DCIRM: Dynamic and Controllable Image Retrieval Scheme in Multi-Owner Multi-User Settings

Chenyang Mao, Zhonghua Shen, Kefei Chen, Yong Liu, Qian Meng, and Fuqun Wang

***Abstract*—The proliferation of cloud computing technology has led to a significant number of users opting to store image data on the cloud. To ensure the confidentiality of image data, it is recommended to apply encryption techniques prior to uploading them to cloud servers. However, the traditional encrypted image retrieval schemes prove inadequate for practical application scenarios due to limitations in supporting multi-owner and multi-user settings, inefficient access control, and a lack of dynamic verifiability. To address the challenges presented by practical application scenarios, in this paper, we propose a Dynamic and Controllable Image Retrieval scheme in the Multi-owner multi-user settings (DCIRM). First of all, we realize the usability in multi-owner and multi-user scenarios through re-encryption settings. Then, we realize the verifiability of dynamic data by applying the dynamic authentication constructed by the chameleon hash function. Finally, we realize lightweight access control through the polynomial-based access strategy. The DCIRM scheme has been demonstrated to be privacy-preserving, efficient, and feasible through rigorous security analysis and experimentation with authentic datasets.**

***Index Terms*—Dynamic verifiable, encrypted image retrieval, multi-owner multi-user settings, polynomial-based access strategy.**

I. Introduction

With the increasing popularity of digital cameras, smartphones, and other devices, massive amounts of image data have entered our lives. However, the computing and storage burdens of these devices also increase accordingly. In order to solve this problem, an increasing number of people are opting to store image data in the cloud. Content-Based Image Retrieval (CBIR) technology [1] and [2] has been widely used to retrieve the images uploaded to the cloud server accurately and efficiently, but the privacy protection of the images uploaded to the cloud server is also essential. Users will encrypt the image before uploading it to prevent the cloud server from revealing the image's privacy, but this will render CBIR technology unavailable. The previously mentioned requirements are conveniently complied with by Searchable Encryption (SE) technology [3]. Unlike traditional encryption schemes, which cannot guarantee the availability of encrypted images or cannot be used on devices with limited resources, image-based searchable encryption can protect image data privacy while allowing operations like image retrieval on cloud servers. Currently, there is plenty of research being done on encrypted image retrieval techniques, and the security of the techniques themselves is constantly being enhanced.

However, there are some non-negligible issues with the application of most existing solutions in daily-use scenarios. First, the existing schemes only take into account single-user scenarios and do not take into account multi-owner or multi-user scenarios [4], [5]. Image owners and image users would share the same key if these schemes were applied directly to scenarios involving multi-owner and multi-user.The owner or user would be able to decrypt encrypted indexes and trapdoors uploaded by all other owners or users. If each image owner is set to encrypt the index with their own key in order to preserve the privacy of distinct owners, the image users must likewise produce different trapdoors for matching calculations, which will result in significant computation and communication overhead. Therefore, it is necessary to design an encrypted image retrieval scheme that supports multi-owner and multi-user scenarios. Second, in the scenario of supporting multi-owners and multi-users, an encryption scheme with an access control strategy is required due to the rise in user numbers. Even though the majority of existing schemes accomplish the function of access control [6], there will be an enormous rise in communication and computation overhead.

Finally, the majority of current schemes operate under the assumption that the cloud server is both truthful and inquisitive. In actuality, the cloud server has the potential to provide inaccurate search outcomes as a means of conserving computational resources. Although most existing solutions achieve result verifiability using an accumulator or Merkel Hash Tree (MHT), it is frequent practice for the image owner to upload new images to the cloud server in addition. In this case, as depicted in Fig. 1, because changing the leaf nodes in the MHT causes the entire tree to change, it is necessary to rebuild the entire MHT every time the image changes, resulting in significant calculation and communication overhead [7]. Despite the fact that the accumulator can support dynamic result verification, updating it requires a lot of time [8].

Manuscript received 12 July 2023; revised 5 December 2023; accepted 17 January 2024. Date of publication 22 January 2024; date of current version 8 August 2024. This work was supported in part by the National Social Science Foundation of China under Grant BIA210166, in part by the National Natural Science Foundation of China under Grants 61972124 and 62302136, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LQ22F010001, in part by the Ningbo 2025 Major Project of Science and Technology Innovation under Grant 2021Z109, and in part by Starting Research Fund from the Hangzhou Normal University under Grants 4115C50221204137 and 4085C50220204093. *(Corresponding author: Yong Liu, Qian Meng.)*

Chenyang Mao and Yong Liu are with the School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China (e-mail: chenyangmao@stu.hznu.edu.cn; yongliu@hznu.edu.cn).

Zhonghua Shen, Kefei Chen, Qian Meng, and Fuqun Wang are with the School of Mathematics, Hangzhou Normal University, Hangzhou 311121, China (e-mail: szh@hznu.edu.cn; kfchen@hznu.edu.cn; mq@hznu.edu.cn; fqwang@hznu.edu.cn).

Digital Object Identifier 10.1109/TSC.2024.3356650

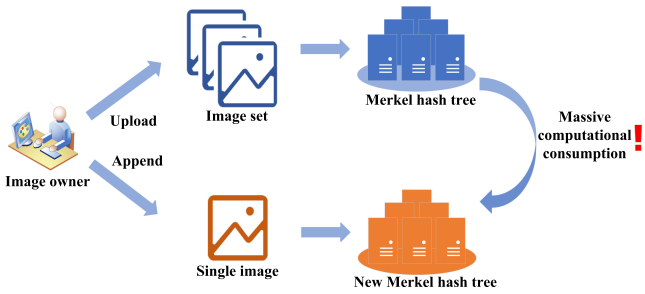

Fig. 1. Limitation of existing schemes.

With these motivations in mind, we propose a Dynamic and Controllable Image Retrieval Scheme in the Multi-owner Multi-user Settings (DCIRM). DCIRM scheme realizes the dynamic verifiability of the retrieval scheme by using the chameleon hash function [9] to construct a dynamic verification tree and realizes the setting of multi-owner and multi-user by re-encrypting the trapdoor and index. In the process of encryption, a polynomial-based access control strategy is used to realize lightweight access control, which further improves the security of the scheme. The main contributions of DCIRM are summarized as follows:

1) Support dynamic result verification: For the purpose of dynamic verifiability, the DCIRM scheme constructs a verification tree utilizing the chameleon hash function. The dynamic verification tree is also uploaded to the cloud server by image owners along with encrypted images and encrypted indexes. When the user appends the uploaded data, the content in the tree can be dynamically updated through the properties of the chameleon hash function, and the verification tree does not need to be regenerated. So that the retrieval scheme's dynamic properties can be realized, as well as the calculation and communication overhead are significantly reduced.

2) Support multi-owner and multi-user settings: DCIRM scheme applies re-encryption settings for image matching. Users receive their own encryption keys from KGC. The cloud server will create re-encrypted indexes and trapdoors after users upload encrypted indexes and encrypted trapdoors, respectively. With this approach, usability for scenarios involving multiple owners and users is realized, the process of exchanging keys is streamlined, and communication and computation costs between owners and users of images are decreased.

3) Realize lightweight access control: In the process of calculating the similarity between the trapdoor and index, the DCIRM scheme employs a polynomial-based access control strategy to achieve lightweight access control. When unauthorized users retrieve images, the similarity between the trapdoor and index will far exceed the threshold, causing the cloud server to not send images beyond the threshold to unauthorized users, thereby realizing access control.

4) Realize high efficiency and high accuracy: DCIRM scheme employs the convolutional neural network (CNN) to extract image features. As opposed to traditional methods of feature extraction, CNN can take the original image as its input directly without the need for a complex feature extraction algorithm, significantly lowering the computational burden of the system and enhancing its effectiveness. In addition, CNN can adapt the model to different image data and extract the most distinguishing features. This method of feature extraction has higher precision, better and more accurate image expression, and higher retrieval accuracy.

II. Background

A. Related Work

Lu et al. [10] introduced the first image encryption technique, according to order-preserving encryption and Min-hash. Later, Xia et al. [4] proposed a content-based image retrieval scheme that supports privacy protection. They used color layout descriptors and edge histogram descriptors to generate image features, but the accuracy of these two methods was low, resulting in low retrieval accuracy. To increase retrieval accuracy, Li et al. [11] employed the Scale Invariant Feature Transform (SIFT) to extract image features, which dramatically improved the accuracy of feature extraction. Recently, Li et al. [5] extracted image features using the CNN model. However, as the majority of these plans do not take into account the issue of multiple owners and users, they have significant limitations in real-world application scenarios.

Many schemes have proposed and implemented verifiable functions to stop malicious cloud servers from altering retrieval results and fabricating retrieval contents. Xu et al. [12] proposed a verification structure based on accumulator to verify the results efficiently. Liu et al. [8] used RSA accumulator to dynamically verify the correctness of retrieval results, however, this method generated a lot of computational overhead. In addition, Catalano et al. [13] proposed an efficient homomorphic MAC for arithmetic circuits. Unfortunately, this scheme only supports verification in plaintext-limited fields. Wan et al. [14] designed a verifiable privacy-preserving keyword search scheme, which combined homomorphic MAC with traditional ordered multi-keyword retrieval. But it only supports single-owner and single-user scenarios. Later, Rao et al. [15] and Li et al. [16] put forward a result integrity verification scheme based on the Merkel hash tree, which realized efficient result verification. Merkel hash tree ensures the verifiability of data by storing the hash value of data in the leaf nodes of the tree, but these schemes do not support dynamic verification, because the change of leaf node data will cause the change of the whole Merkel hash tree, thus generating huge computational overhead. In order to realize the dynamic verifiability of the scheme, xu et al. [17] designed a dynamic verification tree based on the Chameleon hash function, but it was not used in the application scenario of encrypted image retrieval.

At present, many schemes [18], [19], [20] have realized the fine-grained access control of encrypted cloud data based on Ciphertext-Policy ABE (CP-ABE) [6] technology. The encrypted image retrieval scheme proposed by Zhang et al. [21] supports fine-grained access control based on CP-ABE, but complex access policies and a large number of user attributes lead to high encryption and decryption overhead. Nair et al. [22] designed a fine-grained access control scheme based on the

bilinear accumulator, but it also did not support encrypted image retrieval.

B. Preliminary

We introduce the preliminary knowledge used in this article, including polynomial-based access control and the chameleon hash function.

1) Polynomial-Based Access Strategy: Polynomial-based Access strategy (PBA) [23] is a lightweight access control model. The image owner can set the identity of the user who is allowed to access the image according to the polynomial coefficient. Specifically, it is assumed that the user set is $\Phi = \phi_1, \phi_2, \ldots, \phi_n$, where n represents the number of identities. If only ϕ_k is allowed to access the image IC_i, the image owner will construct an access control polynomial $G_i(x)$ for the image IC_i according to (1), where M is a positive integer and n is is the number of roles.

$$G_i(x) = M \cdot \prod_{\phi_k \in \Phi_n} (x - \phi_k) = M \cdot \sum_{j=0}^{n} a_{ij} x^j \tag{1}$$

If the role ϕ_k of the image user is the root of $G_i(x)$, then $G_i(\phi_k) = 0$, the image user is allowed to query the current image. Otherwise $|G_i(\phi_k)| > M$, the image user is not allowed to query the current image.

2) Chameleon Hash Function: The chameleon hash function is a kind of anti-collision hash function, but different from the ordinary anti-collision hash function, it has a trapdoor, through which hash collision can be easily found.

Definition 1. The chameleon hash function is a tuple of three PPT algorithms CH = (chamGen, Ch, Col).

$chamGen(1^\lambda) \rightarrow (cpk, csk)$*:* Key generation algorithm, given the security parameter λ, outputs the public and private key pair (cpk, csk) of the chameleon hash function.

Ch(cpk, m, r) $\rightarrow$ *h:* Chameleon hash algorithm. Given the public key cpk, message m and random value r, and output chameleon hash value h.

Col(csk, m, r, m'*)* $\rightarrow r'$: Computational collision algorithm. Given the trapdoor csk, the message m, the random value r and the new message m' that needs to match the collision, a random value r' is output, which satisfies $Ch(cpk, m, r) = Ch(cpk, m', r')$

Definition 2. A chameleon hash function CH is invertible if it is surjective and there exists an efficient algorithm:

scol(csk, m, h) $\rightarrow r$*:* Given trapdoor csk, message m, and hash value h, a random value r is output, which satisfies $h = Ch(m, r)$.

3) Secure K-Nearest Neighbor Query: Secure k-nearest neighbor (Secure kNN) [24] query is a method that can calculate the euclidean distance between encrypted vectors. It mainly consists of the following parts:

$kNN.Genkey(1^\lambda) \rightarrow (k_s)$*:* Given the security parameter λ, outputs the key group k_s. k_s includes two $(d+1) \times (d+1)$-dimensional random reversible matrices M_1 and M_2, and a $(d+1)$ dimensional binary vector B.

$kNN.Genindex(f_i, k_s) \rightarrow (f_i^*)$*:* Given a plaintext vector f_i, first expand it to $f_{ie} = \{f_{i1}, \ldots, f_{id}, \sum_{j=1}^{d} f_{i,j}^2\}$. Then it is divided into f_{ie1} and f_{ie2} according to the vector B. Specifically, if $B[k] = 0$, then $f_{ie1}[k] = f_{ie2}[k] = f_{ie}[k]$; if $B[k] = 1$, then $f_{ie1}[k] + f_{ie2}[k] = f_{ie}[k]$. Get the encrypted vector $f_i^* = \{M_1^\top f_{ie1}, M_2^\top f_{ie2}\}$

$kNN.Gentrap(f_q, k_s) \rightarrow (f_q^*)$*:* Given a trapdoor vector f_q, first expand it to $f_{qe} = \{-2f_{q1}, \ldots, -2f_{qd}, 1\}^\top$. Then it is divided into f_{qe1} and f_{qe2} according to the vector B. Specifically, if $B[k] = 1$, then $f_{qe1}[k] = f_{qe2}[k] = f_{qe}[k]$; if $B[k] = 0$, then $f_{qe1}[k] + f_{qe2}[k] = f_{qe}[k]$. Get the encrypted trapdoor $f_q^* = \{M_1^{-1} f_{qe1}, M_2^{-1} f_{qe2}\}$.

$kNN.Match(f_i^*, f_q^*) \rightarrow (IP_{i,q})$: The inner product of f_i^* and f_q^* is $IP_{i,q} = f_{qe}^\top f_{ie} = \|f_q - f_i\|^2 - \|f_q\|^2$.

When comparing the euclidean distance between f_i, f_j and the trapdoor vector f_q, it can be obtained by calculating $IP_{i,q} - IP_{j,q} = \|f_q - f_i\|^2 - \|f_q - f_j\|^2$. If the result is greater than 0, it means that the distance between f_j and f_q is small. If the result is less than 0, the distance between f_i and f_q is small.

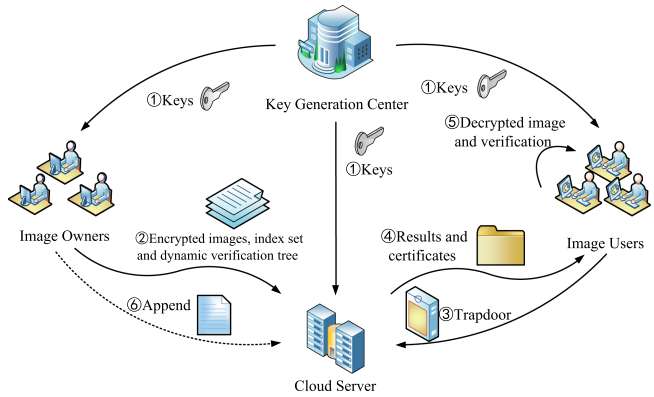

Fig. 2. System model of DCIRM scheme.

III. Problem Formulation

In this section, we mainly introduce the system model and threat model.

A. System Model

As depicted in Fig. 2, the DCIRM scheme consists of four entities: Key Generation Center, Image Owners, Image Users, and Cloud Server. The subsequent are the functionalities of these entities.

- *Key Generation Center (KGC):* KGC is primarily in charge of key creation and distribution, as well as generating the system parameters. During the system initialization phase, the KGC distributes the corresponding keys to the IOs, the IUs, and the CS through the secure channel.
- *Image Owners (IOs):* IOs first encrypt the original image and upload the encrypted image to CS. Furthermore, IOs extract features through CNN, generate an encrypted index set and a dynamic verification tree, and upload the encrypted index and dynamic verification tree to CS. After this, the IOs can update the encrypted image set and the dynamic verification tree.

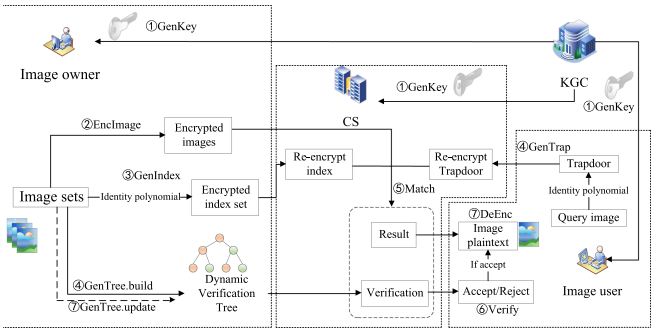

Fig. 3. Overview of DCIRM scheme.

- *Image Users (IUs):* Each IU generates a trapdoor according to its own query image and sends it to the CS for the query. Upon receipt of the output from CS, the IUs proceed to verify the image, subsequently decrypting it and obtaining the plaintext of the image.
- *Cloud Server (CS):* CS stores the encrypted image set and the dynamic verification tree transmitted by the IOs. As per the trapdoor provided by IU, the process involves re-encryption of the index and trapdoor, followed by similarity computation and extraction of query result proof information from the tree. Finally, the obtained query results and supporting information are sent to the IU.

B. Threat Model

We assume that IOs, IUs, and KGC are completely credible, but CS is malicious. While IOs, IUs, and KGC can execute commands correctly without leaking information, CS may tamper with or forge images uploaded by IOs, and even return wrong query results. Based on the information obtained in CS, we consider the following threat model:

- *Known Ciphertext Attack Model:* The cloud server attempts to obtain information related to the image content from the encrypted image, encrypted index, dynamic verification tree received from IO, the encrypted trapdoor received from IU, and query results.
- *Known Background Attack Model:* In this model, the cloud server will have more information based on the known ciphertext model. For example, the cloud server obtains IO's access records on other websites to try to infer the image type, and analyze the query content or purpose of the IU. Or infer plaintext and ciphertext pairs based on a large number of query requests and query results.

IV. Proposed Scheme

This section presents the main idea of the DCIRM scheme and provides a comprehensive description of its particulars.

A. Main Idea

As shown in Fig. 3, KGC will first assign a key to each user and send the re-encryption key corresponding to each user to the cloud server in step ①. IO encrypts the image in step ②, uses CNN to extract image features, and transmits the encrypted vector in step ③. Next, in order to accomplish dynamic verification, IO constructs a dynamic verification tree and transmits the verification tree to CS in step ④. Then, IU expands the vector to generate a trapdoor in the same way. When CS receives the trapdoor, it will re-encrypt the trapdoor and the encryption vector and compare the inner product value in step ⑤. Provided that the inner product value falls below the designated threshold, IU shall be granted permission to retrieve the image and subsequently return both the image and proof. Finally, IU will verify the image after receiving the encrypted image and certificate, and decrypt the image after verification in step ⑥ and step ⑦.

TABLE I
Comparative Summary

Schemes	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$
[5]	image	CNN			
[7]	image	Fisher			Polynomial-based
[8]	text	-	✓		
[4]	image	MPEG-7			
[11]	image	SIFT			
[14]	text	-			
[15]	text	-	✓		
[16]	image	CNN		✓	
[17]	text	-	✓		
[6]	text	-			CP-ABE
[6]	text	-			CP-ABE
[6]	text	-			CP-ABE
[21]	image	SIFT		✓	CP-ABE
[22]	text	-			Accumulator
DCIRM	image	CNN	✓	✓	Polynomial-based

Note. $\mathbb{F}_1$: Data type; $\mathbb{F}_2$: The way of feature extraction; $\mathbb{F}_3$: Support dynamic verification; $\mathbb{F}_4$: Support multi-owner multi-user; $\mathbb{F}_5$: Access control policy type;

TABLE II
Notations

Symbols	Descriptions
M_{p1}, M_{p1}^{-1}	p's random real matrix and its invertible matrix
SK_p	p's private key
RK_p	p's re-encryption key
IO_p,IU_t	image owner and image user
IP, IC	image plaintext set and image ciphertext set
Γ, tp	index set and trapdoor
IC_r, π_r	query results and corresponding proofs
DVT	dynamic verification tree
I_q	query image

B. Details of Scheme

First, we give some notations in Table II. Then, we describe in detail the concrete scheme composed of eight algorithms including *Init*, *GenKey*, *EncImage*, *GenIndex*, *GenTree*, *GenTrap*, *Macth*, *DeEnc_and_Verify*.

Init(1^λ)*:* Given the secure parameter λ, KGC generates two $(d+\alpha+\beta+1)\times(d+\alpha+\beta+1)$-dimensional random invertible real matrices M_A and M_B, as well as a random $(d+\alpha+\beta+1)$-dimensional binary vector B. They are defined as $prms=\{M_A, M_B, B\}$.

GenKey($p, prms$)*:* Upon registration of each IO or IU with its corresponding identity p, the KGC generates two random

Algorithm 1: Generate Index.

Input: SK_p, IS
Output: $\Gamma = \{v'_{1e}, v'_{2e}, \ldots, v'_{ne}\}$
for $1 \le i \le IS_n$ **do**
  Extract feature vector v_i
end
for $1 \le i \le v_n$ **do**
  Set access control structure coefficients $\{\rho_{i1}, \rho_{i2}, \ldots, \rho_{i\alpha}\}$
  Choose random binary vector $\{y_1, y_2, \ldots, y_\beta\}$
  v_i is extended to $v_{ie} = \{v_i, \|v_i\|, \rho_{i1}, \rho_{i2}, \ldots, \rho_{i\alpha}, y_1, y_2, \ldots, y_\beta\}$
  v_{ie} is cut into v_{ie1} and v_{ie2} according to Eq. (2)
  v_{ie1} and v_{ie2} are encrypted as $v^*_{ie} = \{M^\top_{p1} v_{ie1}, M^\top_{p2} v_{ie2}\}$
end
return $\Gamma = \{v'_{1e}, v'_{2e}, \ldots, v'_{ne}\}$.

invertible real matrices, denoted as M_{p1} and M_{p2}, for the registrant. KGC computes $M_{p3} = M^{-1}_{p1} M_A$ and $M_{p4} = M^{-1}_{p2} M_B$ respectively, and then generates an image encryption key K_{pe}. Thus, KGC generates

$$SK_p = \{M_{p1}, M_{p2}, K_{pe}, B\}, RK_p = \{M_{p3}, M_{p4}\}.$$

Finally, KGC sends the private key set SK_p to user I and RK_p to CS.

EncImage(SK_p, IP): Give image plaintext set $IP = \{IP_1, IP_2, \ldots, IP_n\}$, where n is the total number of images. The IO uses the image encryption key K_{pe} to encrypt the image to generate the cipher image collection $IC = \{IC_1, IC_2, \ldots, IC_n\}$. The encryption method uses the traditional AES encryption method. After the encryption is completed, IO uploads the ciphertext image collection to CS.

GenIndex(SK_p, IP): Given image plaintext set $IP = \{IP_1, IP_2, \ldots, IP_n\}$, IO runs Algorithm 1 to extract image features and obtain the set of encrypted image indexes Γ.

- First, IO_p utilizes the pre-trained convolutional neural network model to extract features from each image IP_i and denotes them as $v_i = \{v_{i1}, \ldots, v_{id}\}$, where d is the dimension of the feature vector.
- Second, IO_p expands the features. Specifically, feature $v_i = \{v_{i1}, v_{i2}, \ldots, v_{id}\}$ is expanded to $v_{ie} = \{v_{i1}, \ldots, v_{id}, \|v_i\|, \rho_{i1}, \ldots, \rho_{i\alpha}, y_1, y_2, \ldots, y_\beta\}$, where $\|v_i\|$ denotes the modulus length of vector v_i, $\{\rho_{i1}, \rho_{i2}, \ldots, \rho_{i\alpha}\}$ denotes the coefficients of the access control structure of this image and $\{y_1, y_2, \ldots, y_\beta\}$ denotes a random binary vector.
- Next, the extended features are sliced by IO_p. Specifically, the feature v_{ie} is sliced into two vectors v_{ie1} and v_{ie2} based on the binary vector B, using (2).

$$\begin{cases} v_{ie1}[k] = v_{ie2}[k] = v_{ie}[k], & \text{if } B[k] = 0; \\ v_{ie1}[k] + v_{ie2}[k] = v_{ie}[k], & \text{if } B[k] = 1. \end{cases} \quad (2)$$

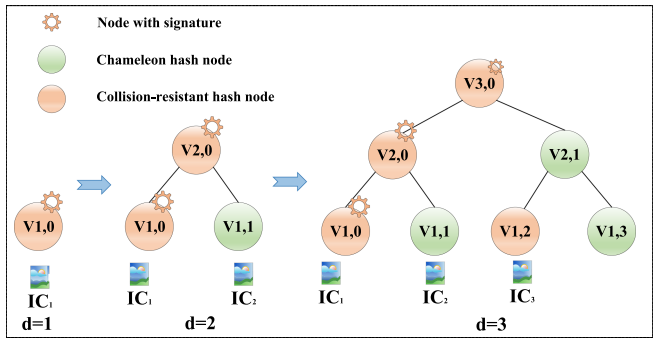

Fig. 4. Build process of dynamic verification tree.

- Finally, IO_p encrypts the index set using matrix M_{p1} and matrix M_{p2} to generate $v^*_{ie} = \{M^\top_{p1} v_{ie1}, M^\top_{p2} v_{ie2}\}$. Final generation of encrypted index set $\Gamma = \{v'_{1e}, v'_{2e}, \ldots, v'_{ne}\}$.

GenTree: In this section, we will elaborate on each of the three steps related to dynamic verification trees, including initialization, tree building, and appending images.

1) Init(λ): Given the security parameter λ, IO_p generates a key pair dpk and dsk. H_1 and H_2 represent two different normal collision-resistant hash functions. $CH = (chGen, Ch, Col)$ is the chameleon hash function. IO_p executes $chGen$ of the chameleon hash function to obtain public key cpk and private key csk.

2) Build(IC_n, dsk): We use $l_i = H_1(IC_i || id_{IC_i})$ to denote the hash value of the encrypted image IC_i. Before the images are deposited, the tree structure is empty and there are no nodes in the tree. When the first encrypted image is added, as shown in Fig. 4, the tree's depth d increases to 1. At this time, the root node of the tree $v_{1,0}$ is also a leaf node of the tree, and IO_p deposits the hash $H_2(l_1)$ of the encrypted image into this node and signs the current hash with IO_p's own private key dsk. The maximum number of images that the tree can currently hold is $2^{d-1} = 1$.

Before storing the second image, first, determine whether the deepest leaf node of the tree at the current depth is already full of data. The original tree is used as the left child of the new tree because the original tree is full, and IO_p then increases the depth of the tree by 1 level. The new tree then generating a virtual chameleon hash node $v_{1,1} = ch(m_{v_{1,1}}, r_{v_{1,1}})$ as the right child of the new root, where $m_{v_{1,1}}$ and $r_{v_{1,1}}$ are random numbers. The root of the new tree $v_{2,0} = H_2(v_{1,0} || v_{1,1})$ is a hash of the original root connected to the value of the virtual chameleon hash node and signed with its own private key dsk. The second image is deposited into the newly generated virtual chameleon hash node, and to keep the hash value unchanged, the hash collision $r^*_{v_{1,1}} = col(csk, m_{v_{1,1}}, r_{v_{1,1}}, l_2)$ needs to be computed and the random number $r_{v_{1,1}}$ in the virtual node $v_{1,1}$ is updated to $r^*_{v_{1,1}}$. IO_p stores virtual chameleon hash node $v_{1,1}$ in st.

Before depositing the third image, the same judgment is made whether the tree is full or not. Because the current tree is stored full again, IO_p increases the depth of the tree by 1 again. The original tree is then used as the left subtree of the new tree, while a

Algorithm 2: Build Dynamic Verification Tree.

Input: IC_n, dsk
Output: Dynamic Verification Tree DVT

for $1 \leq i \leq IC_n$ **do**
  $l_i = H_1(IC_i || id_{ICi})$
end
$sum \leftarrow 0$
$d \leftarrow 0$
// Generate the first node and deposit the first image.
$d \leftarrow d + 1$
$v_{d,0} \leftarrow H_2(l_1)$
Sign the hash using dsk and store it in the node.
$sum \leftarrow sum + 1$
for $2 \leq i \leq l_i$ **do**
  if *sum is a power of two* **then**
    $d \leftarrow d + 1$
    $v_{d-1,1} \leftarrow Ch(m_{v_{d-1,1}}, r_{v_{d-1,1}})$
    $v_{d,0} \leftarrow H_2(v_{d-1,0} || v_{d-1,1})$
    Sign the hash using dsk and store it in $v_{d,0}$.
  end
  Generate a subtree down the lowest right node $v_{i,j}$
  for *each child node in the subtree* **do**
    if *the node is a left child* **then**
      $v_{1,i-1} \leftarrow H_2(l_i)$
      $v_{m,n} \leftarrow H_2(v_{m-1,2n} || v_{m-1,2n+1})$
      // If the right child node is missing, generate the chameleon hash node with a random number and fill it.
      $v_{m-1,2n+1} \leftarrow Ch(m_{v_{m-1,2n+1}}, r_{v_{m-1,2n+1}})$
    else
      $r_{1,i-1} \leftarrow Col(dsk, v_{1,i-1}, r_{1,i-1}, l_i)$
    end
  end
  $sum \leftarrow sum + 1$
end
return DVT.

new chameleon hash node $v_{2,1} = ch(m_{v_{2,1}}, r_{v_{2,1}}$ is generated as the root of the right subtree. The new root $v_{3,0} = H_2(v_{2,0} || v_{2,1})$ is the hash of the connection value between the original root node and the new chameleon hash node. Generate a subtree along the right child node of the new tree, the left child node in this subtree employs the collision-resistant hash function, while the right child node populates the node with a virtual chameleon hash function. Specifically, $v_{1,2} = H_2(l_3)$, $v_{1,3} = ch(m_{v_{1,3}}, r_{v_{1,3}})$. At this time, it is necessary to update the random value $r_{v_{2,1}}$ of the node $v_{2,1}$ due to a change in its child value. IO_p stores virtual chameleon hash nodes $v_{2,1}$ and $v_{1,3}$ in st.

Thus, three images IC_1, IC_2, and IC_3 are stored in the dynamic hash tree. Further, if IO_p wants to add a fourth image, then calculate $r^*_{v_{1,3}} = col(csk, m_{v_{1,3}}, r_{v_{1,3}}, l_4)$ and replace $r_{v_{1,3}}$, and update $r_{v_{2,1}}$ in the same way.

In conclusion, the process of building a tree can be summarized in the following steps:

- First, before each new node is added, a judgment is made on the existing tree to determine if the tree is full. In the event that the tree is at full capacity, the original tree is incorporated as the left subtree of the newly constructed tree. The new virtual node is then utilized as the right child node of the aforementioned tree. The root node of the newly constructed tree is a hashed representation of the connection value between the root node of the original tree and the new virtual node, and it is signed. If the tree is not full, the subsequent procedures are executed directly.
- Then, commence from the root node and descend to the location where the image is to be inserted. The nodes in the path are to be completed in accordance with the subsequent instructions: 1) The non-leaf nodes' values are computed as hashes of the concatenated values of their left and right children; 2) While computing the left child, the collision-resistant hash function is applied, and while computing the right child, the chameleon hash function is utilized. The chameleon hash node is simultaneously marked by IO_p; 3) In cases where there are absent children in the calculation of the hash function, the resultant value is calculated virtually.
- Finally, the data is added to the target leaf node. Specifically, when the leaf node is situated on the left branch, the collision-resistant hash function denoted as $H_2(\cdot)$ is executed and stored. Conversely, when the leaf node is located on the right branch, the computation of $r_{new} = col(dsk, m, r_old, l)$ takes place, and the node's random value is subsequently upgraded with r_{new}.

IO_p uploads the generated dynamic verification tree DVT to the cloud server.

3) Append(DVT, Dsk, l): IO_p will append a new image l to the dynamic verification tree DVT:

- Initially, the signature of the root node in the verification tree is verified by IO_p.
- Then, IO_p stores the data in the last position of the lowest level in the dynamic verification tree. In the event that it is a left child node, generates a virtual right child node and stores the chameleon hash values of their connection values in their root nodes. Conversely, if it is a right child node, only the r value is computed and substituted, thereby keeping the chameleon hash value of the node.
- Eventually, IO_p transmits the new dynamic verification tree DVT to CS.

GenTrap(SK_t, I_q): The IU with identity t (denoted as IU_t) generates a trapdoor tp based on the query image I_q. Specifically, it includes the following steps:

- Initially, IU_t extract the features $v_q = \{v_{q1}, \ldots, v_{qd}\}$ of the query image I_q uploaded by itself, where d is the dimension of the feature vector.
- Second, according to the access control factor $\{\phi_{j1}, \phi_{j2}, \ldots, \phi_{j\alpha}\}$ of IU_t, IU_t expands the feature vector $v_{qe} = \{-2v_q, 1, \phi_{j1}, \phi_{j2}, \ldots, \phi_{j\alpha}, b_1, b_2, \ldots, b_\beta\}$, where $\{b_1, b_2, \ldots, b_\beta\}$ denotes a random binary vector.
- Next, IU_t slices the extended vector v_{qe}. As opposed to *GenIndex*, IU_t slices v_{qe} into v_{qe1} and v_{qe2} according

Algorithm 3: Append Data to Tree.

Input: DVT, dsk, l_i, l_i*
Output: DVT
if i *is an odd number* **then**
  delete $v_{1,i-1}$
  $v_{1,i-1} \leftarrow H_2(l_i)$
else
  delete $v_{1,i-1}$
  $r_{1,i-1} \leftarrow Col(dsk, v_{1,i-1}, r_{1,i-1}, l_i)$
end
Update all nodes on the path from the parent of $v_{1,i-1}$ upwards until the node with signature is updated and re-signed.
return DVT

to (3).

$$\begin{cases} v_{qe1}[k] + v_{qe2}[k] = v_{qe}[k], & \text{if } B[k] = 0; \\ v_{qe1}[k] = v_{qe2}[k] = v_{qe}[k], & \text{if } B[k] = 1. \end{cases} \tag{3}$$

- Finally, IU_t encrypts the index set using matrix M_{J1} and matrix M_{J2} to generate $tp = \{M_{J1}^{-1} v_{qe1}, M_{J2}^{-1} v_{qe2}\}$. IU_t sends tp to cs.

Match($RK_p, RK_t, tp, \Gamma, DVT$): CS will first re-encrypt the index and trapdoor according to the user identity based on the uploaded Γ and tp. Each index v_{ie}^* in the index set Γ sent by IO_p will be re-encrypted to $v_{ire}^* = \{M_{p3}^\top M_{p1}^\top v_{ie1}, M_{p4}^\top M_{p2}^\top v_{ie2}\}$ by the corresponding re-encryption key RK_p stored in CS. Similarly, the trapdoor tp sent by IU_t will be re-encrypted to $tp^* = \{M_{t3}^{-1} M_{t1}^{-1} v_{qe1}, M_{t4}^{-1} M_{t2}^{-1} v_{qe2}\}$ by the corresponding re-encryption key RK_t stored in CS. After processing (4), CS obtains the re-encrypted index and trapdoor.

$$\begin{cases} v_{ire}^* & = \{M_{p3}^\top M_{p1}^\top v_{ie1}, M_{p4}^\top M_{p2}^\top v_{ie2}\} \\ & = \{M_A^\top v_{ie1}, M_B^\top v_{ie2}\} \\ tp^* & = \{M_{t3}^{-1} M_{t1}^{-1} v_{qe1}, M_{t4}^{-1} M_{t2}^{-1} v_{qe2}\} \\ & = \{M_A^{-1} v_{qe1}, M_B^{-1} v_{qe2}\} \end{cases} \tag{4}$$

Then CS gets the top-k query results by doing an inner product operation on the encrypted index and trapdoor according to (5) to get the closest k results, where $\rho(\phi)$ is a polynomial constructed from $\rho_{i1}, \rho_{i2}, \ldots, \rho_{i\alpha}$ and $\phi_{j1}, \phi_{j2}, \ldots, \phi_{j\alpha}$ that implements fine-grained access control. Denote M as the upper bound of (5).

$$\begin{aligned} v_{ire}^* \cdot tp^* &= (M_A^\top v_{ie1}, M_B^\top v_{ie2}) \cdot (M_A^{-1} v_{qe1}, M_B^{-1} v_{qe2}) \\ &= v_{ie}^\top \cdot v_{qe} \\ &= \|v_i - v_q\|^2 - \|v_q\|^2 + \rho(\phi) + \sum_{i=1}^{\beta} y_\beta b_\beta. \end{aligned} \tag{5}$$

- If the IU_t is an authorized user, then ϕ is a root of $\rho(x)$. At this point the polynomial $\rho(\phi) = 0$ and $v_{ire}^* \cdot tp^* \leq M$. In this case, the IU_t with role factor $\phi_{j1}, \phi_{j2}, \ldots, \phi_{j\alpha}$ can access the target image.
- If the IU_t is a non-authorized user, then ϕ is not a root of $\rho(x)$. At this point the polynomial $\rho(\phi) \neq 0$ and $v_{ire}^* \cdot tp^* \gg M$. In this case, the IU_t with role factor $\phi_{j1}, \phi_{j2}, \ldots, \phi_{j\alpha}$ cannot access the target image.

Algorithm 4: Match.

Input: $RK_p, RK_t, tp, \Gamma, DVT$
Output: IC_r, π_r
// Trapdoor re-encryption.
$tp^* = \{M_{t3}^{-1} M_{t1}^{-1} v_{qe1}, M_{t4}^{-1} M_{t2}^{-1} v_{qe2}\}$
for *each* v_{ire}^* *in* Γ **do**
  $v_{ire}^* = \{M_{p3}^\top M_{p1}^\top v_{ie1}, M_{p4}^\top M_{p2}^\top v_{ie2}\}$
  Compute $v_{ire}^* \cdot tp^*$ and store the k smallest values of the result to IC_r.
end
for *each* IC_{ri} *in* IC_r **do**
  Find the root $v_{(i,j)}$ of the smallest subtree constructed from l_i to l_0 in DVT, and from $v_{(i,j)}$ to the target image location, store all the node values on the path, and the node values of their respective brothers in π_r.
end
return IC_r, π_r

After matching the images that match the access rights, the cloud server executes converting the previous image number, denoted as $i-1$, into a binary vector with d dimensions. Here, d corresponds to the depth of the tree. The principle of left 0 right 1 is employed in the search of the dynamic verification tree. After finding the corresponding numbered leaf node, the proof π_i, π_i is returned containing the root node of the leftmost child tree of the query node i to the leftmost leaf node as the path starting point until all nodes on the path of the query node, and their sibling nodes. The result set $IC_r = \{IC_{r1}, IC_{r2}, \ldots, IC_{rk}\}$ along with the proof set $\pi_r = \{\pi_{r1}, \pi_{r2}, \ldots, \pi_{rk}\}$ is transmitted by the cloud server to the IU_t.

DeEnc_and_Verify(IC_r, π_r): After IU_t receives the matching result and the proof, it first verifies the matching result. Specifically, IU_t first validates the signature of the root node, and if IO_p's signature can be verified, it calculates the hash value of the encrypted image. Specifically, if the node is located on the right-hand side of its parent node, then its hash value is determined through the utilization of the chameleon hash function denoted as $Ch(\cdot)$. Conversely, if the node is located on the left-hand side of its parent node, then its hash value is determined through the utilization of the anti-collision hash function denoted as $h(\cdot)$. Subsequently, the hash or chameleon hash is computed in a sequential manner, following the path outlined in the proof π until the root node is reached. The resulting value of the root node is then compared to the root node value provided in the proof π. If the two values are identical, the verification process is deemed successful. IU_t decrypts the image with the image encryption key for the successfully verified image and obtains the decrypted image of the query.

V. Security Analysis

This section presents a demonstration of the privacy of image, index, and trapdoor, as well as a discussion of the verifiability and multi-owner multi-user setting of the DCIRM scheme.

A. Image Privacy

In the DCIRM scheme, each image $IS_i(i \in [1,n])$ is encrypted by the symmetric key k_{ie} using AES. As long as the symmetric key k_{ie} is not leaked, the image privacy can be well protected.

B. Index and Trapdoor Privacy

Theorem 1:

DCIRM can guarantee the confidentiality of indexes and trapdoors under ciphertext-known Attack and Background-known Attack.

Proof: In generating the index, the image's expanded feature vector v_{ie} is divided into v_{ie1} and v_{ie2} by the binary vector B. Since the adversary does not know the vector B, he has to model v_{ie1} and v_{ie2} as two $(d+\alpha+\beta+1)$-dimensional vectors. The adversary then establishes equations $M_{I1}^{\top}v_{ie1}$ and $M_{I2}^{\top}v_{ie2}$ based on matrices M_{I1} and M_{I2}. The adversary cannot solve M_{I1} and M_{I2}. Therefore, in the *Known Ciphertext Attack*, the plaintext information will not be leaked under the condition that B_i, M_{I1} and M_{I2} are kept secret. In the same way, the trapdoor is also encrypted through B, M_{J1} and M_{J2}, so the trapdoor is also protected *Known Ciphertext Attack*.

When the opponent knows some plaintext information, he doesn't know the corresponding ciphertext information, so he needs to find the matrices M_{I1} and M_{I2} through the equation. If the adversary wants to construct equations to solve M_{I1} and M_{I2}, then he needs to guess the encrypted form of these plaintexts in all ciphertexts. However, this is exponentially expensive and impossible for the adversary with polynomial-time since there are at least $\mathcal{O}(n^{d+1})$ possible combinations of ciphertexts. Therefore, in the *Known Background Attack*, the plaintext information of the feature vectors will not be leaked. ■

Theorem 2: DCIRM can realize the query unlinkability among different query requests.

Proof: For different eigenvectors v_{q1} and v_{q2} extracted from the same image I_q through the neural network model, the user IU_t will add random binary vectors $\{b_1, b_2, \ldots, b_\beta\}$ when constructing the query trapdoor. After that, IU_t will also segment the expansion vector according to the binary vector B. If $B[k]=0$, then $v_{qe1}[k]+v_{qe2}[k]=v_{qe}[k]$. According to the above, if $tp_1=tp_2$, then $\Pr[tp_1=tp_2]=\frac{1}{2^\beta}\cdot\psi^{N_0}$, where β is the size of random binary vectors $\{b_1, b_2, \ldots, b_\beta\}$, N_0 are the number of 0 in the binary vector B, ψ is the probability of $v_{qe1}[k]$ and $v_{qe2}[k]$ corresponding to tp_1 and tp_2 when $B[k]$ equals 0. As ψ approaches 0, therefore Pr $[tp_1=tp_2]\to 0$. The adversary cannot find the connection between different trapdoors, therefore, DCIRM guarantees the unlinkability of trapdoors. ■

C. Multi-Owner Multi-User Setting

Theorem 3: DCIRM is privacy-preserving in the multi-owner multi-user setting if clients do not collude with CS.

Proof: In the DCIRM scheme, the parameter $prms$ is divided into SK and RK in the key generation stage, and each user's private key consists of two reversible matrices, a binary vector and an image encryption key. Users cannot infer other users' keys from their own keys. In addition, if the image user and CS do not conspire to obtain the re-encrypted trapdoor, then the adversary can only obtain the encrypted trapdoor and cannot decrypt it. Therefore, DCIRM is privacy-preserving in the multi-owner multi-user setting. ■

D. Verifiability Discussion

A dynamic verification tree is considered secure if the following two properties are satisfied.

1) It is impossible for the adversary to modify the node information in the dynamic verification tree. The aforementioned characteristic is referred to as structure-preserving.
2) It is not feasible for the adversary to add data to the dynamic verification tree without being detected. The aforementioned characteristic is referred to as the one-way property.

- *Setup:* The challenger executes the *Init*(1^λ) procedure, which produces a set of key pairs dsk and dpk, respectively. The key dpk is subsequently conveyed to the adversary credited as $\mathcal{A}$.
- *Append:* Adversary $\mathcal{A}$ selects n pieces of data $d_1, d_2, \ldots, d_n$ to transmit to the challenger. The challenger appends the data to the dynamic verification tree and subsequently furnishes adversary $\mathcal{A}$ with the corresponding proof $\pi_r=\pi_{r_1},\pi_{r_2},\ldots,\pi_{r_n}$.
- *Output:* Adversary $\mathcal{A}$ generates π_{r^*}. The game is won by the adversary $\mathcal{A}$ if the conditions hold that $1\le r^*\le n$, $\pi_{r^*}\notin\pi_r$, and the verification of π_{r^*} is successful. The variable Adv_A^{sp} represents the probability of success for adversary $\mathcal{A}$ in the game.

Definition 3: If Adv_A^{sp} is negligible for any value n and any PPT opponent $\mathcal{A}$, then the dynamic verification tree is structure-preserving.

Similarly, to describe the unidirectionality of the dynamic verification tree, we use a game analogous to the above.

- *Setup:* The challenger executes the **Init**(1^λ) procedure, which produces a set of key pairs dsk and dpk, respectively. The key dpk is subsequently conveyed to the adversary $\mathcal{A}$.
- *Append:* Adversary $\mathcal{A}$ selects n pieces of data $d_1, d_2, \ldots, d_n$ to transmit to the challenger. The challenger appends the data to the dynamic verification tree and subsequently furnishes adversary $\mathcal{A}$ with the corresponding proof $\pi_r=\pi_{r_1},\pi_{r_2},\ldots,\pi_{r_n}$.
- *Output:* Adversary $\mathcal{A}$ generates π_{r^*}. The game is won by the adversary $\mathcal{A}$ if the conditions hold that $n\le r^*$ and π_{r^*} is successful. The variable Adv_A^{sp} represents the probability of success for adversary $\mathcal{A}$ in the game.

Definition 4: If Adv_A^{sp} is negligible for any value n and any PPT opponent $\mathcal{A}$, then the dynamic verification tree is one-way.

Theorem 4: Assuming that H_2 is a collision-resistant hash function and CH is a collision-resistant chameleon hash function, the dynamic verification tree is structure-preserving.

Proof: In the event of a contradiction where the structure of the dynamic verification tree is not preserved, it is possible for the adversary $\mathcal{A}$ to achieve a non-negligible probability Adv_A^{sp}

TABLE III
COMPARISON OF TIME COMPLEXITY

Schemes	Genkey	GenIndex	GenTrap	Match
DCIRM	$\mathcal{O}(d^3)$	$\mathcal{O}(nd)$	$\mathcal{O}(d)$	$\mathcal{O}(\log n)$
EPCBIR	$\mathcal{O}(d^2)$	$\mathcal{O}(nd^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(\log n)$

of winning the game. Assuming the probabilities of collision between the hash function H_2 and the chameleon hash function as τ_{H_2} and τ_{CH}, respectively, τ_{H_2} is negligible because the hash function H_2 is collision-resistant. Likewise, τ_{CH} is also negligible. Therefore, the probability Adv_A^{sp} of the opponent winning the game is negligible. This statement presents a contradiction to our prior assumption. It can be asserted that the dynamic verification tree is structure-preserving. ■

Theorem 5: If the hash function H_2 and the chameleon hash function CH are both collision-resistant, then the dynamic verification tree is one-way.

Proof: In the event that the dynamic verification tree fails to meet the one-way criterion, adversary $\mathcal{A}$ has the ability to produce a hash node that is not present within the tree and get $\pi_{r^*} \notin \pi_r$, where $n \leq r^*$. This means that adversary $\mathcal{A}$ can get a hash collision. However, because the hash function and the chameleon hash function in the dynamic verification tree are both collision-resistant hash functions, the dynamic verification tree is one-way in this situation. ■

According to Theorems 4 and 5, the dynamic verification tree is secure against PPT adversaries, which guarantees the verifiability of the dynamic verification tree.

VI. PERFORMANCE ANALYSIS

In this section, we analyze the theoretical efficiency and experimental efficiency of the DCIRM scheme.

A. Theoretical Analysis

In Table III we analyzed the algorithm complexity of each module in the DCIRM scheme and compared it with the EPCBIR scheme[4].

GenKey: The KGC generates a corresponding key for each user during the key generation stage. The matrix inversion operation accounts for the majority of the algorithm's computation costs. $\mathcal{O}(d^3)$ is the algorithm's time complexity. The process of generating a key for a user is run by the KGC only once during the user registration phase, so the overall performance of the DCIRM scheme is not affected. The algorithm complexity of the EPCBIR scheme is $\mathcal{O}(d^2)$, but EPCBIR does not support multi-owner and multi-user settings.

GenIndex: The matrix and vector multiplication operations are performed once for each feature during the index generation process of each IO, so the complexity is $\mathcal{O}(nd)$. Where n represents the total number of image datasets. The algorithm complexity of the EPCBIR scheme is $\mathcal{O}(nd^2)$. This is because the EPCBIR scheme uses CBIR technology to generate indexes, which will cause higher computational overhead.

GenTrap: Similar to the index generation process, the IU uses its query image to generate a trapdoor, which requires a matrix and vector multiplication operation. The complexity of this process is $\mathcal{O}(d)$. The algorithm complexity of the EPCBIR scheme is $\mathcal{O}(d^2)$, which is also due to the greater computational overhead caused by the use of CBIR to generate indexes.

Match: Since both DCIRM and EPCBIR use secure kNN technology for image matching, the time complexity of both in the image matching stage is $\mathcal{O}(\log n)$.

TABLE IV
PERFORMANCE OF FEATURE EXTRACTION

Feature	size	depths	time1	time2
Googlenet	49.7MB	22	111.12S	47.19S
VGG16	527MB	16	820.82S	277.91S
ResNet18	44.6MB	18	89.53S	39.74S
ResNet50	97.7MB	50	123.59S	56.55S

[1] "time1" indicates the time required to extract features from 4000 images.
[2] "time2" indicates the time required to extract features from 2000 images.

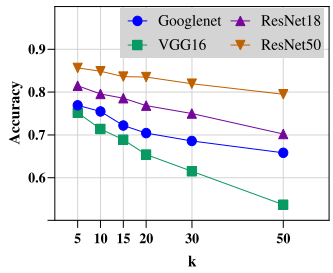

Fig. 5. Retrieval accuracy under different models.

B. Experimental Analysis

First, we will explain the environment, dataset, and indicator of the experiment.

- *Environment:* Our testing platform is a PC with a 3.2 GHz CPU and 16 GB of RAM. The Python programming language is utilized with the Numpy Library.
- *Dataset:* Caltech 256 is a widely used real image dataset that contains 256 different categories of images with a total of about 30,000 pieces. These image categories cover many fields, such as animals, plants, furniture, food, vehicles, and so on. Each image has a resolution of 300x200 or 200x300 pixels.
- *Indicator:* Precision at top-k is a method for assessing the precision of a proposal. It is mathematically defined as $P@k = num\_correct/k$ [5], where k represents the number of images returned and $num\_correct$ represents the number of correct images returned. By varying the values assigned to k, the retrieval accuracy of the scheme was evaluated.

Then, we analyze the retrieval accuracy of the whole scheme. Table IV shows the characteristics of different neural network models and the time required for them to extract features from images of different scales. Fig. 5 depicts the influence of different feature extraction models on retrieval accuracy when the feature dimension is reduced to 128 by PCA. When we use the ResNet50

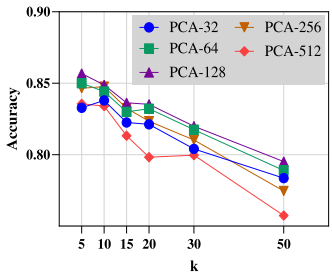

Fig. 6. Retrieval accuracy under different dimensions.

TABLE V
RETRIEVAL ACCURACY

k	5	10	15	20	30	50
PCA-32	0.8328	0.8380	0.8225	0.8212	0.8040	0.7836
PCA-64	0.8520	0.8442	0.8300	0.8322	0.8175	0.7889
PCA-128	0.8568	0.8488	0.8363	0.8352	0.8199	0.7952
PCA-256	0.8465	0.8476	0.8325	0.8234	0.8103	0.7746
PCA-512	0.8356	0.8340	0.8132	0.7982	0.7996	0.7574

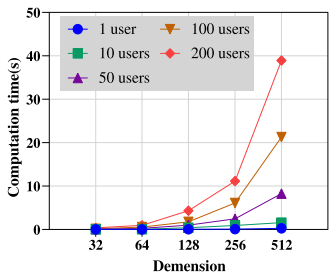

Fig. 7. *GenKey* computation costs.

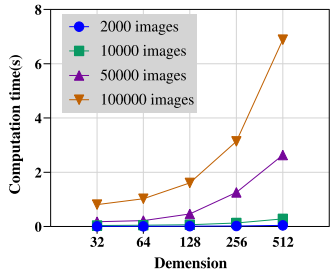

Fig. 8. *GenIndex* computation costs.

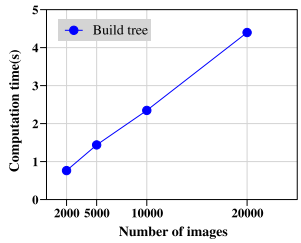

Fig. 9. *Build* computation costs.

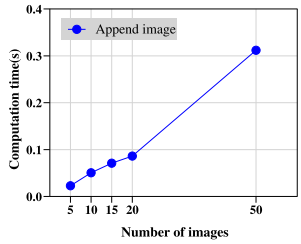

Fig. 10. *Append* computation costs.

model to extract features, retrieval accuracy is the highest. In Fig. 6, we show the retrieval accuracy of extracting features using the ResNet model. When the feature vector length is 128, the retrieval accuracy is the highest. However, when the feature vector length is 512 or 32, the retrieval accuracy is the lowest relative to other dimensions. In general, the retrieval accuracy of DCIRM can reach between 80% and 90% under this model. More accurate data are listed in Table V.

Following that, we examine the performance of each DCIRM module.

1) Genkey: Fig. 7 shows the time taken by the system to generate the key. For each user, the KGC will create an image encryption key, two random reversible real number matrices, and conduct matrix multiplication twice. KGC must build more matrices and carry out more operations as the number of users who have registered rises, which adds to the overall processing time. Additionally, when the feature vector dimension of the image rises, a larger encryption matrix must be generated, adding time and computational overhead. When the dimension of the feature vector is 128, the time cost of generating a key for a single user is 0.057 s, and that for 200 users is 4.32 s.

2) GenIndex: In Fig. 8, we show the process of generating an image index. Specifically, IO will expand, segment, and encrypt the image feature vectors after extracting the image's features. The size of the feature vectors and the number of images will influence how long it takes to generate the indexes in this process. When the data volume is 10000 images and the feature vectors are of all dimensions, DCIRM can generate the encryption index in 0.3 s. When there are 100,000 images, a 128-dimensional feature vector may create an encrypted index in 1.6 s and a 512-dimensional feature vector in 7 s. We also conducted an experimental investigation of how long it takes to process various image data in a given dimension. When the feature vector is 128 dimensions, and the data amount is 2000,10000,50000 and 100000 respectively, the time required is 0.012 s, 0.065 s, 0.464 s, and 1.610 s respectively.

3) Tree: The time needed to generate and update the dynamic verification tree is shown in Figs. 9 and 10, respectively. With more images, it will take longer to create a tree, linearly. The dynamic verification tree takes 2.35 s to create when there are 10,000 images altogether. In less than 0.05 s, 10 images are added to the dynamic verification tree.

4) Gentrap: The time consumption of image users using query images to generate trapdoors is shown in Fig. 11. In DCIRM, the size of the feature vectors has a positive correlation with how long it takes to generate trapdoors. Specifically, when the feature vector has 128 dimensions, generating a trapdoor

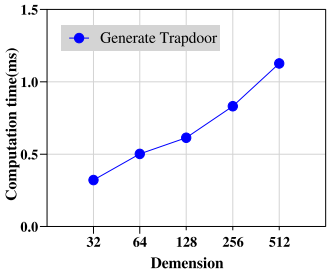

Fig. 11. *GenTrap* computation costs.

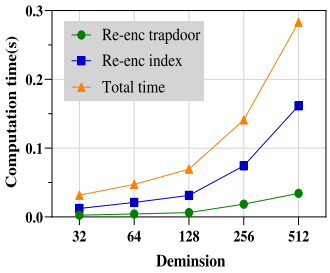

Fig. 12. *Match* computation costs under different dimensions.

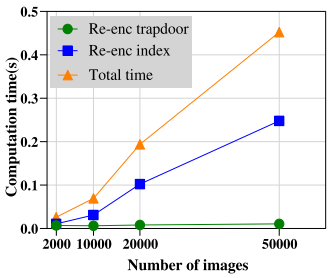

Fig. 13. *Match* computation costs under different image numbers.

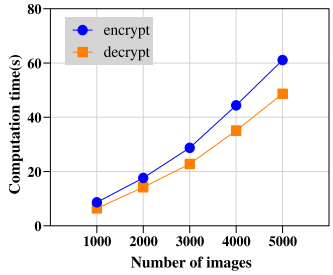

Fig. 14. *EncImage* and *DecImage* computation costs.

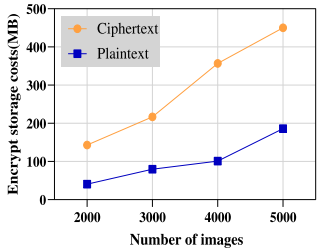

Fig. 15. *EncImage* and *DecImage* storage costs.

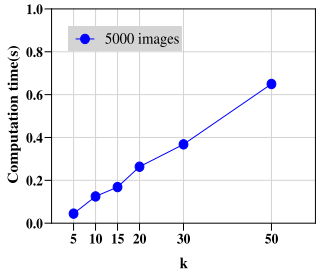

Fig. 16. *Verify* computation costs.

takes 0.61 ms of time, and when the feature vector has 512 dimensions, generating a trapdoor takes 1.12 ms of time.

5) Match: When there are 10,000 images overall, Fig. 12 displays the time needed to match the indexes and trapdoors of various dimensions. The lengthening of the feature vector will result in longer times for matching, re-encryption index, and re-encryption trapdoor. Specifically, when the feature vector is 128 dimensions, the matching can be completed within 0.07 s. Although the feature vector reaches 512 dimensions, DCIRM can still complete the matching process of 10000 images in 0.3 s. Fig. 13 shows the influence of different image data on matching time when the feature vector is 128 dimensions. Because the trapdoor is generated by the query image, the time consumed by re-encrypting the index is not affected by the number of images. The total time and the time consumed by re-encrypting the index are linear with the number of images. Specifically, 20,000 images can be re-encrypted in 0.1 s and matched in 0.25 s.

6) Enc_and_dec: Figs. 14 and 15 show the time consumption and storage consumption of image encryption and decryption respectively. It takes 61 s to encrypt 5000 images and 49 s to decrypt them. 5000 encrypted images need 450 MB storage, while plaintext images need 186 M storage.

7) Verify: The amount of time needed to verify images is shown in Fig. 16. It takes 0.26 s to verify 20 images and 0.65 s to verify 50 images when the entire data volume is 5000 images.

C. Comparative Analysis

Both schemes [17] (namely SVMIS) and DCIRM support verifiability. But DCIRM not only supports verifiable functions in dynamic scenarios, but also supports lightweight access control functions. In addition, compared with SVMIS which can only be applied to single-owner multi-user scenarios, DCIRM uses re-encryption settings to achieve availability in multi-owner multi-user scenarios. Both DCIRM and scheme [18] (namely DATVS) achieve dynamic verifiability, but DCIRM focuses on the scenario of encrypted image retrieval and implements multi-owner and multi-user settings. We also compare the time required for index generation, trapdoor formation, matching, and image update operations with existing encrypted image retrieval schemes. Specifically, since scheme [22] (namely PIC) and SVMIS are both encrypted image retrieval scheme, and both

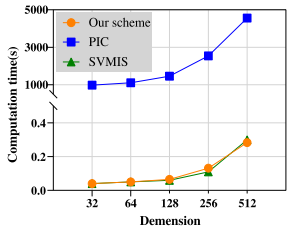

Fig. 17. Computation cost comparison of *GenIndex*.

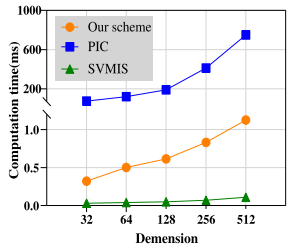

Fig. 18. Computation cost comparison of *GenTrap*.

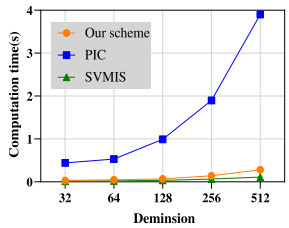

Fig. 19. Computation cost comparison of *Match*.

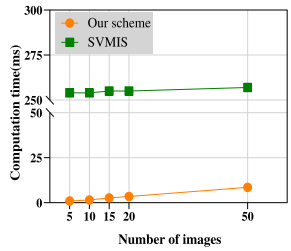

Fig. 20. Computation cost comparison of *Append*.

TABLE VI
COMPARISON OF RETRIEVAL ACCURACY

Schemes	Feature extraction	Feature dimension	Accuracy
DCIRM	CNN	128	0.85
[16]	CNN	128	0.83
[25]	SURF	64	0.65

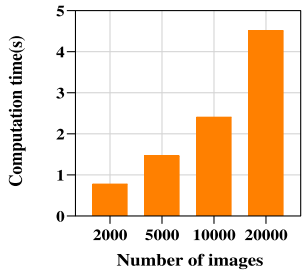

Fig. 21. Computation cost for IO.

SVMIS and DCIRM support verifiability, we compared DCIRM with PIC and SVMIS.

GenIndex and GenTrap: Figs. 17 and 18 show the computational consumption of *GenIndex* and *GenTrap* under different feature dimensions, respectively. Although PIC uses distributed and parallel computing technology to improve the efficiency based on SIFT feature extraction, when the total number of images reaches 10000, it still needs more than 1000 s to generate the index. However, the DCIRM scheme and SVMIS use CNN to extract features for index generation. When the feature dimension is 128, it takes 0.064 s and 0.060 s respectively. Different from SVMIS, the DCIRM scheme realizes the lightweight access control function in the process of index generation, while ensuring the efficiency of performance approximation. Similarly, the time required to generate a trapdoor also grows as the feature dimension does, and the time consumption of PIC is significantly greater than that of the DCIRM scheme and SVMIS.

Match: We compared the time consumption of the matching process between the DCIRM scheme and other schemes. Fig. 19 shows that when the feature dimension is 128 dimensions, the matching process of PIC needs 0.99 s, while that of SVMIS only needs 0.35 s. DCIRM can keep 0.69 s under the condition of realizing multi-owner and multi-user and access control functions. More functionality is realized and the operation efficiency is ensured.

Append: Fig. 20 shows the calculation consumption of the DCIRM scheme and SVMIS when adding images respectively. Because SVMIS uses the traditional Merkel tree for integrity verification when the image owner appends an image, a complete Merkel tree needs to be regenerated every time the image is appended, so a lot of calculation costs will be incurred. DCIRM only needs 8.53 ms to update the authentication tree when adding 50 images. Therefore, the DCIRM scheme effectively realizes the dynamic integrity verification of the image.

Accuracy: We compare the retrieval accuracy with several schemes in Table VI, and the results show that the DCIRM scheme can achieve various functions and the accuracy can reach 85%. As a result, the DCIRM scheme is practicable.

D. Computational Cost of Entities

Figs. 21, 22, and 23 show the total computational cost of IO, IU and CS respectively. The computational cost of IO mainly includes encrypting indexes and generating dynamic verification trees. Fig. 21 shows that under the 128-dimensional vector, when the number of images is 5000, IO takes 1.47 s. The computational cost of IU includes generating trapdoors and verifying images. Since DCIRM supports multi-owner and multi-user settings, compared with traditional schemes, image users only

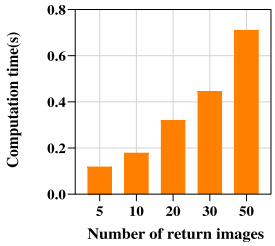

Fig. 22. Computation cost of IU.

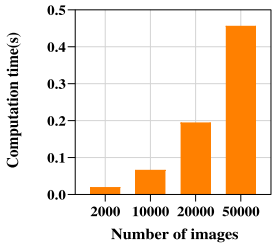

Fig. 23. Computation cost of CS.

need to generate a trapdoor once, and do not need to generate corresponding trapdoors for each image owner. Not only is the computational cost reduced, but the image owner does not need to transmit his or her key to the image user through a secure channel, thus reducing the communication cost. Fig. 22 shows that when the number of returned images is 20, IU takes 0.32 s. The computational cost of CS mainly includes re-encryption index, re-encryption trapdoor and matching. Fig. 23 shows that when the number of images is 20000, CS takes 0.19s.

VII. Conclusion

In this paper, we design an encrypted image retrieval scheme that allows dynamic result verification in multi-owner and multi-user settings while incorporating access control measures. Compared with the traditional encrypted image retrieval scheme, the DCIRM scheme uses a re-encryption scheme to realize multi-owner and multi-user settings and creates a dynamic verification tree that supports dynamic verifiability using the chameleon hash function. In addition, we have achieved lightweight access control. Finally, we proved that the DCIRM scheme is safe and efficient through analysis and experiment.

In the future, we will work on enhancing the scheme's retrieval efficiency by creating an index tree.

References

[1] N. V. Gudivada and V. V. Raghavan, "Content based image retrieval systems," *Computer*, vol. 28, no. 9, pp. 18–22, 1995.
[2] W. M. Arnold, M. Smeulders, S. Worring, A. S. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.
[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.
[4] Z. Xia, N. N. Xiong, A. V. Vasilakos, and X. Sun, "EPCBIR: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing," *Inf. Sci.*, vol. 387, pp. 195–204, 2017.
[5] X. Li, Q. Xue, and M. C. Chuah, "CASHEIRS: Cloud assisted scalable hierarchical encrypted based image retrieval system," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[6] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.
[7] J. Yuan, S. Yu, and L. Guo, "SEISA: Secure and efficient encrypted image search with access control," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2083–2091.
[8] Q. Liu, X. Nie, X. Liu, T. Peng, and J. Wu, "Verifiable ranked search over dynamic encrypted data in cloud computing," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Serv.*, 2017, pp. 1–6.
[9] G. Ateniese and B. De Medeiros, "Identity-based chameleon hash and applications," *Financial Cryptogr.*, vol. 3110, pp. 164–180, 2004.
[10] W. Lu, A. Swaminathan, A. L. Varna, and M. Wu, "Enabling search over encrypted multimedia databases," *Media Forensics Secur.*, vol. 7254, pp. 404–414, 2009.
[11] M. Li et al., "InstantCryptoGram: Secure image retrieval service," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 2222–2230.
[12] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling verifiable Boolean range queries over blockchain databases," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 141–158.
[13] D. Catalano and D. Fiore, "Practical homomorphic message authenticators for arithmetic circuits," *J. Cryptol.*, vol. 31, no. 1, pp. 23–59, 2018.
[14] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1083–1095, Nov./Dec. 2018.
[15] L. Rao, H. Zhang, and T. Tu, "Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated merkle hash tree," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 451–463, May/Jun. 2020.
[16] Y. Li, J. Ma, Y. Miao, L. Liu, X. Liu, and K.-K. R. Choo, "Secure and verifiable multikey image search in cloud-assisted edge computing," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5348–5359, Aug. 2021.
[17] J. Xu, F. Li, K. Chen, F. Zhou, J. Choi, and J. Shin, "Dynamic chameleon authentication tree for verifiable data streaming in 5G networks," *IEEE Access*, vol. 5, pp. 26448–26459, 2017.
[18] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 522–530.
[19] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 226–234.
[20] Y. Miao, R. Deng, X. Liu, K.-K. R. Choo, H. Wu, and H. Li, "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1667–1680, Jul./Aug. 2021.
[21] L. Zhang et al., "PIC: Enable large-scale privacy preserving content-based image search on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 11, pp. 3258–3271, Nov. 2017.
[22] S. ManjuNair and M.S. Rajasree, "Fine-grained search and access control in multi-user searchable encryption without shared keys," *J. Inf. Secur. Appl.*, vol. 41, pp. 124–133, 2018.
[23] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Apr. 2019.
[24] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure KNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.
[25] 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, Jan.-Mar. 2018.

Chenyang Mao received the BS degree with the Department of Internet of Things Engineering from Xi'an University of Science and Technology, Xi'an, China, in 2020. He is currently working toward the master's degree with the School of Information Science and Technology, Hangzhou Normal University. His current research interests include cloud computing security and applied cryptography.

Zhonghua Shen received the BS degree in mathematics education and the MS degree in basic mathematics from Hangzhou Normal University, Hangzhou, in 1996 and 2002, respectively, and the PhD degree from the Institute of Education, Xiamen University, Xiamen, China, in 2020. He is currently a professor with the School of Mathematics, Hangzhou Normal University. His research interests include information security, applied cryptography, and high education management research.

Kefei Chen received the BS and MS degrees from Xidian University, China, in 1982 and 1985, respectively, and the PhD degree from Justus-Liebig University Giessen, Germany, in 1994. From 1996 to 2012, he was a professor with Shanghai Jiao Tong University, China. He is currently a professor with Hangzhou Normal University, China. His main research areas are cryptography and theory and technology of network security.

Yong Liu received PhD degree in communication and information system from Xidian University, Xi'an, China, in 2021. He is currently an associate professor with Hangzhou Normal University. His research interests include data center networks, software defined networking, and optical interconnected networks.

Qian Meng received the MS degree with the Department of Mathematics from Hangzhou Normal University, Hangzhou, China, in 2016, and the PhD degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China, in 2020. She is currently a lecturer with the Department of Mathematics in Hangzhou Normal University. Her research interests include information security and applied cryptography.

Fuqun Wang received the BS degree in Information and Computing Science and the MS degree in Pure mathematics from the Zhengzhou University, Zhengzhou, Henan, China, in 2003 and 2006, respectively, and the PhD degree in Information Security from the Institute of Information Engineering of Chinese Academy of Sciences, Beijing, China, in 2016. He is currently an associate professor with the School of Mathematics, Hangzhou Normal University. His research interests include fully homomorphic cryptography, lattices-based cryptography, public key cryptography, etc.