2 in pooling layer. The first two convolutional layers have 64 and 128 filters, respectively. The rest of the convolutional layers include 256, 512 and 512 filters, respectively. Border pixels are padded before each convolutional operation, which can preserve the features maps size same to the input. The VGG-16 is ended with three fully connected layers. The first two *FC* layers consist of 4096 neurons while the final *FC* layer compresses the features to 1000 dimensions.

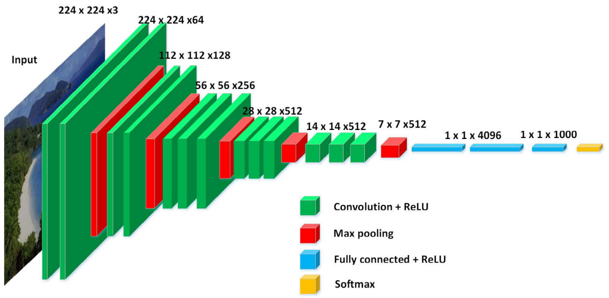


FIGURE 1. VGG-16 network architecture for feature extraction.

B. HOMOMORPHIC EVALUATION OF CNN

Gazelle [25] is a scalable and low latency system for secure evaluation of convolutional neural networks. It provides three basic homomorphic operations: Single Instrument Multiple Data (SIMD) addition (*SIMDAdd*), SIMD scalar multiplication (*SIMDScMult*), and permuting the plaintext slots (*Perm*). It also supports advanced homomorphic operations: homomorphic matrix-vector multiplications and homomorphic convolution. The homomorphic operations are based on packed additive homomorphic encryption (PAHE). The encryption scheme includes an encryption algorithm, a deterministic decryption algorithm, and a homomorphic evaluation algorithm. The encryption algorithm takes a plaintext message vector $\vec{u}$ from some message space and encrypts it using a private key s_k into a ciphertext denoted as $[\vec{u}]$. The decryption algorithm takes the ciphertext $[\vec{u}]$ and the key s_k and recovers the message vector $\vec{u}$. The homomorphic evaluation algorithm takes as input one or more ciphertexts that encrypt messages $\vec{u}_0, \vec{u}_1, \cdots$, and outputs another ciphertext that encrypts a message $\vec{u} = f(\vec{u}_0, \vec{u}_1, \cdots)$ for some function f constructed using the *SIMDAdd*, *SIMDScMult* and *Perm* operations. The PAHE constructions are parameterized by four constants that are the cyclotomic order m, the ciphertext modulus q, the plaintext modulus p, and the standard deviation σ of a symmetric discrete Gaussian noise distribution (χ). The framework satisfies IND-CPA security, which requires that ciphertexts of any two messages $\vec{u}$ and $\vec{u}'$ be computationally indistinguishable.

The Gazelle framework is based on the alternating use of PAHE and Yao's garbled circuits (GC), which can efficiently and securely convert between the data representations required for the two cryptographic primitives. As is shown in Figure 2, a linear layer is evaluated by utilizing homomorphic operations. The server A and the client B posses an additive share s_y and c_y respectively of the client's input y, that $y = s_y + c_y$. The client B first encrypts its share using the PAHE scheme and sends it to the server A. A in turn homomorphically adds its share s_y to obtain an encryption of the client input y, that $[c_y] + [s_y] = [y]$. The security of the homomorphic encryption scheme guarantees that B cannot recover y from this encryption. The server A then uses homomorphic matrix-vector multiplications and homomorphic convolution to evaluate linear layer (which is either *Conv* or *FC*). The result is a packed ciphertext that contains the input to a non-linear layer. The homomorphic scheme ensures that A learns nothing about B's input. B has not received any input from A yet and thus has no way of learning the model parameters.

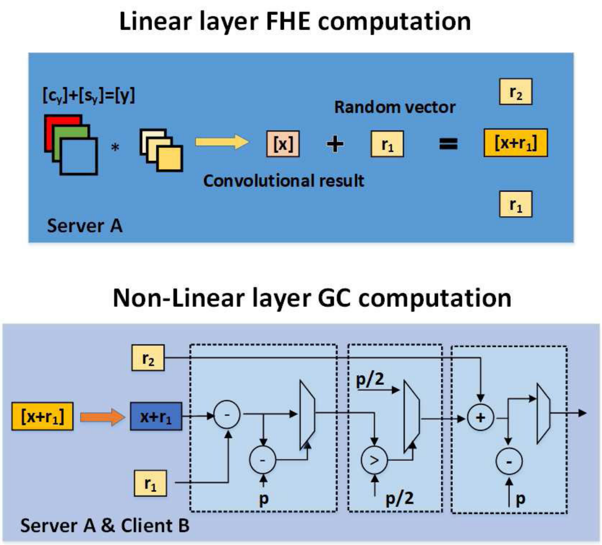


FIGURE 2. Gazelle: A low latency library for secure neural network inference.

For any non-linear layer, the server A keeps a ciphertext $[\vec{x}]$. It generates a random vector $\vec{r}_1$ and add it to ciphertext homomorphically, after that, A obtains an encryption $[\vec{x} + \vec{r}_1]$ and sends it to the client B. B recovers the plaintext $\vec{x} + \vec{r}_1$ with private key s_k. Yao's garbled circuits [29] is used for non-linear functions $f(\vec{x})$, which typically are *ReLU* and *MaxPool*. Taking the boolean circuits of *ReLU* as an example in Figure 2, the inputs are three vectors: random vectors $\vec{r}_1$ and $\vec{r}_2$ from server A, and $\vec{x} + \vec{r}_1$ from the client B. The output is a pair of shares $f(\vec{x}) + \vec{r}_2$ for the client B and $\vec{r}_2$ for the server A.

IV. THE SECURE CBIR FRAMEWORK

A. SYSTEM ARCHITECTURE

Our system architecture is illustrated in Figure 3. It consists of two entities, i.e, the cloud servers (S_1 and S_2), and the (multiple) users (*User*). The authorized users can encrypt and store their images onto the cloud servers, as well as issue a content-based image query with a query image.

- *The cloud servers* store encrypted images and perform content-based image retrieval by utilizing their huge storage capacity and computation power. We use two cloud servers, S_1 and S_2, to store the encrypted sub-images respectively. Specifically, S_2 holds the pre-trained CNN model (e.g., VGG-16 model) and takes on the homomorphic operations of linear layers. S_1 keeps its private key s_k and runs Yao's GC protocols with S_2 to carry out the secure computations of the non-linear layers. S_1 and S_2 hold a share of the deep features extracted from the encrypted images respectively. They answer the user's approximate nearest neighbor (ANN) query over the encrypted images together.

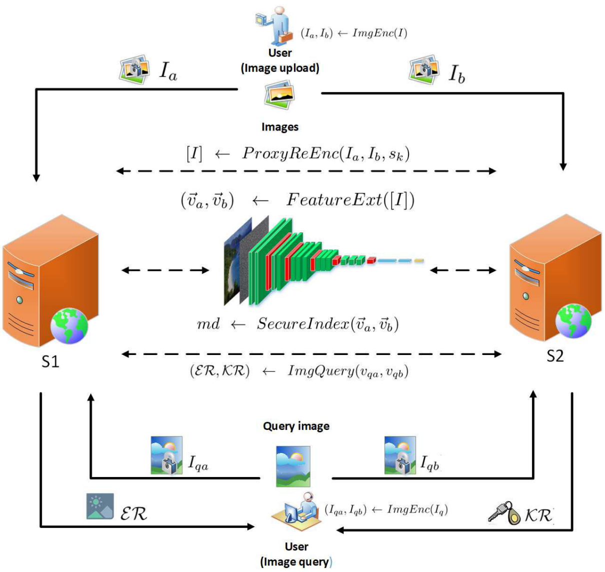

$(I_a, I_b) \leftarrow ImgEnc(I)$
User
(Image upload)
I_a
I_b
Images
$[I] \leftarrow ProxyReEnc(I_a, I_b, s_k)$
$(\vec{v}_a, \vec{v}_b) \leftarrow FeatureExt([I])$
$md \leftarrow SecureIndex(\vec{v}_a, \vec{v}_b)$
S1
S2
$(\mathcal{ER}, \mathcal{KR}) \leftarrow ImgQuery(v_{qa}, v_{qb})$
Query image
I_{qa}
I_{qb}
$\mathcal{ER}$
$(I_{qa}, I_{qb}) \leftarrow ImgEnc(I_q)$
$\mathcal{KR}$
User
(Image query)

FIGURE 3. The system architecture with the VGG-16 model.

- *The users* can both encrypt and upload their own images onto the cloud servers and/or search with a query image. For any image I, a user obtains the encrypted image I_a by adding a random matrix I_b onto it. Then, I_a is uploaded onto the cloud server S_1, and I_b is sent to S_2. When searching ANN images, a user generates the query trapdoor in the same way. After received I_a, S_1 carries out Proxy Re-Encryption (PRE) protocols [30] with S_2 to convert the encrypted image with key I_b to the encrypted image with key s_k.

The primary functionalities of our system are as follows.

- $(I_a, I_b) \leftarrow ImgEnc(I)$. Given an image I, it can be regarded as a matrix each element of which is represented with 8-bit. A random matrix I_b of the same shape is selected to encrypt I, the encrypted result $I_a = I + I_b$. Each element in I_b is randomly chosen from the integer interval $[0, 255]$. $I \leftarrow ImgDec(I_a, I_b)$ is the inverse operation of $ImgEnc(I)$. Note that the potential overflow can be ignored because I_b will be replaced by s_k when invoking Proxy Re-Encryption (PRE) protocols between S_1 and S_2.
- $[I] \leftarrow ProxyReEnc(I_a, I_b, s_k)$. Given the encrypted sub-images I_a holding by S_1 and I_b holding by S_2, S_1 encrypts I_a with its private key s_k, denoted as $[I_a]$, and sends it to S_2. The latter executes homomorphic subtraction to get the ciphertext of I, denoted as $[I] = [I_a] - I_b = [I_a - I_b]$.
- $(\vec{v}_a, \vec{v}_b) \leftarrow FeatureExt([I])$. Given the proxy re-encrypted ciphertext $[I]$ of the image I, it returns a pair of vectors $(\vec{v}_a, \vec{v}_b)$ that $\vec{v}_a$ is holding by S_1 and $\vec{v}_b$ is holding by S_2. For the deep feature vector of an image I that is extracted by a deep CNN model (e.g., VGG-16), $\vec{v}$, we have $\vec{v} = \vec{v}_a - \vec{v}_b$.
- $md \leftarrow SecureIndex(\vec{v}_a, \vec{v}_b)$. Given the deep feature vector pair $(\vec{v}_a, \vec{v}_b)$ that returned by invoking the function $FeatureExt([I])$, it generates the n-bit hash code that represents $\vec{v}$, $\vec{v} = \vec{v}_a - \vec{v}_b$. Similar images have similar hash codes, i.e., their Euclidean distances are close.
- $(\mathcal{ER}, \mathcal{KR}) \leftarrow ImgQuery(\vec{v}_{qa}, \vec{v}_{qb})$. Given the deep feature $\vec{v}_{qa}$ and $\vec{v}_{qb}$ of a query image I_q, it finds the ANN results according to the image similarity scoring function. Here, we use Euclidean distance between the deep features $\vec{v}_q$ of I_q and $\vec{v}$ of I. $\mathcal{ER}$ is the set of encrypted sub-images returned by S_1, and $\mathcal{KR}$ is the set of corresponding image decryption keys returned by S_2. The query user recovers the plaintext images by invoking the function $I \leftarrow ImgDec(I_a, I_b)$ that $I_a \in \mathcal{ER}$ and $I_b \in \mathcal{KR}$.

An authorized user encrypts image I by invoking the function $ImgEnc(I)$, and sends the encrypted sub-images (I_a, I_b) to the cloud server S_1 and S_2, respectively. The proxy re-encryption protocols are carried out by calling the function $ProxyReEnc(I_a, I_b, s_k)$ which converts (I_a, I_b) into the encrypted image $[I]$ by subtracting I_b homomorphically from $[I_a]$. S_1 and S_2 cooperatively extract the deep feature $\vec{v}$ of the image I by utilizing a deep CNN model. After which, S_2 gets $[\vec{v}]$ and sends $[\vec{v} + \vec{v}_b] = [\vec{v}] + \vec{v}_b$ to S_1, where $\vec{v}_b$ is a random sequence generated and kept by S_2. When received $[\vec{v} + \vec{v}_b]$, S_1 decrypts it with its private key s_k to get $\vec{v} + \vec{v}_b$ and keeps it as $\vec{v}_a$. An query user generates the query trapdoor (I_{qa}, I_{qb}) by invoking the function $ImgEnc(I_q)$. I_{qa} and I_{qb} are submitted to S_1 and S_2, respectively. S_1 and S_2 cooperatively finds the matched images and return the results ($\mathcal{ER}$ and $\mathcal{KR}$) to the query user respectively.

B. ADVERSARY MODEL AND DESIGN GOALS

Following the works done in [31], [32], we adopt the semi-honest adversary model for both the cloud servers, S_1 and S_2, i.e, they follow the protocol specifications and the algorithms exactly, but may attempt to learn additional information by analyzing intermediate computations. In general, secure protocols under the semi-honest model are more efficient than those under malicious adversary model, and most of the practical SMC protocols are secure under the semi-honest model. We refer the readers to [33] for more details about the security definitions and models. By the semi-honest model, we implicitly assume that the cloud servers do not collude. This model is realistic in the current cloud market. S_1 and S_2 could be cloud servers which are provided by legitimate, well-known companies (e.g., Amazon, Google, and Microsoft). Collusion between any of them is highly unlikely. We also assume that the users are honest, which could be easily guaranteed by access control and authorization mechanisms. At the same time, SSL, TLS, and other secure communication methods can be used to ensure channel security.

Similar to the work in [34], given the common private key s_k, consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game consists of the following steps.

- The adversary $\mathcal{A}$ chooses two plaintext m_0 and m_1, and sends them to the challenger $\mathcal{C}$.

- The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the protocol to obtain the ciphertext $C_b = P(s_k, m_b)$, and then sends C_b back to the adversary $\mathcal{A}$. $P(s_k, m_b)$ is used to denote the execution of the protocol with common public s_k and plaintext m_b.
- The adversary $\mathcal{A}$ can experiment with the code of C_b in an arbitrary non-black-box way, and finally outputs a bit $b' \in \{0, 1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be

$$Adv_{\mathcal{A}}(k) = |Pr(b' = b) - 1/2|$$

where k is the security parameter.

Then we give our design goals as follows.

- **Image data confidentiality.** The image data confidentiality is guaranteed if for any probabilistic polynomial time (PPT) adversary, the advantage $Adv_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary. That is, the cloud server S_1 (or S_2), as an adversary, knows nothing about the exact image data and its features except its shape.
- **Query security.** The query security is guaranteed if for any probabilistic polynomial time (PPT) adversary, the advantage $Adv_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary. In other words, the cloud server S_1 (or S_2), as an adversary, knows nothing about the query image I_q or the search results of I_q.
- **Model security.** The model security is guaranteed if for any probabilistic polynomial time (PPT) adversary, the advantage $Adv_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary. In other words, the cloud server S_1 (or S_2), as an adversary, knows nothing about the weights, the filter size, and the stride rate of the CNN model.
- **Efficiency**. The linear search is quite inefficient and computationally impracticable for a large database. The proposed scheme aims to achieve a better-than-linear search efficiency through constructing an efficient index.

In our framework, the encrypted sub-image I_a is purely random because I_b is random. If it has the size of 256×256 and each pixel has 8 bits, the adversary has to use brute-force approach to recover the image I which needs $256^{256\times256}$ operations. It is computationally infeasible in practice. We adopt proxy re-encryption to convert encrypted sub-images I_a and I_b to IND-CPA secure ciphertext $[I]$. The conjunctive use of the IND-CPA secure PAHE scheme and garbled circuits for evaluation of convolutional neural networks guarantees the image data confidentiality, Query security, and Model security.

V. THE IMPLEMENTATION OF OUR FRAMEWORK

We use the libs provided by Gazelle [25] to implement the secure deep feature extractor in $FeatureExt(\cdot)$ function. Gazelle satisfies IND-CPA security which guarantees our security goals described in Section IV. It has shown its very high efficiency, however, considering the operations of deep CNN models such as VGG-16, several technical improvements are necessary. These can help us implement a secure and accurate CBIR framework.

A. REAL NUMBERS COMPUTATION

Gazelle does not support real number computation because it uses 64-bit word (a single machine word) to represent an integer in order to achieve high efficiency. Unfortunately, the weights of the CNN model are real numbers in reality. Typically, we can solve this problem by encoding the digits after the decimal point as the highest degree coefficients of the polynomial [10].

More precisely, a real number $y = y^+.y^-$, where y^+ denotes the binary digits $b_{I^+}b_{I^+-1}...b_1b_0$, and y^- denotes the binary digits $b_{-1}b_{-2}...b_{-I^-}$, is encoded as the plaintext polynomial:

$$\sum_{i\leq I^+} X^i b_i - \sum_{1\leq i\leq I^-} X^{n-i} b_{-i} \tag{1}$$

This technique was adopted by several systems [20], [35]–[37]. However, Gazelle uses the PAHE scheme as an efficient and secure implementation, which supports packing multiple plaintexts into a single ciphertext and performing SIMD homomorphic additions. Since a real number is represented by a sequence of coefficients of a polynomial, when carrying out homomorphic operations, real numbers have to be encoded beforehand. The previous evaluations [25] showed that it dramatically slowed down the speed. For example, CryptoNets [20] cost around 297.5 seconds to carry out the operations in the CNN model which only has one *Conv* layer and two *FC* layers. Hence, we directly scale up real numbers by 10^n times and convert them into 64-bit integers, which represents real numbers with a fixed precision of n decimal points. Obviously, larger n will lead to better precision, but may cause result overflow when performing the homomorphic addition and multiplication. This forces us to use plaintext modulus p that is larger than 64 bits, which substantially slows down the homomorphic computation because it overflows a single machine word. Instead, we determine proper n so that the homomorphic addition (or multiplication) result of any two after-scaling integers does not overflow p.

We explain this by an example. Consider the first convolutional layer of the VGG-16 model, which accepts a $3 \times 224 \times 224$ size of input image. Its filter size is 3×3. Each value of the input image is normalized to the range of $[0, 1]$. If the maximum value in the filter is 0.5, the maximum output value of the layer is $3 \times 0.5 \times 9 = 13.5$. Hence, with the plaintext modulus p, the maximum scaling factor equals $\log_{10}(\frac{\frac{p}{2}-1}{13.5})$ because the range of the corresponding two's complement is $\left[-\frac{p}{2}, \frac{p}{2} - 1\right]$. When multiplying two numbers that have been scaled up by 10^n times, the result will be scaled up by 10^{2n} times. As results, the output of the first layer needs to be

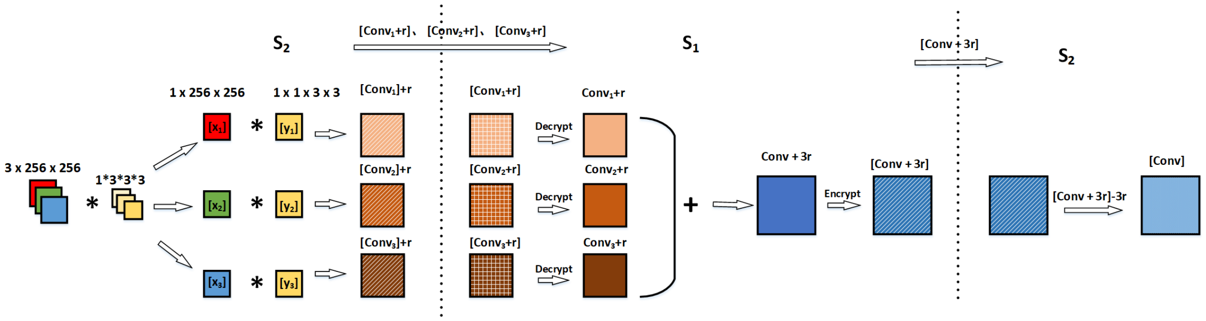


FIGURE 4. The divide and conquer convolutional layer computation.

divided by 10^n before it can be accepted by the subsequent layer. In this way, the scaling up factors can be applied to each layer of the VGG-16 model.

B. DIVIDE AND CONQUER LINEAR LAYERS COMPUTATIONS

Without loss of generality, a *Conv* layer can be represented by a tuple (w_i, h_i, c_i), where w_i is its input width, h_i is its input height, and c_i is the number of its input channels. A convolutional layer has c_o filter banks each consisting of c_i many $f_w \times f_h$ filters. This is represented by a tuple (f_w, f_h, c_i, c_o). The output of a *Conv* layer can be represented by a tuple (w_o, h_o, c_o) which is c_o many $w_o \times h_o$ output images. Similarly, The input of an *FC* layer is a vector $\vec{v}_i$ of length n_i and its output is a vector $\vec{v}_o$ of length n_o. An *FC* layer is specified by a tuple $(\boldsymbol{W}, \vec{b})$ where $\boldsymbol{W}$ is an $n_o \times n_i$ weight matrix and $\vec{b}$ is an n_o element bias vector. The output of an *FC* layer is $\vec{v}_o = \boldsymbol{W} \cdot \vec{v}_i + \vec{b}$.

With these parameters, the number of homomorphic multiplications and additions in a *Conv* layer is given by $(c_o \cdot c_i \cdot f_w \cdot f_h)$, and those in an *FC* layer is $n_i \cdot n_o$. This makes both the *Conv* and *FC* layers homomorphic operations quadratic in the input size. Unfortunately, when the number of the input and output channels, i.e., c_i, c_o, becomes large, the number of homomorphic operations will be dramatically increased. When performing SIMD homomorphic additions, the filters are packed into one vector and have to be loaded into memory. However, memory consumption increases with $c_i \cdot f_w \cdot f_h$. For example, when $c_i = 256$, the RAM space needed by VGG-16 model is more than 284GB. Disk RAM technology can be used to support large memory consumptions, but it slows down the *Conv* and *FC* layers computations and is expensive. More importantly, the noise level introduced by lattice-based FHE schemes will also arise, which may cause the decryption failure after homomorphic operations. The noise level rising can be hindered by avoiding one ciphertext involved in more than once homomorphic operations. However, it will still fail when the input channels becomes large (e.g.,$c_i = c_o = 256$ in the VGG-16 model). Another way to tolerate the noise is to select larger modulus p, but this will dramatically slow down the computation when p is larger than 2^{64} (a single machine word).

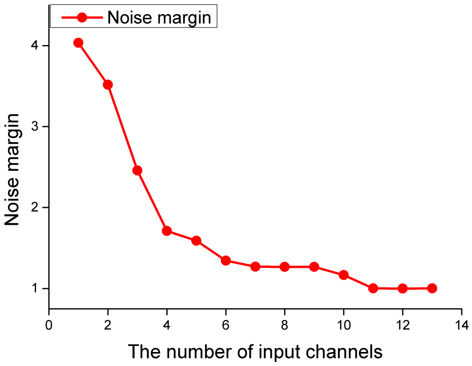
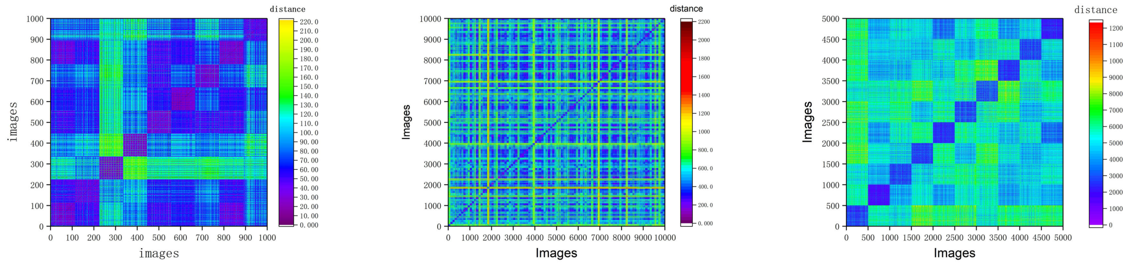


FIGURE 5. Noise margin vs. the number of input channels.

Considering our framework in Figure 3 again, the server S_1 keeps the private key s_k and decrypts the encrypted intermediate results before performing non-linear layer computations based on Yao's garbled circuits. Obviously, if the decryption succeeds, the noise is cleared. Hence, we adopt divide and conquer linear layers computation. The idea is to divide the input channels into groups to perform convolutional computations separately on S_2. After adding a uniform random vector $\vec{r}$ to each intermediate ciphertext homomorphically, S_2 sends them to S_1, the latter performs the decryption. After the additions in plaintext, S_1 encrypts the results with s_k and sends it back to S_2. S_2 gets the convolutional results by subtracting $\vec{r}$ homomorphically. In this way, *Conv* and *FC* layers can accept a large number of input channels (e.g., 512 input channels in VGG-16 model), and correctly perform homomorphic convolutional computations (homomorphic additions and multiplications) with low memory consumption (see Section VI).

Taking a *Conv* layer as an example, which is shown in Figure 4. The size of the encrypted input map is $3 \times 256 \times 256$, the size of the encrypted filter is $1 \times 3 \times 3 \times 3$. S_2 divides the encrypted input map into three groups, i.e., $[\vec{x}_1]$, $[\vec{x}_2]$ and $[\vec{x}_3]$. The corresponding filters are $[\vec{y}_1]$, $[\vec{y}_2]$, $[\vec{y}_3]$. S_2 performs homomorphic convolutional computations for each group to get $[\overrightarrow{Conv}_i] = [\vec{x}_i] * [\vec{y}_i]$, $i = 1, 2, 3$. Then, S_2 homomorphically adds uniform random vector $\vec{r}$ to $[\overrightarrow{Conv}_i]$, $i = 1, 2, 3$, that is $[\overrightarrow{Conv}_i + \vec{r}]$, $i = 1, 2, 3$. S_2 sends them to S_1. The latter does decryptions with private key s_k to get the plaintexts,



(a) Corel-1k (b) Corel-10k (c) MNIST

FIGURE 6. **The deep feature similarity of the three datasets.**

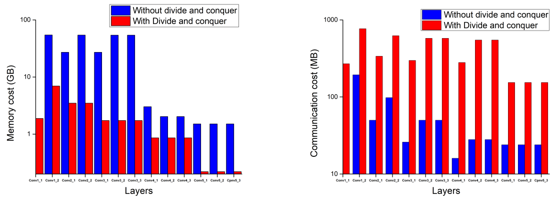
which are $\overrightarrow{Conv}_1 + \vec{r}$, $\overrightarrow{Conv}_2 + \vec{r}$, and $\overrightarrow{Conv}_3 + \vec{r}$. The summation of the plaintexts is $\overrightarrow{Conv}_1 + \overrightarrow{Conv}_2 + \overrightarrow{Conv}_3 + 3 \cdot \vec{r}$, which is encrypted again by S_1 and sent back to S_2. S_2 removes the random vector $\vec{r}$ by the homomorphically subtracting $3 \cdot \vec{r}$ to get the correct and encrypted convolutional results, i.e., $[\overrightarrow{Conv}]$.

The remaining problem is to determine the maximum number of the input channels in a group so that the ciphertext can be successfully decrypted after homomorphic convolutional computations. Let us consider the noise in the ciphertext. In lattice-based FHE schemes, the noise is bounded by the coefficients of the sampled error polynomials, the plaintext size, and the number of additions (or multiplications). We refer readers to [38] for detailed analysis. In our framework, the size of a plaintext is a single machine word (64 bits). The coefficients of the sampled error polynomials are pre-defined constants. The number of homomorphic operations are determined by filter size, $f_w \times f_h$, and the number of input channels, c_i. Given the ciphertexts $[\vec{x}]$ and $[\vec{y}]$, the noise growth of homomorphic addition, $[\vec{x} + \vec{y}]$, is at most $\eta_x + \eta_y$ where η_x (resp. η_y) is the amount of noise in the ciphertext $[x]$ (resp. $[y]$). Given the ciphertext $[\vec{x}]$ and the plaintext $\vec{y}$, the noise growth of homomorphic multiplication, $[\vec{x} \circ \vec{y}]$, is at most $p \cdot \sqrt{n} \cdot \eta_x$ where $\circ$ denotes component-wise multiplication of vectors, n is the number of slots in a packed AHE ciphertext. The convolutional computation results can be correctly decrypted if $\eta < q/(2p)$ where η is the overall at most noise growth. We have $\eta = c_i \cdot p \cdot \sqrt{n} \cdot \eta_x$. Hence, the maximum number of input channels, c_i, should satisfy $c_i < q/(2 \cdot p^2 \cdot \sqrt{n} \cdot \eta_x)$.

Consider the VGG-16 model again, we evaluate the noise margin with the increase of the number of input channels as is shown in Figure 5. The noise margin is defined as $\log_2(\frac{q}{p}) - \log_2(\eta)$. When the number of input channels is 11, the noise margin after the convolutional computations is 1.0016, which indicates that the decryption will fail. Thus, S_2 can at most accept 10 input channels. The first *Conv* layer of the VGG-16 model has 64 input channels, which are then divided into $\lceil 64/10 \rceil = 7$ groups in our framework.

C. IMAGE INDEXING

Given an image I, we add an *FC* layer and a *SoftMax* layer after the secure deep feature extractor in *FeatureExt*$(\cdot)$. The *FC* layer is to compress the deep features generated by the CNN model. The *SoftMax* layer is used as the final layer of a neural network-based classifier, which provides 'probabilities' for each class, the correct class could always have a higher probability and the incorrect classes always a lower probability. Thus, we get a pair of the deep feature vector $(\vec{v}_a, \vec{v}_b)$ that $\vec{v}_a$ is holding by S_1 and $\vec{v}_b$ is holding by S_2. We have $\vec{v} = \vec{v}_a - \vec{v}_b$, where $\vec{v}$ is the deep feature vector of the image I. The dimension with the maximum probability value of $\vec{v}$ represents the clustering center of the image I.



(a) Memory cost (b) Communication cost

FIGURE 7. **The performance of divide and conquer convolutional computation.**

Now, the problem comes to allow S_1 and S_2 to find the dimension with the maximum value under the condition that neither of them knows $\vec{v}$. The idea is to let S_1 and S_2 independently compute the partial values and exchange the intermediate results to compare and find the dimension with the maximum value correctly.

Given two elements $x, y \in \vec{v}$, $x - y$ can be transformed as:

$$\begin{aligned} compare(x, y) &= x - y \\ &= x_a - y_a - (x_b - y_b) \\ &= (x_a - x_b) - (y_a - y_b) \end{aligned} \tag{2}$$

where $x_a, y_a \in \vec{v}_a$ kept by S_1, $x_b, y_b \in \vec{v}_b$ kept by S_2. We have $x = x_a - x_b$ and $y = y_a - y_b$. Thus, S_1 and S_2 can calculate $x_a - y_a$ and $x_b - y_b$ independently. Then, S_2 sends the intermediate results to S_1. The latter can get $x - y$. Consequently, S_1 can find the dimension with the maximum value in the vector $\vec{v}$. Each deep feature vector $\vec{v}$ is classified into a clustering center according to its dimension with the maximum value. Finally, we simply use a hash table (HT) as an index to store the information on the cloud server S_1 (resp. S_2), each entry

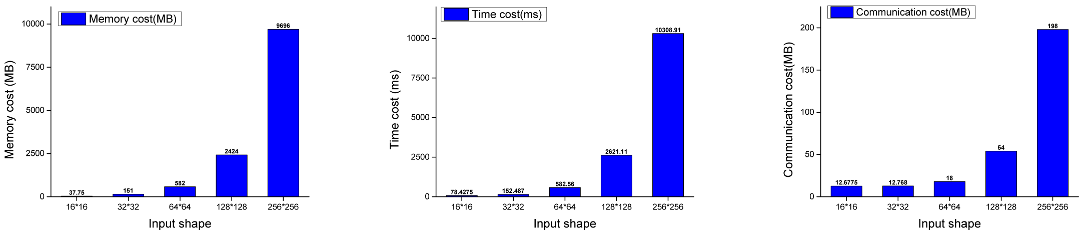


FIGURE 8. Performance of feature extraction v.s. input shapes.

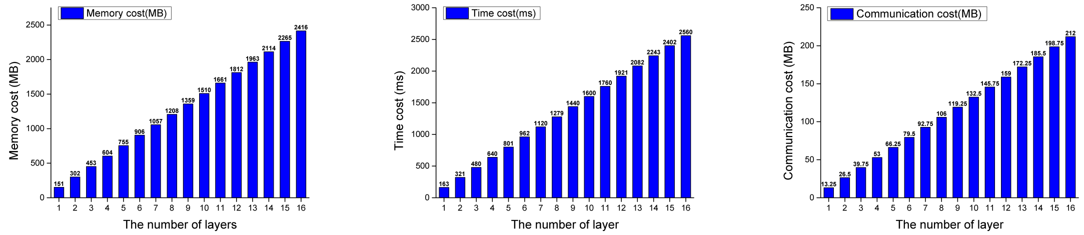


FIGURE 9. Performance of feature extraction v.s. the number of convolutional layers.

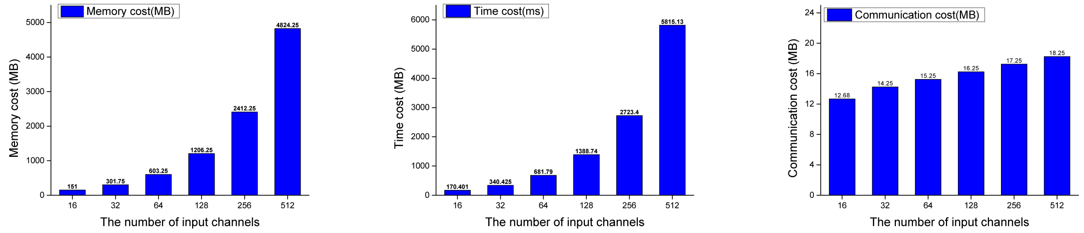


FIGURE 10. Performance of feature extraction v.s. the number of input channels.

of which includes the local path of the encrypted sub-images (e.g., I_a (resp. I_b)) and the corresponding partial deep feature vectors (e.g., $\vec{v}_a$ (resp. $\vec{v}_b$)).

D. IMAGE SIMILARITY SCORING AND IMAGE QUERY

Given the deep feature $\vec{v}_q$ of the image I_q, S_1 keeps its partial deep feature vector $\vec{v}_{qa}$, while S_2 holds the other partial deep feature vector $\vec{v}_{qb}$. We use Euclidean distance between $\vec{v}_q$ and $\vec{v}$ of a target image I as the image similarity scoring function, which is transformed as:

$$\begin{aligned} dist(\vec{v}, \vec{v}_q) &= ||\vec{v} - \vec{v}_q|| \\ &= ||(\vec{v}_a - \vec{v}_b) - (\vec{v}_{qa} - \vec{v}_{qb})|| \\ &= ||(\vec{v}_a - \vec{v}_{qa}) - (\vec{v}_b - \vec{v}_{qb})|| \end{aligned} \tag{3}$$

With the equation (3), S_1 and S_2 can compute the Euclidean distance between $\vec{v}_q$ and $\vec{v}$ without knowing $\vec{v}_q$ or $\vec{v}$.

With the image index HT, S_1 computes the Euclidean distance between $\vec{v}_q$ and any $\vec{v}$ in the same entry of HT according to the equation 3. Finally, S_1 sorts the results and returns $\mathcal{ER}$ to the query user, Similarly, S_2 returns $\mathcal{KR}$. The user can easily recover the result images by invoking the function $ImgDec(I_a, I_b)$ that $I_a \in \mathcal{ER}$ and $I_b \in \mathcal{KR}$.

VI. EXPERIMENTAL EVALUATIONS

A. EXPERIMENTS SETUP

Datasets To measure the feasibility of our method, we use three kinds of public image datasets. We randomly selected different numbers of images from the whole dataset as target images, and randomly selected one as the query image. Details of the three public datasets are as follows.

Corel-1k Dataset: All of the images in the Corel-1k dataset are in colored format. Image sizes are either 256 × 384 or 384 × 256. The images are grouped by content into 10 categories. Each category contains 100 images. These categories are African, Beach, Architecture, Buses, Dinosaurs, Elephants, Flowers, Horses, Mountains, and Food. There are 1000 deep feature vectors extracted in the whole dataset. In the experiments, we selected distinct collections of images, containing 100, 200, 300, ..., and 1,000 distinct images, respectively.

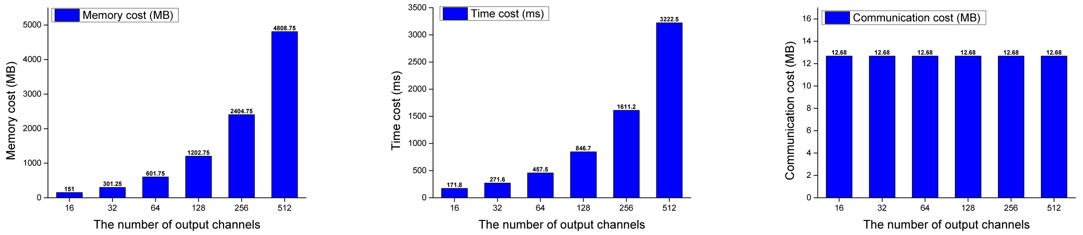
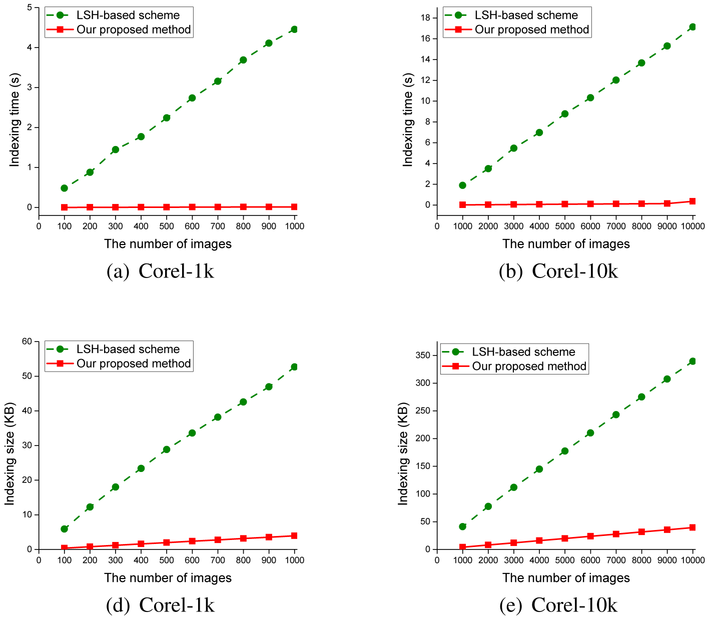


FIGURE 11. Performance of feature extraction v.s. the number of output channels.



(a) Corel-1k
(b) Corel-10k
(c) MNIST
(d) Corel-1k
(e) Corel-10k
(f) MNIST

FIGURE 12. Performance of index construction v.s. the number of images.

Corel-10k Dataset: The Corel-10K dataset contains 100 categories, and there are 10,000 images from diverse contents, such as animals, airplanes, furniture, ships, buildings, car, beach, food, national flag, etc. Each category contains 100 images of size 192×128 or 128×192. There are 10,000 deep feature vectors extracted in the whole dataset. In the experiments, we selected distinct collections of images, containing 1,000, 2,000, 3,000, ..., and 10,000 distinct images, respectively.

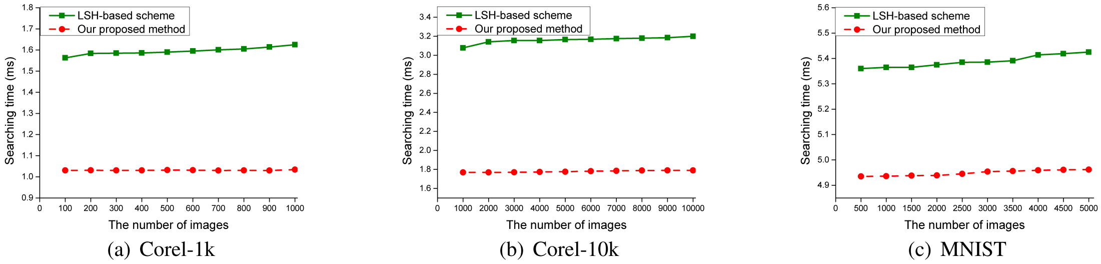
MNIST: The MNIST dataset of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. The training set is made up of numbers written by 250 different people and the value of each label is an integer between 0 and 9. Each image consists of 28 × 28 pixels, each pixel is represented by a gray value. We randomly selected 5,000 images in our experiments. They are grouped by content into 10 categories, each of which contains 500 images and is corresponding to an integer number of 0 to 9. In the experiments, we selected distinct collections of images, containing 500, 1,000, 1,500,..., and 5,000 distinct images, respectively.

TABLE 1. Details for the experimental datasets.

Dataset	# of images	Dataset size	Image shape
Corel-1k	1000	28.4 MB	384×256
Corel-10k	10000	134 MB	128×192
MNIST	5000	9.3 MB	28×28

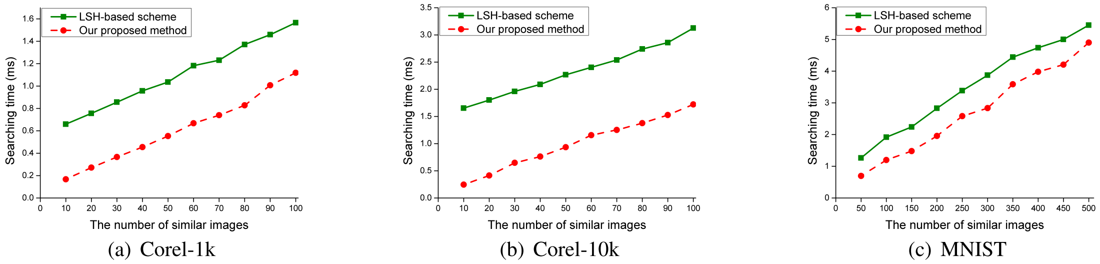
Table 1 shows the overview about the three public datasets.

Because the similarity of the images in the datasets has great impacts on the image retrieval results, we evaluated the deep feature vector similarity (Euclidean distance) between any two images in the datasets. As is shown in Figure 6, the colder the color, the closer the Euclidean distance between images, the higher the similarity between images. On the contrary, the warmer the color, the lower the similarity between images. It can be seen that the selected three image datasets are feasible for image retrieval. That is the images are not too similar or too different, which may cause skewed retrieval results. The similarity difference is almost the same between Corel-1k, Corel-10k, and MNIST datasets, there are around 10% images are considered similar.



(a) Corel-1k (b) Corel-10k (c) MNIST

FIGURE 13. Performance of image search v.s. the number of images.



(a) Corel-1k (b) Corel-10k (c) MNIST

FIGURE 14. Performance of image search v.s. the query result set size.

Evaluation Metrics: We evaluated the performance of the feature extraction, index construction, and image retrieval, respectively. During the feature extraction, we measured the consumed time (termed as *time cost*), memory space (termed as *memory cost*), and the communication overhead between the cloud servers S_1 and S_2 (termed as *communication cost*). During the index construction, we measured the consumed time (termed as *indexing time*) and memory space (termed as *indexing size*). As for the image retrieval, we measured the query processing time (termed as *searching time*). The *communication cost* refers to the size of the intermediate data in bytes exchanged between the two servers. We only measured the *time cost* consumed by each server, and the communication delay is ignored. In addition, we evaluated the correctness of the image retrieval by precision and recall curves.

Image Query Generation: We generate 10 query images that are randomly selected from the dataset. We average the evaluation metrics as the final results.

Implementation Details: The experiments were carried out on 2 PCs each with an Intel Core i7-5500U processor and 16GB RAM, running on Linux 64-bit OS. The source code can be found at https://github.com/pzimao/ppcbir.

B. PERFORMANCE OF FEATURE EXTRACTION

In feature extraction of the function *FeatureExt*$(\cdot)$, we evaluated the *memory cost* and *communication cost* of each VGG-16 linear layer, which includes the preprocess of filter, the preprocess of input filter map, and the homomorphic convolutional computation. We compared the performance with and without divide and conquer linear layer computation, as is shown in Figure 7. Obviously, the *memory cost* dropped dramatically from 54GB to 6.75GB (around 8 times) for the second convolutional layer. The *communication cost* between S_1 and S_2 increases around 4 times. These results show the effectiveness of divide and conquer linear layer computation.

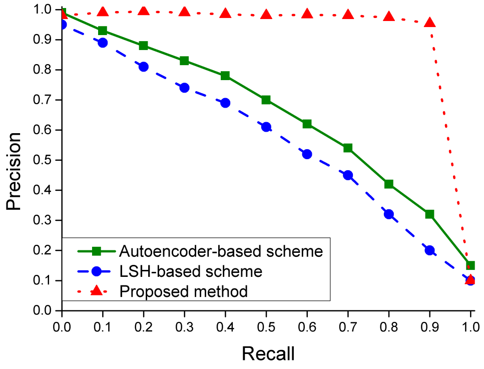


FIGURE 15. Performance of image search in comparison with other schemes.

We further evaluate the impacts of the input shapes (Figure 8), the number of convolutional layers (Figure 9), the number of input channels (Figure 10), and the number of output channels (Figure 11) that are posed on the function *FeatureExt*$(\cdot)$. It can be seen that *memory cost*, *time cost* and *Communication cost* increase linearly with respect to each parameter.

C. PERFORMANCE OF INDEX CONSTRUCTION

Figure 12 shows the experimental results conducted on three public datasets. The *indexing time* and *the indexing size* increase linearly with respect to the number of images. It is because the larger number of deep feature vector

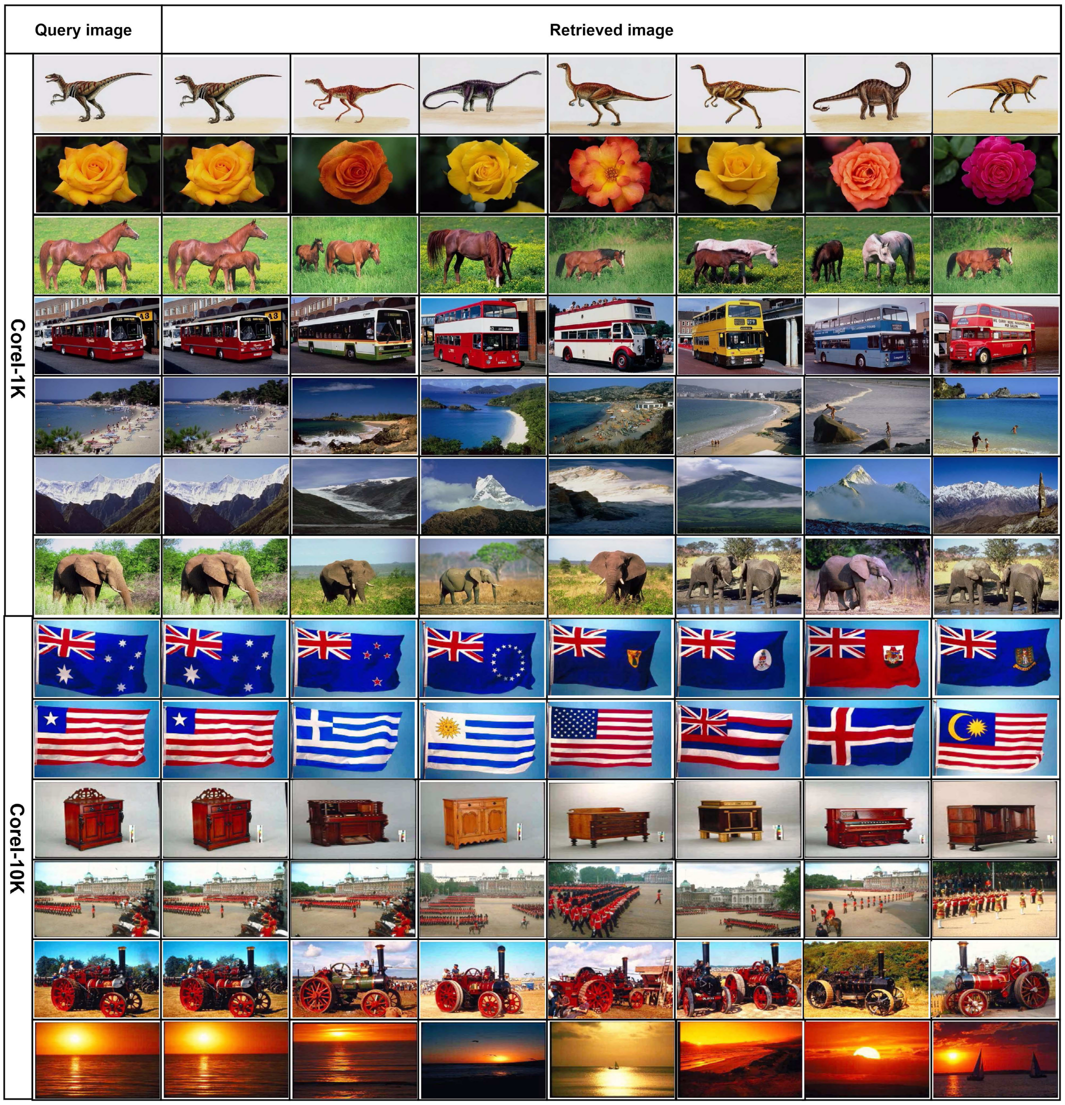


FIGURE 16. Samples of retrieved similar images.

contributes to the more calculations of the equation 2. We compared our proposed index with the LSH-based index [26]. We noticed that our proposed index performs much better than LSH-based index, since it has a simpler index structure than LSH-based index, which has shown its priority over other index. For the Corel-10k dataset (Figure 12(b)) which has the largest number of images, the indexing time of LSH-based index is within 18 seconds, the corresponding indexing size is around 349MB, in stark contrast to 0.35 seconds and 40MB of ours.

D. PERFORMANCE OF IMAGE SEARCH

Our *searching time* includes determining the position of the query feature vector in the index *HT* and calculation of image similarity scoring function. Figure 13 shows the impacts of the number of images on *searching time*. It shows that the *searching time* keeps stable. That is, it does not increase as the number of images increases. It can be seen that the overall trend of LSH-based schemes [26] is rising, because it retrieves dissimilar images.

Figure 14 shows the impacts of query result set size on *searching time*. Obviously, the *searching time* increases linearly with respect to the size of query result set. It is because that more similar images in the result set contribute to more image similarity scoring function computations.

E. THE PERFORMANCE COMPARISON

For feature extraction function $FeatureExt(\cdot)$, we compared *time cost* and *communication cost* with the works in [25], [36], [39] over MNIST dataset. As is shown in Table 2, the runtime and communication overhead of ours are much lower than those of MniONN [36] and Chameleon [39]. It has similar performance with Gazelle [25].

TABLE 2. The performance comparison on MNIST.

Works	Time cost (s)			Communication (MB)		
	Offline	Online	Total	Offline	Online	Total
Gazelle [25]	0.481	0.33	0.81	47.5	22.5	70.0
MniONN [36]	3.58	5.74	9.32	20.9	636.6	657.5
Chameleon [39]	1.34	1.36	2.7	7.8	5.1	12.9
Ours	1.76	0.701	2.461	47.5	22.5	70.0

We also compared with these works by running VGG-16 model over Corel-1k and Corel-10k datasets. Unfortunately, these works failed because of huge memory consumption. For example, Gazelle [25] failed because it requests around 54GB memory at one time and cannot recover the plain-text after the second convolutional layer. Chameleon [39] and MniONN [36] failed because they cannot successfully decrypt the outputs after the second convolutional layer.

F. IMAGE SEARCH CORRECTNESS EVALUATION

We evaluated our image search correctness by the widely adopted precision-recall curves, which are defined as:

$$\text{Recall} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of relevant images in dataset}} \tag{4}$$

$$\text{Precision} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of retrieved images}} \tag{5}$$

Given a recall value, a high accuracy means that a better retrieval performance. We carried out the experiments on Corel-10k dataset and compared the results with the deep auto-encoder scheme in [26] and the LSH-based scheme in [40], both of which have shown their better accuracy than other CBIR schemes. The results are shown in Figure 15. The *FC* layer after the VGG-16 model has $1 \times 1 \times 100$ neurons for Corel-10k dataset. The *SoftMax* layer consists of 100 maps where each map refers to one particular class of the Corel-10k dataset.

Figure 16 shows the image retrieval samples on Corel-1k and Corel-10k datasets, where the most relevant images have been successfully retrieved at top ranks. In the figure, the first column is the query image. The remaining columns are the retrieved images that are sorted according to their image similarity scoring function.

VII. CONCLUSION

In this paper, we study the problem of intelligent and secure CBIR for mobile users. We proposed a secure CBIR framework that uses VGG-16 as an accurate deep feature extractor. We implemented a real number computation mechanism and a divide-and-conquer CNN evaluation protocol to enable our framework to securely and efficiently evaluate deep CNN with a large number of inputs. We further proposed a secure image similarity scoring protocol, which enables the cloud servers to compare two images without knowing any information about their deep features. The experimental evaluation results indicate the efficiency and accuracy of our framework under various parameter settings. Unfortunately, the proposed solution in this article cannot take advantage of multi-core processing units, like GPUs. In future work, we plan to further improve the image retrieval efficiency by introducing more compact deep features in the current framework. We are also planning to build a framework that can run on GPUs.

REFERENCES

[1] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT*, J. Stern, Ed. Berlin, Germany: Springer, 1999, pp. 223–238.

[2] Y. Polyakov, K. Rohloff, and G. W. Ryan, "Palisade lattice cryptography library," Cybersecur. Res. Center, New Jersey Inst. Technol., Newark, NJ, USA, Tech. Rep., 2018. [Online]. Available: https://gitlab.com/palisade/palisade-release/tree/master

[3] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Information, Security and Cryptology—ICISC*, D. Lee and S. Hong, Eds. Berlin, Germany: Springer, 2010, pp. 229–244.

[4] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," in *Advances in Cryptology—EUROCRYPT*, D. Pointcheval and T. Johansson, Eds. Berlin, Germany: Springer, 2012, pp. 465–482.

[5] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Advances in Cryptology—CRYPTO*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Germany: Springer, 2012, pp. 868–886.

[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[7] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. IEEE 52nd Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2011, pp. 97–106.

[8] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Proc. 31st Annu. Conf. Adv. Cryptol. (CRYPTO)*. Berlin, Germany: Springer-Verlag, 2011, pp. 505–524.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 13:1–13:36, 2014.

[10] *Microsoft SEAL (Release 3.3)*, Microsoft Res., Redmond, WA, USA, Jun. 2019. [Online]. Available: https://github.com/Microsoft/SEAL

[11] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Advances in Cryptology—ASIACRYPT*, J. H. Cheon and T. Takagi, Eds. Berlin, Germany: Springer, 2016, pp. 3–33.

[12] S. Halevi and V. Shoup, "Faster homomorphic linear transformations in HElib," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 10991. Santa Barbara, CA, USA: Springer, 2018, pp. 93–120.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT*. Prague, Czech Republic: Springer-Verlag, 1999, pp. 223–238.

[14] 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.

[15] 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 Trans. Image Process.*, vol. 25, no. 7, pp. 3411–3425, Jul. 2016.

[16] P. Zheng and J. Huang, "An efficient image homomorphic encryption scheme with small ciphertext expansion," in *Proc. 21st ACM Int. Conf. Multimedia (MM)*, 2013, pp. 803–812.

[17] P. Li, T. Li, Z.-A. Yao, C.-M. Tang, and J. Li, "Privacy-preserving outsourcing of image feature extraction in cloud computing," *Soft Comput.*, vol. 21, no. 15, pp. 4349–4359, Aug. 2017.

[18] B. Ferreira, J. Rodrigues, J. Leitao, and H. Domingos, "Privacy-preserving content-based image retrieval in the cloud," in *Proc. 34th IEEE Symp. Reliable Distrib. Syst.*, Sep. 2015, pp. 11–20.

[19] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, Feb. 1989.

[20] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2016-3, Feb. 2016.

[21] I. Chillotti, N. Gama, and M. Georgieva, M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, to be published.

[22] Q. Mao, L. Wang, and I. W. Tsang, "A unified probabilistic framework for robust manifold learning and embedding," *Mach. Learn.*, vol. 106, no. 5, pp. 627–650, May 2017.

[23] Q. Zou, J. Wang, J. Ye, J. Shen, and X. Chen, "Efficient and secure encrypted image search in mobile cloud computing," *Soft Comput.*, vol. 21, no. 11, pp. 2959–2969, Jun. 2017.

[24] Z. Xia, Y. Zhu, X. Sun, Z. Qin, and K. Ren, "Towards privacy-preserving content-based image retrieval in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 276–286, Oct. 2018.

[25] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Secur. Symp. (USENIX Secur. 18)*, 2018, pp. 1651–1669.

[26] N. Rahim, J. Ahmad, K. Muhammad, A. K. Sangaiah, and S. W. Baik, "Privacy-preserving image retrieval for mobile devices with deep features on the cloud," *Comput. Commun.*, vol. 127, pp. 75–85, Sep. 2018.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst. 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[28] 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

[29] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, Oct. 1986, pp. 162–167.

[30] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology—EUROCRYPT*, K. Nyberg, Ed. Berlin, Germany: Springer, 1998, pp. 127–144.

[31] X. Yi, E. Bertino, F.-Y. Rao, and A. Bouguettaya, "Practical privacy-preserving user profile matching in social networks," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 373–384.

[32] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 633–644.

[33] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. 2. New York, NY, USA: Cambridge Univ. Press, 2004.

[34] M. Sepehri, S. Cimato, and E. Damiani, "Privacy-preserving query processing by multi-party computation," *Comput. J.*, vol. 58, no. 10, pp. 2195–2212, Oct. 2014.

[35] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 619–631.

[36] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[37] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *Proc. 55th Annu. Design Autom. Conf. (DAC)*, 2018, pp. 2:1–2:6.

[38] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan, "Functional encryption for inner product predicates from learning with errors," in *Advances in Cryptology—ASIACRYPT*, D. H. Lee and X. Wang, Eds. Berlin, Germany: Springer, 2011, pp. 21–40.

[39] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur. (ASIACCS)*, 2018, pp. 707–721.

[40] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 128–131, Mar. 2008.

FEI LIU received the B.S. degree in computer science and technology from the University of Electronic Science and Technology of China, Chengdu, China, in 2018. His research interests include spatial database, spatial query processing, and privacy enhancing technologies.

YONG WANG received the Ph.D. degree in computer science and technology from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2008. He is currently an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include spatial database, spatial query processing, and privacy enhancing technologies.

FAN-CHUAN WANG received the B.S. degree in computer science and technology from the University of Electronic Science and Technology of China, Chengdu, China, in 2019. His research interests include spatial database, spatial query processing, and privacy enhancing technologies.

YONG-ZHENG ZHANG received the B.S. and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2001 and 2006, respectively. He is currently a Professor and a Ph.D. Supervisor with the Institute of Information Engineering, Chinese Academy of Sciences (CAS), Beijing, China. His research interest includes network security, particularly, cyberspace security situational awareness.

JIE LIN received the B.S. and Ph.D. degrees in computer science and technology from the University of Electronic Science and Technology of China, Chengdu, China, in 2006 and 2009, respectively, where he is currently an Associate Professor with the School of Computer Science and Engineering. His research interests include image processing and data mining.