Segmented Hash-based Privacy-Preserving Image Retrieval Scheme in Cloud-Assisted IoT

Xiehua Li, Wanting Lei, Wenjuan Tang, Yingzhu Wang, Xiaoju Yang, Xin Liao

***Abstract*—The proliferation and application of the Internet of Things (IoT) have significantly increased the production and processing of large volumes of images. Due to privacy concerns, such data are often encrypted before being outsourced to third parties for subsequent retrieval and processing. Existing encrypted image retrieval schemes primarily aim to improve search accuracy but are limited in terms of retrieval efficiency and storage expenses, rendering them challenging to be implemented on IoT devices. To address these issues, we propose a Segmented Hash-based Privacy-Preserving Image Retrieval (SHPIR) scheme that enables accurate and efficient retrieval of encrypted images in IoT networks. First, we propose a feature extraction model based on Convolutional Neural Network (CNN) to generate low-dimensional segmented hash codes that represent both image categories and features accurately. Next, we design the cryptographic hierarchical index structure based on the segmented hash codes to improve retrieval efficiency and accuracy. We also design a secure Hamming distance computation algorithm and employ the Learning With Errors (LWE)-based secure k-Nearest Neighbour (kNN) algorithm to preserve the privacy of feature vectors and file-access patterns. Finally, we provide a security analysis verifying that the SHPIR scheme can protect image privacy as well as indexing and query privacy. Additionally, we constructed a real IoT environment to test the practical effectiveness and applicability of our scheme. Experimental evaluation indicates that our proposed scheme outperforms existing state-of-the-art schemes in retrieval accuracy, search efficiency, and storage costs, making it more suitable for real-world IoT applications.**

***Index Terms*—Segmented hash code, encrypted image retrieval, convolutional neural network, learning with errors.**

I. Introduction

With the rapid expansion of Internet of Things (IoT) applications in various fields, such as healthcare and industry, the collection and sharing of multimedia data, particularly image data, have grown exponentially[1], [2]. However, resource-constrained IoT devices face challenges in managing the massive amount of data collected daily. Therefore, cloud-assisted data outsourcing services have been employed for IoT applications [3], [4]. Despite the benefits of outsourcing data, once the data are uploaded to cloud servers, they become vulnerable to various attacks[5]. The recently released IBM *Cost of a Data Breach Report* shows that the average global cost of data breach reached USD 4.35 million in 2022, which is a 12.7% increase from the last year, and the highest ever noted across the history of IBM reports[6]. Image data can reveal more sensitive information, such as facial information, monitoring images, locations, and other privacy information, which makes it riskier to upload plaintext images to cloud servers. Encryption can ensure image confidentiality, but it makes image processing and retrieval challenging and less accurate. In practical applications, cloud-assisted IoT services need to provide accurate and efficient image retrieval schemes that can protect privacy in various image processing scenarios. This has motivated researchers to study privacy-preserving content-based search and retrieval methods that can process image data efficiently without revealing sensitive information.

The main challenges for secure image retrieval in the IoT applications lies in the constrained resources of end devices, which impede efficient data processing. Traditional privacy-preserving content-based image retrieval (CBIR) methods predominantly address retrieval accuracy and security issues, often neglecting the application limitations inherent in IoT environments. Consequently, existing approaches continue to face significant challenges regarding retrieval efficiency and storage space constraints in IoT applications.

Low Retrieval Efficiency. The low retrieval efficiency of existing CBIR schemes can be attributed to several factors. Certain CBIR schemes prioritize enhancing the security of image retrieval by initially encrypting the image and subsequently extracting features from the ciphertext image. While these methods offer improved privacy protection, they are characterized by high computational complexity, rendering them impractical for deployment in IoT environments. For instance, Hsu *et al.* utilized homomorphic encryption to safeguard images and enable the cloud server to extract SIFT (Scale Invariant Feature Transform) features from the encrypted image for retrieval[7]. Similarly, many schemes apply homomorphic encryption on images to support feature extraction and similarity computation[8][9]. The retrieval efficiency of homomorphic encryption is suboptimal due to its high computational cost. Alternative approaches, such as probabilistic encryption[10], bag-of-encrypted-words(BOEW)[11], local binary pattern (LBP) histograms[12] and histogram of oriented gradients (HOG) features[13], also employ complex encryption and feature extraction strategies, further diminishing retrieval efficiency. Although these solutions for ciphertext images can offload the intensive tasks of feature extraction and similarity comparison to cloud servers, their intricate encryption and feature extraction methods reduce computational

Manuscript received xx, xx; revised xx, xx ; accepted xx, xx. Date of publication xx, xx; date of current version xx, xx. This work was supported by the National Natural Science Foundation of China under Grant 62202157, 92067104, U22A2030, and is also supported by the Hunan Provincial Funds for Distinguished Young Scholars under Grant 2024JJ2025, Hunan Provincial Natural Science Foundation under Grant 2021JJ30140. (*Corresponding author: Xiehua Li.*)

Xiehua Li, Wanting Lei, Wenjuan Tang,Yingzhu Wang, Xiaoju Yang and Xin Liao are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (email: beverly@hnu.edu.cn, wantinglei@hnu.edu.cn, wenjuantang@hnu.edu.cn, wyz22@hnu.edu.cn, yang_xiao_ju@hnu.edu.cn, xinliao@hnu.edu.cn)

efficiency. Furthermore, feature extraction based on ciphertext images compromises retrieval accuracy. Additionally, these solutions necessitate that data owners execute specified image encryption algorithms before outsourcing images to the cloud, which is impractical for resource-constrained IoT devices.

High Storage Cost. To enhance the accuracy of CBIR, existing solutions typically involve the extraction of high-dimensional image features, which consequently increases the storage space required for these features. For instance, the first proposed the ciphertext image retrieval scheme employed order-preserving encryption (OPE) and Min-hash[14], utilizing a large number of visual words as the feature presentations to improve the accuracy. Other methods, such as Fisher vectors[15], MPEG-7 visual descriptors[16], also use high-dimensional vectors as feature representations. With the rapid development of deep learning, Convolutional Neural Network (CNN) models have emerged as a prevalent tool for image feature extraction[17][18]. Compared to prior approaches, the utilization of pre-trained CNN models facilitates a more precise depiction of image content, consequently enhancing search accuracy[19][20]. All these schemes employ high-dimensional features from the plaintext images to build indexes, thus enabling more accurate retrieval over encrypted images. For instance, in ref.[3], a comparison of retrieval accuracy across different feature vector dimensions was presented. When the feature dimension was reduced from 128 to 32, the retrieval accuracy for returning the top-5 most similar images out of 2000 decreased from 55% to less than 50%. Consequently, employing high-dimensional feature vectors not only diminishes retrieval efficiency but also increases the index storage overhead and escalates storage costs in cloud-assisted IoT environments. Thus, while high-dimensional feature extraction improves retrieval accuracy, it simultaneously reduces retrieval efficiency and increases index storage overhead.

Low Retrieval Accuracy. Recent research on privacy protection in CBIR within IoT networks has explored various advanced techniques. These include the use of blockchain and secure multi-party computation(SMC) to protect medical image retrieval[21], the application of local sensitive hashing and secure k-nearest neighbor (kNN) methods to enhance the search process in cloud-assisted IoT environments[22], and the delegation of computationally intensive tasks, such as Scale-Invariant Feature Transform (SIFT) feature extraction to cloud servers[23]. Additionally, thumbnail-preserving encryption (TPE) has been utilized to improve the accuracy and efficiency of ciphertext image retrieval with the assistance of cloud servers[24]. The employment of similarity thresholds has also been proposed to safeguard the privacy of user queries[25], which enhances the privacy protection in CBIR. However, existing solutions in the IoT environment still face limitations regarding retrieval accuracy and efficiency and are not well-suited for dynamic updates.

In real-world cloud-assisted IoT networks, content-based image retrieval (CBIR) schemes encounter several challenges due to the vast volume of data collected and processed. These challenges include the necessity to reduce computational complexity to facilitate rapid processing of massive data, improve retrieval efficiency to conserve server resources, minimize storage space requirements to reduce cloud server resource utilization, enhance retrieval accuracy to increase system usability, and support dynamic update to adjust to the real-world IoT applications. Additionally, ensuring the security of images, feature vectors, and search queries is paramount.

To address the above issues, we propose a Segmented Hash-based Privacy-preserving Image Retrieval (SHPIR) scheme. The proposed scheme enhances the performance of CBIR in terms of feature extraction, retrieval efficiency, accuracy, and security. Key technical advancements include: 1. an improved CNN-based image feature extraction model is proposed to provide more accurate and lower-dimensional image feature representations; 2. a hierarchical indexing structure is designed to balance the requirements of retrieval efficiency and accuracy; 3. a combination of secure Hamming distance and LWE-based encryption is applied to guarantee the security and query unlinkability of feature vectors; 4. additionally, this SHPIR scheme supports dynamic updates in cloud-assisted IoT environments, further improving its applicability and robustness. Compared to existing CBIR schemes, the SHPIR scheme effectively reduces the dimensionality of feature vectors, thereby enhancing image retrieval efficiency, decreasing storage costs, and preserving high retrieval accuracy. These attributes render SHPIR particularly well-suited for application in IoT environments, surpassing the performance of existing methodologies.

Our contributions in this paper can be summarized as follows:

- *Novel Privacy-preserving CBIR Scheme*. This paper proposes a novel image retrieval scheme, termed SHPIR, that can support privacy protection. The scheme reduces the dimensionality of feature vectors by improving the feature extraction model, which improves the efficiency of image retrieval and reduces the storage cost while ensuring the accuracy of image retrieval, rendering it better adapted to the IoT environment. Additionally, the design of the secure Hamming distances algorithm not only guarantees the privacy of query vectors but also hides the access pattern between query vectors and stored images.
- *Improved Feature Extraction Model*. The feature extraction model is designed to generate low-dimensional, two-level segmented hash code, which can give a more precise representation of images. Through this model, the category hash code representing the image category and the vectors representing each image feature can be obtained simultaneously. The finely designed feature extraction model enables accurate presentation of each image with fewer feature vectors, which improves the accuracy of image similarity calculation. Moreover, the low-dimensional hash codes reduce the feature vector extraction and storage overhead on both IoT devices and cloud servers.
- *Security Analysis and Performance Evaluation*. Theoretical security analysis of access pattern protection is presented. Furthermore, the computational efficiency, storage overhead, and retrieval accuracy of the proposed SHPIR scheme are evaluated using a real-world dataset. The performance comparison between SHPIR and other state-of-

the-art encrypted content-based image retrieval schemes is conducted and the experimental results validate the security, effectiveness, and accuracy of the proposed SHPIR scheme.

The rest of this paper is organized as follows. In Section II, we briefly discuss related work. Section III describes the system structure and design ideas. In Section IV, we describe the hash code generation algorithm and the feature extraction model. Section V provides the main algorithms, followed by the detailed search and retrieval procedure of SHPIR. Section VI gives the formal security analysis and proof of SHPIR. We report our experimental and comparison results in Section VII. Finally, we conclude this paper in Section VIII.

II. Related Work

The main challenges of privacy-preserving CBIR technology are to improve retrieval accuracy and efficiency and to guarantee information security. Several related works have designed schemes to address these issues.

Improve retrieval accuracy. Lu *et al.*[14] introduced the first CBIR-based encrypted image search scheme utilizing visual word frequencies as image features, incorporating secure inverted indexing and secure min-hash to protect these features. Subsequently, Xia *et al.* [26] proposed a scheme employing Color Layout Descriptor (CLD) and Edge Histogram Descriptor (EHD) as image features, although the limited content representation of CLD and EHD resulted in unsatisfactory retrieval accuracy. To improve feature accuracy, HSV(Hue, Saturation, Value) histogram was used as image features, and a combination of k-nearest neighbor (kNN) and SVM(Support Vector Machine) was employed for similar image searches [27]. Various features, such as text descriptions, semantic matrices, GPS data, and multi-feature fusion of contextual information, can also be used for image retrieval [28][29][30]. Recently, adoption of machine learning, particularly convolutional neural networks(CNN), has further improved retrieval accuracy by extracting image features more effectively [17][31][3]. However, a significant challenge remains in the similarity ranking of encrypted feature vectors. To address this, some Learning With Errors (LWE)-based secure kNN encryption methods have been proposed [32][18][19], enabling Euclidean distance computation with encrypted features. Although these LWE-based schemes supported relevance ranking, their search efficiency was hindered by the high dimensionality of feature vectors, making them difficult to apply on resource-constrained devices. To improve retrieval accuracy for encrypted medical images, Zhu et al. [33] proposed the Privacy-preserving Mahalanobis Distance Comparison (PMDC) that supported Marcian distance comparison under ciphertext form.

Improve retrieval efficiency. Ref. [34] proposed a clustering algorithm to initialize K-means clustering using the centers of affinity propagation (AP) clusters, addressing the sensitivity of K-means clustering to initial values and achieving more accurate classification. Yin *et al*. [20] adopted Zhu's CAK-means (Combination of AP and K-means clustering) algorithm to construct a tree-based index, applying LWE encryption to achieve higher query accuracy and retrieval efficiency. Yuan *et al*. [15] proposed a retrieval scheme supporting access control, utilizing a K-means clustering algorithm to construct a tree-based index structure to expedite the searching process. Xia *et al.*[16] used local features to represent images, Earth Mover's Distance (EMD) to evaluate image similarity, and Locality Sensitive Hashing (LSH) to improve search efficiency. Other schemes, such as Ref. [35] and Ref. [26] have also used local sensitive hash to construct indexes for enhanced retrieval efficiency. Guo *et al.* [36] proposed a scheme to find the exact nearest neighbor over encrypted medical images, by computing a lower bound on the Euclidean distance with respect to the data's mean and standard deviation to eliminate low-similarity candidates. Index structure also impacts search accuracy. Traditional solutions typically use linear index to improve search accuracy [37], or tree-based index to achieve high search efficiency[17][20]. However, these two types of index have some limitations: linear indexing requires the server to explore the entire database to return the top-k retrieval results, which is inefficient for massive image retrieval in IoT applications, and tree-based indexing is time-consuming to build and does not always offer satisfactory retrieval accuracy.

Protect the image and feature privacy. To support security computation of ciphertext, homomorphic encryption(HE) is commonly used in secure image retrieval schemes. Hsu *et al.*[7] proposed a secure image retrieval scheme that encrypted images using homomorphic encryption, but still allowed cloud servers to extract the Scale Invariant Feature Transform(SIFT) features for retrieval. Despite the advantages, schemes based on homomorphic encryption are often impractical due to high computational complexity and low efficiency. Other encryption methods have been proposed to achieve feature extraction over ciphertext images. Ferreira *et al.*[10] proposed Image Encryption Scheme (IES) that prioritized texture features over color features, using probabilistic encryption for texture features and deterministic encryption for color features. The cloud server then directly extracted features from the encrypted image for retrieval. Xia *et al.*[11] proposed a scheme based on the Bag-of-Encrypted-Words(BOEW) model, encrypting images through pixel value replacement, block replacement and pixel replacement within blocks. The cloud server built encrypted BOEW models by clustering the local color histograms and counting the frequency of visual word occurrences as image features. There are some other schemes that employed secure Local Binary Pattern(LBP) to calculate Euclidean or Manhattan distance on encrypted images to improve searching security and accuracy [38][12]. Ref. [38] combined the homomorphic encryption and scalar-product-preserving encryption (ASPE) to maintain key confidentiality. To authenticate and verify the image retrieval process, Guo *et al.*[39] proposed an authentication algorithm that used the Merkle randomized k-d tree and the Merkle inverted index to ensure the integrity of query results.

Although many encrypted image retrieval schemes have been proposed, they predominantly focus on enhancing accuracy, efficiency, and security in general-purpose applications and are not fully adapted to the IoT environment. Recently, some researches have been conducted on image privacy pro-

tection and secure retrieval in IoT networks. Shen *et al.*[21] presented a privacy-preserving medical image retrieval scheme by using the blockchain and secure multi-party computation techniques in IoT networks. Sibahee *et al.*[22] designed a lightweight CBIR search scheme using speed-up robust feature(SURF) as the feature vector that can be used in cloud-assisted IoT environment. The local sensitive hash and secure kNN were employed to improve search efficiency. Wang *et al.*[23] proposed a privacy-preserving image search framework for mobile cloud computing, which shifted SIFT feature extraction, index generation, and image search onto the cloud server. Song proposed a threshold-based token generation process to safeguard the similarity ranking results of user queries[25], which enhances the privacy protection in CBIR.

The existing schemes that are applied in the IoT environment still have some limitations on retrieval accuracy, and retrieval efficiency, and are often unsuitable for dynamic updating[40]. To address these issues, we propose the Segmented Hash-based Privacy-preserving Image Retrieval (SHPIR) scheme for cloud-assisted IoT networks. This scheme aims to enhance image privacy while improving retrieval accuracy and efficiency. Table I provides a comparative summary of SHPIR and other existing schemes, where SH denotes the Segmented Hash codes utilized in our approach.

TABLE I
COMPARATIVE SUMMARY OF TYPICAL CBIR SOLUTIONS

Schemes	$\mathbb{P}_1$	$\mathbb{P}_2$	$\mathbb{P}_3$	$\mathbb{P}_4$	$\mathbb{P}_5$
[3]	CNN	Tree	kNN	✓	✓
[17]	CNN	Tree	Hamming	✓	
[20]	CNN	Tree	kNN	✓	
[21]	EHD	Linear	SMC		
[22]	SURF	LSH	Euclidean		
[23]	SIFT	LSH	SMC		
[25]	CNN	Linear	kNN	✓	
[26]	MEPG-7	Linear	kNN		
[37]	CNN	Linear	kNN	✓	
SHPIR	SH	Hierarchical	Hamming +kNN	✓	✓

$\mathbb{P}_1$: Feature type; $\mathbb{P}_2$: Index structure; $\mathbb{P}_3$: Feature protection method; $\mathbb{P}_4$: Access pattern protection; $\mathbb{P}_5$: Dynamic updates.

III. The System Model and Design Ideas

A. System Model

The system model of SHPIR is shown in Fig.1. In the proposed SHPIR system, there are five principals: Image owner, authentication center, edge server, user, and cloud server.

Image owner is equipped with IoT device that generates images. The image owner extracts the segmented hash codes with the pre-trained feature extraction model and then uploads the segmented hash codes to the edge servers.

Authentication center (AuC) is a trusted third party that authorizes users, generates keys, and distributes keys to the image owners and search users.

Edge Server is responsible for building a pre-trained CNN model and extracting image features on plaintext images with this model. The edge server also needs to construct an index and encrypt both images and the index. Then edge server submits the secure index, encrypted images and retrieval parameters to the cloud server.

User runs the pre-trained CNN model to extract features of queried images, builds a trapdoor and submits it to the cloud server. After receiving the search results from the cloud server, the search user decrypted results with the authorized keys.

Cloud Server stores encrypted index and images. When receiving queries, the cloud server implements a retrieval algorithm and returns top-k most similar images to search user.

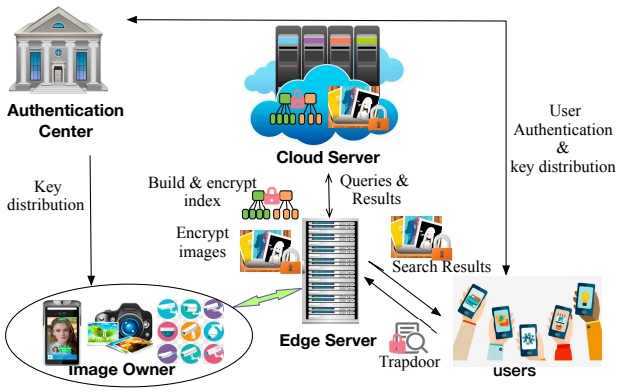


Fig. 1. System model of proposed SHPIR

B. Threat Model

In this scheme, we assume that image owners and edge servers are fully trusted. This assumption is warranted by the significant resource and power limitations inherent to end devices within the IoT environment, which restrict their functional capacities to basic data collection and transmission. Consequently, these end devices are incapable of executing tasks such as feature extraction and advanced encryption. Therefore, reliance on fully trusted edge servers for subsequent data processing is essential. In this case, we assume that the edge server is trusted. However, it is important to recognize that if the image owner or the edge server is compromised, the security of the entire system is weakened, as an attacker could access plaintext information. Therefore, edge servers and image owners should be adequately protected.

Conversely, the cloud server in our system is considered "honest but curious." While it faithfully executes the protocol, it also attempts to analyze the data and search the index for additional information related to the plaintext images. Based on the information accessible to the cloud server, we define the following two threat models for our scheme. Users within this scheme are not considered trustworthy and can only access authorized information. Additionally, we assume that the encryption algorithms and secret keys are secure and not susceptible to compromise.

Based on the information available to the adversaries, we consider the following two threat models.

- *Known ciphertext attack model*. In this model, the cloud server has access only to encrypted information, such

as encrypted images, encrypted indexes, and query trapdoors. However, it can collect and analyze the stored ciphertext data to infer information related to image content and feature vectors. In addition, the cloud server can obtain information about the access pattern by analyzing the similarity between query trapdoors and the relationship between the trapdoors and the returned images.

- *Known background attack model*. This is a stronger model in which the cloud server can learn more information than that is accessible in *known ciphertext attack model*, such as information about ranking results. In this model, the cloud server may also analyze the query and search result pairs, utilizing statistical analysis to infer the plaintext content of the queried images, index and feature vectors.

IV. Image Feature Extraction Model of SHPIR

In the CBIR-based searchable encryption scheme, feature vectors are used for searching similar images and should be stored in the cloud server. Therefore, under the premise of ensuring accurate query, the lower the vector dimension is, the smaller the storage space required and the less the computational resources demanded, which is conducive to improving search efficiency in IoT networks. To reduce the dimensionality of the feature vector, this paper designs a low-dimensional segmented hash code as the representation vector of each image, and a CNN-based feature extraction model is proposed to obtain this segmented hash code. This two-level segmented hash code consists of two parts, which are the image category hash code and the image feature hash code. Due to the limited resources of IoT devices, all image feature training tasks are offloaded to edge servers to obtain the feature extraction model. After building up the CNN-based feature extraction model, edge servers distribute this pre-trained model to IoT devices for query image feature extraction.

A. Category hash code generation

The accuracy of image retrieval depends on its probability of finding the correct category, in this paper, we define a category hash code to represent the classification information of each image. The design of category hash code needs to satisfy certain criteria. First, the Hamming distance between any two category hash codes is required to be large enough to reduce the probability of category search error. However, the dimensionality of the hash codes needs to be kept low to enhance the retrieval efficiency and reduce the index storage overhead. An apparent contradiction is formed between these two requirements. For example, given the number of image categories of 50, if we set the minimum Hamming distance between image categories to 4, we can get 50 hash codes with dimension 12. But if the minimum Hamming distance is set to 7, although the error of the category search is reduced, the 50 categories need a hash code with a dimension greater than 16 to represent.

To generate the shortest hash code to represent all image categories in the image set, we propose an algorithm for hash code generation. Suppose $H^{(1)}$ is the hash code to represent the image category. If the number of image categories is K, the length of the category hash code is d_1. To minimize the false positive rate of finding the correct image category, the minimum Hamming distance between any two category hash codes is t. Our proposed random hash code generation algorithm needs to find the hash code that can satisfy Eq.1.

$$d(H_i^{(1)}, H_j^{(1)}) > t, i \neq j \tag{1}$$

Where $d()$ is the Hamming distance between category hash codes, $H_i^{(1)}, H_j^{(1)}$ are the hash codes representing two different image categories. The process repeats until there are K hash codes have been obtained. The category hash code generation algorithm is described in Alg.1.

Algorithm 1 Category hash code generation

Input: hash code length d_1, number of categories K, t
Output:
codeset $H^{(1)} = \{H_1^{(1)}, \dots H_i^{(1)}, \dots, H_K^{(1)}, \}, H_i^{(1)} \in \{0, 1\}$

```
while i ≤ K do
    H_i^(1) = GenHashCode();
    shouldAdd = True;        // Flag to determine if H_i^(1) should be added to H^(1)
    for each hash code H_j^(1) ∈ H^(1) do
        distance = HammingDistance(H_i^(1), H_j^(1));
        if distance < t then
            shouldAdd = False;
            Break;
        end if
        if shouldAdd then Add H_i^(1) to H^(1);
        end if
    end for
    i = i + 1;
end while
return H^(1);
```

TABLE II
Category hash code generation time

Hash code length t	12	24	24	24	32	64
Hamming distance	4	4	6	8	4	4
Generation time(s)	ϕ	0.035	0.044	1.55	0.036	0.071

GenerateHashCode() is the function that generates a new hash code, and HammingDistance($H_i^{(1)}, H_j^{(1)}$) calculates the Hamming distance between two hash codes $H_i^{(1)}$ and $H_j^{(1)}$. For different d_1 and t, the length of hash codes may not be a certain number. In our experiment, we choose 200 categories from the Caltech256 dataset. The experimental generation time for 200 hash codes of different length is shown in Table II.

From the experiment result, we can see that for 200 image categories, when the minimum Hamming distance is set to be $t = 4$, the minimum length of the category hash code is 24 and there is no hash code generated if we set the code length to 12. As the hash code length and Hamming distance increase, the code generation time increases accordingly.

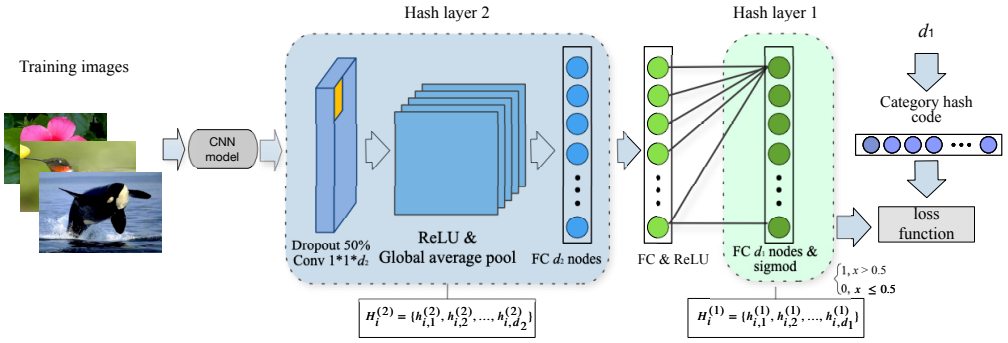


Fig. 2. Segmented hash codes training model of SHPIR

B. Segmented hash codes generation

In this paper, each image category will be assigned a hash code generated by Alg.1 as its category hash code, and all images in that category will be trained with our SHPIR training model to obtain the same category hash code. SHPIR model modifies the training network by adding a latent code generation layer to get its category hash code. The proposed SHPIR training model is shown in Fig.2. In our model, we add two layers after the basic CNN model to train the segmented hash codes. **Hash Layer 2** is used to train the image features so that they can represent the feature information of each image. **Hash Layer 1** is the classification layer used to train the image features, in this way, images belonging to the same class could obtain the same assigned category hash code.

The output of **Hash Layer 2** is the feature vectors of an image. The output of **Hash Layer 1** is the image category hash code. The output of **Hash Layer 1** is $H_i^{(1)} = \{h_{i,1}^{(1)}, h_{i,2}^{(1)}, \ldots, h_{i,d_1}^{(1)}\}$, which represents the category hash code of image i with length d_1. The output of **Hash Layer 2** is $H_i^{(2)} = \{h_{i,1}^{(2)}, h_{i,2}^{(2)}, \ldots, h_{i,d_2}^{(2)}\}$, which represents the feature vector of image i with length d_2.

SHPIR uses DenseNet-169 as the basic CNN model. DenseNet uses a dense block structure, which makes the network narrower and less parameterized. This structure connects each layer with all previous layers, allowing more efficient use of image features, making the convolutional neural network deeper, more accurate and efficient to train, and is able to solve the gradient disappearance problem. As shown in Fig.2, the **Hash Layer 2** consists of a $1 \times 1 \times d_2$ convolutional layer, a ReLU activation function and a global pooling layer, the **Hash Layer 1** combines a d_1 nodes full connection layer and a $sigmoid$ activation function. For each image in the training set, through **Hash Layer 1** and **Hash Layer 2**, we can obtain a d_1-dimensional vector $H_i^{(1)}$ containing the classification information of image i, and a d_2-dimensional vector $H_i^{(2)}$ containing the feature of image i. Finally, we binarize the outputs of both **Hash Layer 1** and **Hash Layer 2** into hash codes, with outputs greater than 0.5 sets to 1 and those less than 0.5 set to 0. The final representation vector of each image is described as:

$$\begin{aligned} f_i &= (H_i^{(1)} || H_i^{(2)}), \\ &= \{h_{i,1}^{(1)}, h_{i,2}^{(1)}, \ldots, h_{i,d_1}^{(1)}, h_{i,1}^{(2)}, h_{i,2}^{(2)}, \ldots, h_{i,d_2}^{(2)}\} \end{aligned} \tag{2}$$

TABLE III
DEFINITIONS OF NOTATIONS

Symbol	Description
$\mathcal{M} = \{m_i\}_{i=1}^N$	Plaintext image set with the number of images N
$\mathcal{M}' = \{m_i'\}_{i=1}^N$	Encrypted image set of $\mathcal{M}$
$f_i = \{H_{i,d_1}^{(1)} \|\| H_{i,d_2}^{(2)}\}$	feature vectors of image i with length of d_1 and d_2
M, M^{-1}	random matrix and its invertible matrix
$f_q = \{H_{q,d_1}^{(1)}, H_{q,d_2}^{(2)}\}$	query vectors with length d_1, d_2
$\tilde{f}_i, \tilde{f}_q$	encrypted f_i, f_q
M_{IO}, M_{IO}'	IO's random matrices where $M = M_{IO} \cdot M_{IO}'$
M_U, M_U'	User's random matrices where $M^{-1} = M_U \cdot M_U'$
$C_1, C_2, \ldots, C_l$	Category centers after K-means clustering
$I, \tilde{I}$	plaintext and encrypted indexes of $\mathcal{M}$

Where "||" is the concatenation character.

V. Similarity Search Based on Segmented Hash Codes

A. Definitions and Notations

There are two mathematical operations used in SHPIR, they are defined as follows.

Definition 1: For $\alpha \in \mathbb{Z}$, define $\lceil \alpha \rfloor$ to be the nearest integer of α, $\lceil \alpha \rfloor_q$ to be the nearest integer of α with modulus q.

Definition 2: For a vector $\vec{V}$ (or a matix M), define $|max(V)|$ (or $|max(M)|$) to be maximum absolute value of their elements.

The notations used in SHPIR is described in Table III.

B. System Structure of SHPIR

We propose a similarity retrieval scheme based on segmented hash codes, which can achieve efficient and accurate encrypted image retrieval while protecting the privacy and security of images and indexes.

The system infrastructure of SHPIR is a tuple of several algorithms, namely: **GenKey, ExtractFea, GenIndex, EncImage, GenQuery, Search, DecImage**, which are depicted in Fig.3. In step ① AuC generates keys and assigns them to IoT devices(image owners), edge server, cloud server and search users. Step ②, image owner extracts features to generate the segmented hash codes with the proposed training model, and

uploads the segmented hash codes to edge server. In ③- ④, edge server collects the segmented hash codes from different image owners, and builds the hierarchical index, then it encrypts the index and images with various encryption methods and uploads them to cloud server. In step ⑤, the authorized search user generates trapdoor from the query images and then submits it to the edge server. Edge server forwards queries to the cloud server. The cloud server implements searching algorithm and return similar ciphertext images to the querying user, which is shown by the step ⑥. In step ⑦, the search user decrypts returned results to obtain images similar to the query images.

In the SHPIR scheme, the heavy image dataset training task is completed by the edge server, and the trained model is sent to each IoT device (image owner) for feature vector extraction.

C. Privacy protection of SHPIR

To guarantee the security of images, feature vectors, and break the linkability between queries and stored data, we use various algorithms to protect different types of data, the details of these algorithms are described below.

GenKey(SK). The AuC generates different encryption keys and distributes them to the image owners and users.

- First, AuC generates two d_1-dimensional vectors $V = \{v_1, v_2, \ldots, v_{d_1}\}$, $R = \{r_1, r_2, \ldots, r_{d_1}\}$, and a permutation π, where $v_j \in [-p_1, p_1]$, $r_j \in [-p_2, p_2]$, $p_1 >> p_2$ are randomly selected, $sk_1 = \{V, R, \pi\}$.
- Second, AuC randomly choses $\gamma \in \mathbb{Z}_{p_1}$, and generates matrix M and its inverse form M^{-1}. $M_{IO}, M'_{IO} \in \mathbb{Z}^{2d_2 \times 2d_2}$ are randomly divided from M. $M_U, M'_U \in \mathbb{Z}^{2d_2 \times 2d_2}$ are randomly divided from M^{-1}, $sk_2 = M_{IO}$.
- Then, AuC generates AES key sk_3, Finally, AuC issues the key set $SK_U = (sk_1, M_U, \gamma, sk_3)$ to users, M'_{IO}, M'_U to cloud server via secure channel.

ExtractFea $(\{m_i\}_{i=1}^N) \rightarrow (\{f_i\}_{i=1}^N)$. For each image m_i in the set $\{m_i\}_{i=1}^N$, its feature f_i is extracted with the SHPIR training model, $f_i = H_i^{(1)}||H_i^{(2)} = \{h_{i,1}^{(1)}, h_{i,2}^{(1)}, \ldots, h_{i,d_1}^{(1)}, h_{i,1}^{(2)}, h_{i,2}^{(2)}, \ldots, h_{i,d_2}^{(2)}\}$.

GenIndex$(\{m_i\}_{i=1}^N, SK, M_{IO}, M'_{IO}) \rightarrow \tilde{I}$. The index generation algorithm consists of three steps, the first step is to extract feature of each image in $\{m_i\}_{i=1}^N$ with our SHPIR training model, the second step is to cluster the images with their category hash codes $H_i^{(1)}$ and build up the hierarchical index, the third step is to encrypt the index.

- Feature extraction. For each image m_i in the set $\{m_i\}_{i=1}^N$, its feature f_i is extracted by the SHPIR training model, $f_i = H_i^{(1)}||H_i^{(2)} = \{h_{i,1}^{(1)}, h_{i,2}^{(1)}, \ldots, h_{i,d_1}^{(1)}, h_{i,1}^{(2)}, h_{i,2}^{(2)}, \ldots, h_{i,d_2}^{(2)}\}$.
- Category clustering and hierarchical index building. K-means algorithm is applied to cluster the images using $H_i^{(1)}$ as input. Since $H_i^{(1)}$ contains the category information of each image, after using K-means, we can get K class centers $\{C_l\}_{l=1}^K$. The index is constructed hierarchically, with $C_l = \{c_{l,1}, c_{l,2}, \ldots, c_{l,d_1}\}$ as the top category level. In building the image index, when the Hamming distance between $H_i^{(1)}$ of an image m_i and the center C_l of a category l is the smallest, the image will be placed in the category represented by that C_l. In each category, there are several images that are organized with their feature vectors $H_i^{(2)} = \{h_{i,1}^{(2)}, h_{i,2}^{(2)}, \ldots, h_{i,d_1}^{(2)}\}$ as the second layer of the hierarchical index.
- Index encryption. In order to protect the feature information, C_l and $H_i^{(2)}$ are encrypted with different methods. Image owner first replaces all 0 in C_l with -1 to get $\hat{C}_l$, and then encrypts it with sk_1. The encryption is defined with Eq.3.

$$\tilde{C}_l = \pi(V \cdot \hat{C}_l \cdot R) = \pi\{(v_j \cdot \hat{c}_{l,j} \cdot r_j)\}_{j=1}^{d_1} \tag{3}$$

The feature vectors $H_i^{(2)}$ are encrypted with LWE-based encryption. When encrypt $H_i^{(2)}$, image owner first extends $H_i^{(2)}$ to $\hat{H}_i^{(2)}$.

$$\begin{aligned}\hat{H}_i^{(2)} = \{&h_{i,1}^{(2)}, h_{i,2}^{(2)}, \ldots, h_{i,d_2}^{(2)}, \\ &-\frac{1}{2}\sum_{j=1}^{d_2} h_{i,j}^{(2)}, \alpha_1, \alpha_2, \ldots, \alpha_{d_2-1}\}\end{aligned} \tag{4}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_{d_2} \in \mathbb{A}_{p_2}$ are random numbers chosen by the image owners for each image feature vector $H_i^{(2)}$. Then, image owner encrypts $\hat{H}_i^{(2)}$ and gets the ciphertext $\tilde{H}_i^{(2)}$.

$$\tilde{H}_i^{(2)} = (\gamma \cdot \hat{H}_i^{(2)} + \zeta_i) \cdot M_{IO} \tag{5}$$

Here $\gamma \in \mathbb{Z}_{p_1}, p_1 \gg p_2, \gamma \gg 2|max(\zeta_i)|. \zeta_i \in \mathbb{Z}_{p_2}^{2d}$ is an integer error vector randomly chosen from probability distributions.

The final encrypted index is represented as $\tilde{I} = \{\tilde{C}_l, \tilde{H}_i^{(2)}\}$, $i \in [1, N]$. After index generation and encryption, image owner uploads the encrypted images and index to the cloud server.

EncImage$(\{m_i\}_{i=1}^N, sk_3) \rightarrow (\{m'_i\}_{i=1}^N)$. Image owner encrypts each image in $\{m_i\}_{i=1}^N$ with key sk_3, then sends the ciphertext image set $\{m'_i\}_{i=1}^N$ to the edge server. The edge server collects images from different image owners and forwards them to the cloud server.

D. Image retrieval

The image query request is generated by search users who first extract feature vectors using the same pre-trained model as the image owner. Then, the search users generate retrieval trapdoors using **GenQuery** and send them to the edge server. Finally, the edge server waits for a random slot of time and forwards the retrieval trapdoors of several search users together to the cloud server so that the file access pattern can be hiden from the cloud server. When the cloud server receives search queries, it implements the **Search** function and returns the top-k ranked results.

GenQuery$(m_q, SK) \rightarrow Q = \{\tilde{f}_q, W_q\}$. When search user intends to search similar images, he first extracts feature vectors $f_q = (H_q^{(1)}||H_q^{(2)})$ from the query image m_q, and then

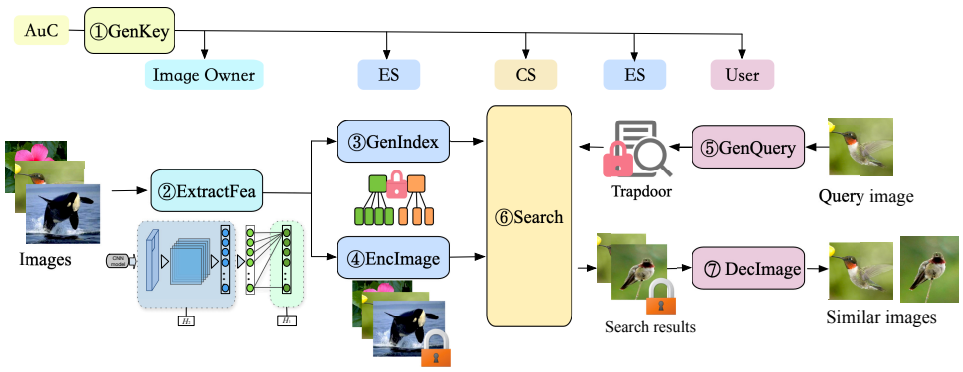


Fig. 3. System infrastructure of proposed SHIPR

generates the trapdoor $Q = \{\tilde{f}_q, W_q\}$ with the key SK_U. The process of trapdoor generation is described by the following steps.

- First, search user replaces all 0 in $H_q^{(1)}$ with -1 and get $\hat{H}_q^{(1)}$.
- Second, search user generates d_1-dimensional random vector $U_q = \{u_{q,1}, u_{q,2}, \ldots, u_{q,d_1}\}$, $u_{q,j} \in [-p_1, p_1]$. Then user encrypts $\hat{H}_q^{(1)}$ with the permutation π and random vector $R = \{r_1, r_2, \ldots, r_{d_1}\}$.

$$\tilde{H}_q^{(1)} = \pi(U_q \cdot \hat{H}_q^{(1)} \cdot R) = \pi\{(u_{q,j} \cdot \hat{h}_{q,j}^{(1)} \cdot r_j)\}_{j=1}^{d_1} \quad (6)$$

  In different queries, search user generates different U_q, so that our SHPIR scheme is resilient to chosen plaintext attack.
- Third, search user extends and encrypts the feature vector $H_q^{(2)}$.

$$\hat{H}_q^{(2)} = \{\rho_q h_{q,1}^{(2)}, \ldots, \rho_q h_{q,d_2}^{(2)}, \rho_q, \beta_1, \ldots, \beta_{d_2-1}\} \quad (7)$$

$$\tilde{H}_q^{(2)} = M_U \cdot (\gamma \cdot \hat{H}_q^{(2)\top} + \zeta_j^\top) \quad (8)$$

  where $\rho_q, \beta_1, \beta_2, \ldots, \beta_{d_2-1} \in \mathbb{Z}_{p_2}$, $\rho_q > 0$, are random numbers selected by search user for each query. $\zeta_j \in \mathbb{Z}_{p_2}^{2d}$ is an integer error vector randomly chosen by search user. The encrypted query feature $\tilde{f}_q$ is described in Eq.7

$$\tilde{f}_q = \tilde{H}_q^{(1)} || \tilde{H}_q^{(2)} \quad (9)$$

- Finally, based on the received secret key SK_U, search user generates $W_q = \pi \cdot \{(v_j + u_{q,j})\}_{j=1}^{d_1}$.

Search user sends $Q = \{\tilde{f}_q, W_q\}$ to cloud server for image search and retrieval. Since query $\tilde{f}_q$ is composed of two segments, the search process contains two steps.

Search$(\tilde{I}, Q) \rightarrow R'$. In the retrieval process, the cloud server first finds the category to which the queried image belongs according to its $\tilde{H}_q^{(1)}$, and then finds the corresponding image in this category and returns the top-k most similar results. The detailed process is described as follows.

- First, cloud server calculates the secure Hamming distance between $\tilde{H}_q^{(1)}$ and $\tilde{C}_l$, the calculation details are shown below.

$$\begin{aligned} EnD(\tilde{H}_q^{(1)}, \tilde{C}_l) &= \sum_{j=1}^{d_1} \{[(\tilde{h}_{q,j}^{(1)} + \tilde{c}_{l,j}) \bmod w_j] \oplus 0\} \\ &= \sum_{j=1}^{d_1} \{[\pi(u_{q,j}\hat{h}_{q,j}^{(1)} r_j + v_j \hat{c}_{l,j} r_j) \bmod w_j] \oplus 0\} \\ &= \sum_{j=1}^{d_1} \{[(u_{q,j}\hat{h}_{q,j}^{(1)} r_j + v_j \hat{c}_{l,j} r_j) \bmod (u_{q,j} + v_j)] \oplus 0\} \end{aligned} \quad (10)$$

  $EnD(*,*)$ is the secure Hamming distance calculation algorithm. The result of $EnD(\tilde{H}_q^{(1)}, \tilde{C}_l)$ equals to the plaintext Hamming distance between $H_q^{(1)}$ and C_l. When $EnD(\tilde{H}_q^{(1)}, \tilde{C}_l) < t$, t is the threshold, the cloud server stops the secure Hamming distance calculation and sets the current C_l as the category which the query image belongs to, and proceeds to retrieve the specific images in the C_l category using $\tilde{H}_q^{(2)}$.
- Second, after finding the category, the cloud server calculates the similarity between $\tilde{H}_q^{(2)}$ and $\tilde{H}_i^{(2)}$ with the followed formula.

$$\begin{aligned} EnS(\tilde{H}_i^{(2)}, \tilde{H}_q^{(2)}) &= \left(\frac{\tilde{H}_i^{(2)} \cdot M'_{IO} \cdot M'_U \cdot \tilde{H}_q^{(2)}}{\gamma^2}\right) \bmod p_1 \\ &= \left(\sum_{j=1}^{d_2} \rho_q h_{q,j}^{(2)} h_{i,j}^{(2)} - \frac{\rho_q}{2} \sum_{j=1}^{d_2} (h_{i,j}^{(2)})^2 + \sum_{j=1}^{d_2-1} \alpha_j \beta_j\right) \bmod p_1 \\ &= -\frac{\rho_q}{2}\left(||H_i^{(2)} - H_q^{(2)}||^2 - ||H_i^{(2)}||^2\right) + \sum_{j=1}^{d_2-1} \alpha_j \beta_j \end{aligned} \quad (11)$$

  $Ens(*,*)$ is the secure Euclidean distances. From the calculation result we can see that the inner product of the encrypted feature vectors between the query and stored image approximates to the Euclidean distance of their plaintexts. This allows the cloud server to rank the results. After the computation and ranking, the cloud server sends the $top-k$ most similar images back to the edge server, and then the edge server forwards these results to each search user separately.

DecImage$(m_i', sk_3) \rightarrow m_i$. When users get the search results, they use the pre-assigned symmetric decryption key to get the original plaintext image. In order to prevent the cloud server from learning the file access pattern, search user generates different random number on encrypting both $H_q^{(1)}$ and $H_q^{(2)}$ each time so that the linkability between search query and returned results can be broken.

E. Dynamic updating

In real-world applications, cloud servers need to support dynamic image updates, which include updating images in existing categories, adding new categories, and so on. The hierarchical index structure of the SHPIR supports flexible dynamic image updates. When adding new categories in the image database, the image owner needs to first run the Alg.1 to generate a new category hash code, and then use our SHPIR training model to get the segmented hash code for newly added images. The image owner uploads the image features to the edge server, and the edge server implements **GenIndex** and **EncImage** to add the new image category and protect their privacy. Finally, the edge server uploads the newly generated image categories and their indexes to the cloud server, which simply appends the new image categories to the hierarchical indexes. If the image owners add new images to the existing categories, they use the SHPIR training model to obtain the segmented hash codes and upload them to the cloud server via edge server, then the cloud server will add the new images to the category they belong to. The image updating algorithm is described in Alg.2.

Algorithm 2 Dynamic updating

Input: new images m_x
Output: image feature $f_x = (H_x^{(1)}||H_x^{(2)})$
Add new category
set $H_x^{(1)} =$ **GenHashCode()**;
Image updating
extract feature of m_x with SHPIR training model;
$f_x = (H_x^{(1)}||H_x^{(2)})$;
GenIndex: $\tilde{I}_x = (\tilde{H}_x^{(1)}||\tilde{H}_x^{(2)})$;
EncImage: $\{m_x'\}_{x=1}^N$;
Upload $\tilde{I}_x$, $\{m_x'\}_{x=1}^N$ to cloud server;
for $\tilde{H}_i^{(1)} \in$ index **do** //Cloud server runs update
  if secure Hamming$(\tilde{H}_x^{(1)}, \tilde{H}_i^{(1)}) < t$ **then**
    set $\tilde{H}_x^{(1)} = \tilde{H}_i^{(1)}$;
    add $\tilde{H}_x^{(2)}$ to category $\tilde{H}_i^{(1)}$;
  else
    append new category $\tilde{H}_x^{(1)}$ to the index;
    add $\tilde{H}_x^{(2)}$ to category $\tilde{H}_x^{(1)}$;
  end if
end for

The update cost evaluation of SHPIR consists of two experiments. One experiment is adding 50 images to each of the existing categories, we test the update cost of adding images to 10, 20, 50, 150, and 200 categories respectively. Another experiment is adding new categories with 50 images each, and we similarly test the update cost of adding 10, 20, 50, 150, and 200 categories. In these two experiments, we mainly test the index update time, because the image update time is related to network bandwidth and server processing power which are less related to the algorithm performance. The update cost of SHPIR is shown in Table IV.

TABLE IV
DYNAMIC UPDATING COST

Add new images to existing categories					
categories	10-cat	20-cat	50-cat	150-cat	200-cat
updating cost(s)	0.076	0.396	0.695	1.381	2.328
Add new categories and images					
categories	10-cat	20-cat	50-cat	150-cat	200-cat
updating cost(s)	0.111	0.431	0.717	1.398	2.333

VI. SECURITY ANALYSIS OF SHPIR

In this section, we prove that our SHPIR scheme meets the privacy-preserving requirements of image privacy, index and query privacy, and query unlinkability. In SHPIR, the ciphertext feature vectors used for indexing and querying consist of two hash codes that are encrypted with different keys and encryption algorithms. The first hash code is used to contain the image category information and the second hash code is used to contain the feature vector of each image. In this section, we analyze the security of these two hash codes separately.

A. Privacy guarantee on images

In SHPIR, image owner encrypts image sets with the traditional symmetric encryption algorithm (e.g., AES) and upload them to the cloud server. The image owner will not allow any unauthorized users(e.g. cloud, adversaries or others) to get the plaintext images, and will also not disclose any information about secret keys. The traditional symmetric key encryption is widely recognized for its safety, as long as the symmetric keys are not compromised, the ciphertext images will not be decrypted by the adversary. Therefore, under the security assumption of SHPIR, the privacy of the image content can be guaranteed from being accessed by unauthorized users.

B. Privacy guarantee on category hash codes

We prove that SHPIR can guarantee that the adversary cannot get the plaintext category hash code from the index and the queries.

Theorem 1: SHPIR can guarantee the privacy of category hash code in both index and queries, and it is resilient to *known ciphertext attack*.

Proof. Security of category hash code in the index. The category hash code C is first transformed to $\hat{C}$ and then encrypted with $sk_1 = \{V, R, \pi\}$. Since $V = \{v_1, v_2, \ldots, v_{d_1}\}$, $R = \{r_1, r_2, \ldots, r_{d_1}\}$, $v_j \in [-p_1, p_1]$, $r_j \in [-p_2, p_2]$, $p_1 >> p_2$, π are randomly selected by the image owner. The information of category hash code C can be protected by the random number and the difficulty of factoring very large numbers.

Security of category hash label in queries. The category hash

code in queries are first converted to $\hat{H^{(1)}}_q$ and encrypted in the same way as $\hat{C}$. However, in the encryption of each query vector, the search user choses a different random vector U_q. Therefore, the same feature vector $H_q^{(1)}$ have different values in each query and it is not feasible for the cloud server or adversary to obtain the plaintext of $H_q^{(1)}$ from the encrypted query.

In summary, the privacy of feature vectors in index and queries can be protected under the *known ciphertext attack model* is proved.

C. Privacy guarantee on feature vectors

In SHPIR, each image is represented with its category code and feature vectors, which are encrypted with random vectors and LWE-based secure kNN, respectively. We first define the LWE problem [41], then give the definition of linear analysis attack, finally we prove the security of category hash code and feature vectors under the known ciphertext attack model and the known background attack model.

Definition 3 (LWE problem): Given polynomial many samples of $(\vec{a_i} \in \mathbb{Z}_{p_1}^{d_2}, b_i \in \mathbb{Z}_{p_1})$

$$b_i = H^{(2)} \times \vec{a_i}^{\top} + \epsilon \tag{12}$$

$\epsilon \in \mathbb{Z}_{p_1}$ is the error term drawn from some probability distribution. Hence, it is computational infeasible to recover the vector $H^{(2)}$ with non-negligible probability.

Theorem 2: SHPIR can guarantee the security of the feature vectors in index and queries under the *known ciphertext attack model*.

Proof. In SHPIR, the feature vectors are first extended to $2d_2$ dimensions and then encrypted to $\tilde{H}^{(2)}$.

$$\begin{aligned} \tilde{H}_i^{(2)} &= (\gamma \cdot \hat{H}_i^{(2)} + \zeta_i) \cdot M_{IO} \\ &= \gamma \cdot \hat{H}_i^{(2)} \cdot M_{IO} + \zeta_i \cdot M_{IO} \end{aligned} \tag{13}$$

As shown in Eq.13, recovering $\hat{H}_i^{(2)}$ from $\tilde{H}_i^{(2)}$ is the LWE problem in Definition 3, so it is considered a hard problem and computational infeasible. Moreover, M_{IO} in our scheme is a secret key generated and stored by the edge server, which is not available to the adversary. Thus, recovering of $\hat{H}_i^{(2)}$ in SHPIR is more difficult than the LWE problem. In our scheme, the feature vectors in index and in queries are encrypted with the same algorithm, so that recovering plaintext features from their corresponding ciphertexts is computationally infeasible for the polynomial-time adversary. In summary, the privacy of feature vectors in index and queries can be protected under the *known ciphertext attack model* is proved.

Theorem 3: SHPIR can guarantee the security of the category hash code and feature vectors in index and queries under the *known background attack model*.

Proof. In the known background attack model, besides the ciphertexts, the adversary also has access to some background information and analysis. Even though the adversary cannot directly recover the category hash code and feature vectors from the index and queries, it may perform linear analysis attack over the Hamming distance and inner product calculation results using the background information.

Suppose the adversary generates a query vector Q, and knows two comparison results of the query and any two clustering centers, $(\tilde{C}_m, \tilde{C}_n)$ in the index. From Eq.10, the adversary can get the equation as

$$\begin{aligned} D_{\tilde{C}_{m,j}} &= EnD(\tilde{C}_m, Q) - EnD(\tilde{C}_n, Q) \\ &= EnD(\tilde{H}_q^{(1)}, \tilde{C}_m) - EnD(\tilde{H}_q^{(1)}, \tilde{C}_n) \\ &= \sum_{j=1}^{d_1} \{[(\tilde{c}_{m,j}) - \tilde{c}_{n,j}) \bmod w_j] \oplus 0\} \end{aligned}$$

From this equation, the adversary knows the query vector Q, so there are $3d_1$ unknowns from $\tilde{C}_m, \tilde{C}_n$ and w_j. The linear analysis attack requires the adversary to obtain more than $3d_1$ plaintext vectors to solve the unknown vectors C_m and C_n, which makes it computational infeasible.

For the feature vectors of each image, from Eq.11, the adversary can get the equation as

$$\begin{aligned} D_{\tilde{H}_{i,j}^{(2)}} &= EnS(\tilde{H}_i^{(2)}, Q) - EnS(\tilde{H}_j^{(2)}, Q) \\ &= \frac{\rho_x}{2}(||H_i^{(2)} - H^{(2)}q||^2 - ||H_j^{(2)} - H_q^{(2)}||^2) \end{aligned}$$

In this equation, the adversary knows the query vector Q, so there are $2d_2$+1 unknowns from $H_i^{(2)}, H_j^{(2)}$ and ρ_x. The linear analysis attack requires the adversary to obtain more than $2d_2$ query vectors to solve the unknown vectors $(H_i^{(2)}, H_j^{(2)})$ in the index. However, in Eq.8, there is a random number ρ for each vector and ρ is different for each query vector, so that the known background attack is not feasible in SHPIR.

Since the adversary knows the values of $D_{\tilde{H}_{i,j}^{(2)}}$, it can derive the following equation:

$$\frac{D_{\tilde{H}_{i,j}^{(2)}}}{\rho_x} = \frac{1}{2}(||H_i^{(2)} - H_q^{(2)}||^2 - ||H_j^{(2)} - H_q^{(2)}||^2)$$

If the adversary has access to more than two comparison results, it can form a system of equations based on the above equation and solve for the unknowns $||H_i^{(2)} - H_q^{(2)}||^2$ and $||H_j^{(2)} - H_q^{(2)}||^2$.

However, since the SHPIR scheme ensures that the inner products are homomorphically encrypted, the adversary cannot directly compute the values of $D_{H_{i,j}^{(2)}}$. Moreover, since the homomorphic encryption scheme used in SHPIR is semantically secure, the adversary cannot recover any information about the encrypted inner products.

In summary, the SHPIR scheme can guarantee the security of the category centers and feature vectors in the index and queries under the *known background attack model*.

D. Query unlinkability

In SHPIR, different trapdoors are generated for each query, even though the queried vectors may be the same. We prove that our scheme can break the linkability of queries and search items.

Theorem 4: In SHPIR, for two encrypted queries, it is difficult for the adversary to distinguish between a game in which they are generated from the same vector and a game in which they are generated from two different vectors.

Proof. Suppose $f_q = H_q^{(1)}||H_q^{(2)}$ is the feature vector extracted from a queried image m_q, and the two queries generated based on f_q are presented as $\tilde{f}_{q,1}, \tilde{f}_{q,2}$.

$$\begin{aligned}\tilde{f}_{q,1} &= \tilde{H}_{q,1}^{(1)}||\tilde{H}_{q,1}^{(2)} \\ &= \pi(U_{q,1} \cdot \hat{H}_q^{(1)} \cdot R)||\ M_U \cdot (\gamma \cdot \hat{H}_{q,1}^{(2)\top} + \zeta_{j_1}^{\top}) \\ \tilde{f}_{q,2} &= \tilde{H}_{q,2}^{(1)}||\tilde{H}_{q,2}^{(2)} \\ &= \pi(U_{q,2} \cdot \hat{H}_q^{(1)} \cdot R)||\ M_U \cdot (\gamma \cdot \hat{H}_{q,2}^{(2)\top} + \zeta_{j_2}^{\top})\end{aligned} \tag{14}$$

where

$$\begin{aligned}\hat{H}_{q,1}^{(2)} &= \{\rho_q h_{q,1}^{(2)}, \rho_q h_{q,2}^{(2)}, \ldots, \rho_q h_{q,d_2}^{(2)}, \rho_q, \beta_1, \beta_2, \ldots, \beta_{d_2-1}\} \\ \hat{H}_{q,2}^{(2)} &= \{\rho_q' h_{q,1}^{(2)}, \rho_q' h_{q,2}^{(2)}, \ldots, \rho_q' h_{q,d_2}^{(2)}, \rho_q', \beta_1', \beta_2', \ldots, \beta_{d_2-1}'\}\end{aligned} \tag{15}$$

Based on the threat model assumptions, the encryption scheme used in SHPIR is semantically secure, which means that given the encrypted data, an adversary cannot learn anything about the plaintext. Then, for the given queries $\tilde{f}_{q,1}$ and $\tilde{f}_{q,2}$, since $U_{q,1} \neq U_{q,2}$, $\rho_q \neq \rho_q'$, from Eq.11, we can see that $\tilde{f}_{q,1} \neq \tilde{f}_{q,2}$. Consequently, the adversary cannot determine whether they were generated from the same query vector f_q. This proves the unlinkability of two encrypted vectors.

VII. Performance Evaluation

In this section, we will illustrate the performance of SHPIR from accuracy and efficiency.

A. Preparations

Experiment environment and dataset. To evaluate the performance of SHPIR, we implement our scheme on a PC with 3.6GHz Intel Core i7 CPU and 8GB memory running Windows 10. The programming language is Python, and the real-world image dataset is Caltech256[42], which consists of 256 categories. We choose 200 categories from Caltech256 for performance evaluation, each category has 80 images, the validation set and test set each contains 20 images in every category. Query images are from the test set.

Compared schemes. In order to illustrate the feasibility and efficiency of SHPIR, we replicated the code from four state-of-art and most relevant schemes, which are DVREI[3], CASHEIRS[17], SEI[20] and SVMIS[37], and compared SHPIR with these four schemes in terms of index construction efficiency and storage costs, retrieval efficiency, and retrieval accuracy. The compared four schemes use the pre-trained CNN model for image feature extraction. CASHEIRS uses random vectors for feature encryption and construct a hierarchical index tree. DVREI and SEI build the hierarchical index trees based on image features and use secure KNN to protect the feature vectors. SVMIS uses LWE-based secure kNN algorithm to protect feature vectors, and construct linear index of images.

Accuracy evaluation metric. We use Precision at top-k(P@k) as the performance metrics to evaluate the retrieval accuracy of SHPIR, DVREI, CASHEIRS, SEI and SVMIS. P@k is defined as follows:

$$\text{P@}k = \frac{correct\_num}{k} \tag{16}$$

where $correct\_num$ denotes the number of similar images in the top-k returned query results. Similar images here means that the returned images and the query images belong to the same category, and k is the total number of images in the returned retrieval results.

B. Dimensions of segmented hash codes

In our SHPIR scheme, the dimensionality of the segmented hash codes will affect the accuracy, search efficiency and storage cost of image retrieval. Higher dimensionality may improve the accuracy of image searching, but at the cost of higher computational complexity and index storage size. Therefore, we need to find the optimal dimensionality of the feature vector so as to find the optimal solution of vector dimensionality that can guarantee the high retrieval accuracy and efficiency with limited storage space. Table V and Table VI show the P@k values of the category hash codes and feature vectors in different dimensions when there are different numbers of top-k images are returned. Based on the experiment result, we chose a 24-dimensional vector as the category hash code $H^{(1)}$, the threshold Hamming distance $t = 4$, and 32-dimensional vector as the feature vector $H^{(2)}$.

TABLE V
Retrieval Accuracy of $H^{(1)}$ in Different Dimensions

dimensionality	k =5	k =10	k =20	k =30	k =40
24	0.841	0.841	0.841	0.841	0.841
32	0.842	0.842	0.842	0.842	0.842
64	0.850	0.850	0.850	0.850	0.850
128	0.852	0.852	0.852	0.852	0.852

TABLE VI
Retrieval Accuracy of $H^{(2)}$ in Different Dimensions

dimensionality	k =5	k =10	k =20	k =30	k =40
12	0.907	0.902	0.899	0.898	0.894
32	0.916	0.915	0.914	0.914	0.904
64	0.923	0.920	0.917	0.915	0.912
128	0.927	0.922	0.917	0.917	0.914

Increasing the dimensionality of feature vectors can improve the accuracy of retrieval; however, this is accompanied by an increase in extraction time due to the complexity of the network. This paper, however, benefits from the use of lower feature dimensions, fewer neurons and fewer parameters in the output layer of the network, resulting in greater extraction efficiency. We tested the feature extraction time of DVREI, CASHEIRS, SEI, SVMIS and our SHPIR scheme for a total of 4000 images from 50 categories. All three of CASHEIRS, SEI, SVMIS and DVREI used 128-dimensional feature vectors, requiring 65.55, 61.78, 59.1, 21.5 minutes, respectively, to extract features from 4000 images. In contrast, SHPIR only required 16.8 minutes due to its 24+44-dimensional feature vector. The experiment results show that SHPIR is more suitable for application in IoT systems.

C. Index building time and storage cost

The index building time is directly proportional to the dimension of the feature vector and the size of the data set, and is related to the complexity of the index structure. The index building time consists of 3 operations: (i) extracting image features with the pre-trained extraction model, (ii) encrypting features of trained images, and (iii) building the index structure. DVREI and CASHEIRS use K-Means to build their hierarchical index trees. SEI applies CAK-means[43] to construct the deeper hierarchical encrypted index tree. SVMIS builds the linear index of all images, its index building time mainly depends on the feature vector dimensions and the total number of images. These three schemes all use 128-dimensional features to construct the index, but in SHPIR we use the short segmented hash code as the image feature, and the index construction time mainly relies on K-Means clustering time. When constructing the index, we use the category hash codes $H^{(1)}$ as the K-Means initial input to stabilize the clustering result. The index generation time for different schemes is shown in Fig.4.

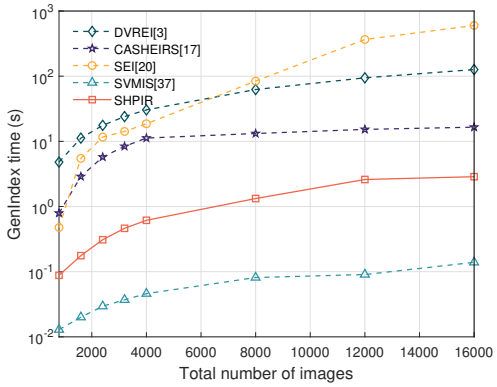


Fig. 4. Index generation time for different number of images

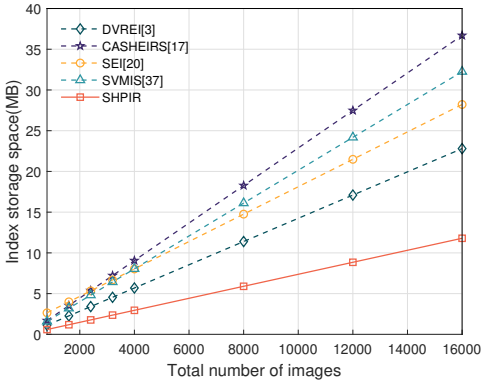


Fig. 5. Storage space cost for different number of images

As shown in Fig.4, the tree-based index structure needs more construction time compared to the linear one. SVMIS utilizes its linear structure and is thus constructed more quickly. SEI requires a significantly longer time for index building due to its use of affinity propagation to divide all images into K clusters, before using these K cluster centers as input for the K-Means tree-based index structure. Based on our experiment, the AP clustering time for 50, 100 and 150 categories are 15.36, 67.55 and 199.39 seconds, respectively.

We also compare the index storage cost of the SHPIR scheme to that of four other schemes, as demonstrated in Fig.5. The comparison does not include the storage space required for encrypted images as it is largely dependent on the encryption algorithm and the size of the images, and is independent of the retrieval algorithm. The storage overhead is mainly composed of the index structure and encrypted features, which increases with the dimensionality of the feature vector and the total number of images. The other three schemes use a 128-dimensional feature vector, while DVREI, SVMIS and SEI extend the feature vector to 256 dimensions after applying the LWE-based encryption. In contrast, our SHPIR scheme uses LWE encryption to extend the 32-dimensional feature vector to 64 dimensions, with the 24-dimensional category hash code remaining unchanged, resulting in an overall number of features totaling 24 + 64 dimensions. Consequently, our SHPIR scheme requires less storage space on the cloud server.

D. Efficiency evaluation and comparison

To assess the time efficiency of retrieval in the ciphertext domain for each scheme, we conducted comparative experiments in terms of search time with varying numbers of returned k. To evaluate these schemes, we present the average search time at the top-k of 5 runs for 50, 100, 150, and 200 randomly selected categories from the Caltech256 dataset. In the experiment, we split the images of each category into two sets (80 images for building the hierarchical index and 20 images for query). In the query process, after receiving a query, SHPIR calculates the secure Hamming distance between $H_q^{(1)}$ in the query vector and C_l in the index. If the Hamming distance is lower than the predetermined threshold t, a linear search is executed in the category represented by C_l, and returns the k most similar images. In our experiment, the threshold is set $t = 4$. The search time for different schemes is shown in Table VII.

Due to the hierarchical index structure, DVREI, CASHEIRS, SEI and SHPIR schemes search by first finding the target categories and then performing the linear search within these categories, while SVMIS performs the linear search within the whole image set. In our experiment, the search time of DVREI, CASHEIRS, SEI and SHPIR includes 3 parts: 1) finding the target categories, 2) performing a linear search within the categories, and 3) verifying whether the returned k images belong to the same categories as the query ones. The retrieval time consumed in the first part is related to the index structure, the second part is related to the total number of images in the target categories, and the third part is proportional to the value of k.

E. Accuracy comparison

The accuracy of an image storage system's retrieval performance can be influenced by multiple factors. Crucial factors among these are the quality of extracted image features, the index structure, and search algorithms. While a linear

TABLE VII
SEARCH EFFICIENCY COMPARISON UNDER VARIOUS CATEGORIES

Categories	Schemes	Search time (ms)				
		k=5	k=10	k=20	k=30	k=40
50 categories	DVREI[3]	0.203	0.264	0.512	0.636	0.816
	CASHEIRS[17]	13.101	13.126	13.179	13.231	13.279
	SEI[20]	0.265	0.266	0.328	0.453	0.450
	SVMIS[37]	9.310	9.336	10.451	10.623	11.775
	SHPIR	**1.243**	**1.255**	**1.262**	**1.270**	**1.278**
100 categories	DVREI[3]	0.184	0.325	0.687	0.718	0.967
	CASHEIRS[17]	27.024	27.045	27.088	27.110	27.169
	SEI[20]	0.335	0.383	0.469	0.547	0.586
	SVMIS[37]	19.666	21.352	20.121	20.313	21.774
	SHPIR	**1.565**	**1.571**	**1.577**	**1.582**	**1.597**
150 categories	DVREI[3]	0.248	0.276	0.434	0.608	0.780
	CASHEIRS[17]	50.482	50.503	50.567	50.622	51.005
	SEI[20]	0.682	0.781	0.960	1.106	1.409
	SVMIS[37]	32.251	32.674	32.997	33.238	34.072
	SHPIR	**1.856**	**1.864**	**1.873**	**1.891**	**1.907**
200 categories	DVREI[3]	0.222	0.273	0.430	0.622	0.920
	CASHEIRS[17]	87.766	87.798	88.002	88.061	88.120
	SEI[20]	1.342	1.411	1.45	1.62	1.78
	SVMIS[37]	43.239	44.113	44.789	45.124	46.051
	SHPIR	**2.166**	**2.179**	**2.185**	**2.208**	**2.219**

index structure like SVMIS can yield more precise retrieval outcomes, it typically comes at the cost of reduced efficiency in searching. On the other hand, tree-based index structures like DVREI, SEI and CASHEIRS offer better efficiency but lower accuracy. To balance the search efficiency and accuracy, our approach in SHPIR employs a hierarchical search strategy. Specifically, the initial search step involves calculating the secure Hamming distance between the query feature $H_q^{(1)}$ and the indexed category centers C_l. If the secure Hamming distance is less than the threshold t, a linear search of $H_q^{(2)}$ and stored image features with LWE-based secure kNN is performed within the relevant category. We have evaluated the search precision of our proposed approach by comparing it with existing schemes, namely [3], [17], [37] and [20], which also utilize CNN models for feature extraction. Notably, schemes[3], [37] and [20] employ LWE-based encryption. To conduct our accuracy comparison experiments, we randomly selected 10, 20, 50, 100, and 200 categories from the Calth256 image set, each category containing 80 images. The results of our experiments are presented in Fig. 6, which shows the average P@k values for various values of k ranging from 1 to 40.

Based on the results of experiment comparison, it can be observed that as the number of categories increases, the retrieval accuracy decreases. Nevertheless, our proposed SHPIR model exhibits consistent retrieval accuracy with the increase of k. This can be attributed to the accurate image classification performance of our SHPIR training model, which enable images to be assigned to their corresponding categories with high probability and stability.

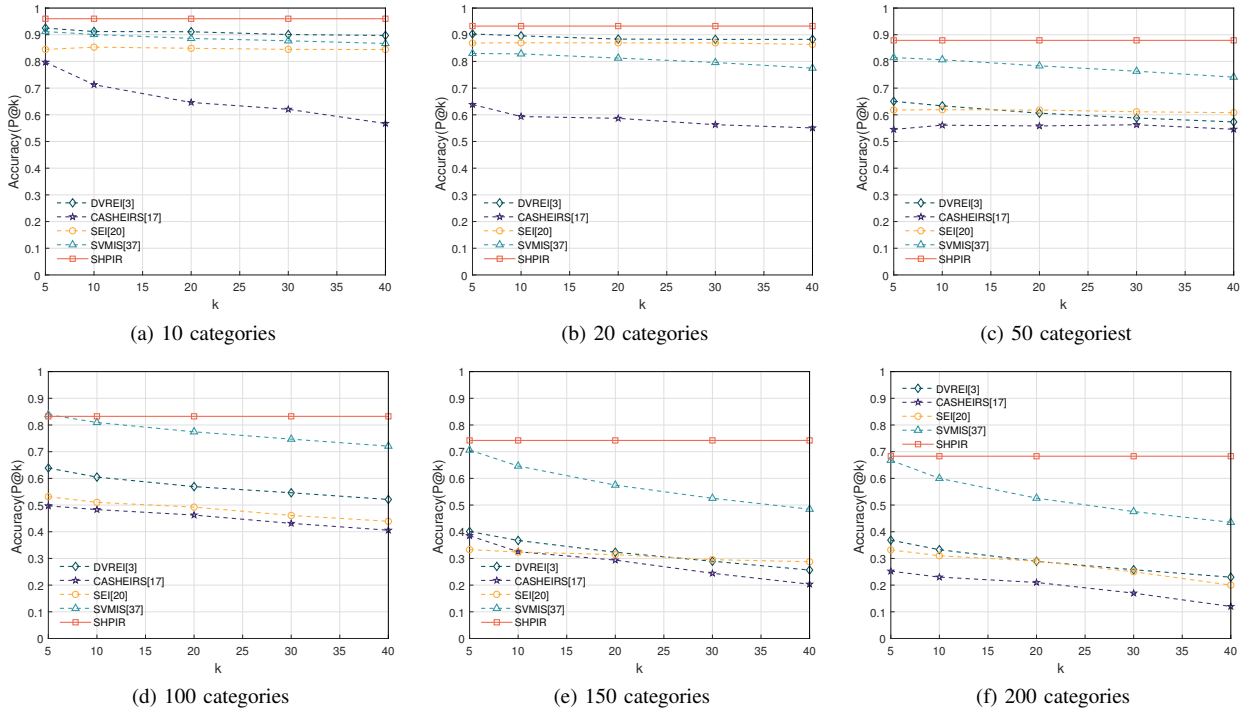
F. System performance evaluation

In the real-world IoT environment, resource constraints typically limit end IoT devices to data collection and transmission, with data processing delegated to servers. This necessitates the incorporation of edge servers into the system. The edge server represents the performance bottleneck, making the overall system performance highly dependent on the edge server's capabilities. Consequently, performance verification of the entire system focuses primarily on evaluating the edge server's performance. For the evaluation, we use our cellphone as the IoT end devices, a PC with a 2.5GHz Intel Core i5 CPU and 64GB memory running Windows 10 as the edge server, and Keras 2.11 is installed on the edge server for executing our feature extraction model. Communication between the cellphone and the edge server is facilitated via WiFi, with an upload speed of 35.51 Mbps and a download speed of 49.29 Mbps. Communication between the edge server and cloud server is facilitated via Internet, with an upload speed of 40.58 Mbps and a download speed of 70.36 Mbps.

The system performance evaluation primarily comprises two phases. The first phase is the system initialization, where the edge server extracts features from all images in the dataset, constructs hierarchical indexes, and encrypts the feature vectors and images. The second phase involves the image query process, in which the query image is transmitted from the cellphone to the edge server. The edge server then extracts the features, constructs the query trapdoor, and sends the trapdoor to the cloud server for retrieval.

System initialization. In this phase, the edge server initially employs the SHPIR feature extraction model to generate segmented hash codes for each image in the dataset. Subsequently, it executes the K-means clustering algorithm using $H^{(1)}$ as input to determine the category centers and construct the hierarchical index. Finally, the edge server encrypts both the index and the images before uploading them to the remote cloud server.

To enhance the efficiency of feature extraction, we first resize all input images to 256×256 pixels. Then we utilize four concurrent threads running the SHPIR model, with each thread processing a batch size of 32 images. The experimental results are presented in Table VIII. The feature extraction time (**ExtracFea**) is evaluated using the SHPIR model on 16000



(a) 10 categories (b) 20 categories (c) 50 categoriest

(d) 100 categories (e) 150 categories (f) 200 categories

Fig. 6. Accuracy comparison of returning the top 5-40 similar images across different schemes

TABLE VIII
SYSTEM INITIALIZATION PERFORMANCE EVALUATION

ExtracFea (s)	EncImage (s)	GenKey (ms)		GenIndex (s)					
		$H^{(1)}$	$H^{(2)}$	10-Cat	20-Cat	50-Cat	100-Cat	150-Cat	200-Cat
7.986	0.338	0.100	4.013	0.088	0.176	0.614	1.326	2.598	2.868

TABLE IX
IMAGE QUERY PERFORMANCE EVALUATION

ExtracFea (ms)	GenQuery (ms)	Search time (ms)					
		10-Cat	20-Cat	50-Cat	100-Cat	150-Cat	200-Cat
49.01	0.041	876.005	875.92	876.288	876.605	876.894	877.206

images in the Caltech256 dataset. The image encryption time (**EncImage**) is assessed using the AES encryption algorithm on these 16000 images, distributed across 200 categories, with a total size of 520MB. The key generation time (**GenKey**) includes the duration necessary for generating encryption keys for both $H^{(1)}$ and $H^{(2)}$. The index construction time (**GenIndex**) encompasses the time invested in constructing the index across a range of 10 to 200 categories, each comprising 80 images. This duration includes the time allocated for K-means clustering, the generation of the hierarchical index, as well as the encryption process for cluster centers and image features. Based on our experimental results, the generation time for the hierarchical index for 200 categories is negligible. The communication cost to upload the encrypted dataset, consisting of 16000 images and the encrypted hierarchical index, to the cloud server via the Internet is approximately 114.46 seconds.

Image query. In this phase, user sends the query image to edge server, then edge server extracts image feature vectors with the pre-trained SHPIR model and generates the query trapdoor. Subsequently, the edge server transmits the query trapdoor to the cloud server for image retrieval. The experimental results for image querying are presented in Table IX. This experiment evaluates the system's performance when querying a single image. The feature extraction time (**ExtracFea**) for a single image is determined by extracting features from 1000 images and calculating the mean duration required for feature extraction per image. The query generation time (**GenQuery**) represents the average time required to generate a query trapdoor for a single image, calculated by averaging the time taken to generate query trapdoors for 1000 images. The experiments also assessed the total time required to query an image across 10 to 200 categories, returning the top 5 most similar images. This search process includes the time needed to upload the image from the cellphone to the edge server, the time for the edge server to extract features and generate the trapdoor, and the time for the edge server to upload the query trapdoor to the cloud server and for the cloud server to perform the search. In the experiment, the photo taken from the cellphone has a resolution of 3072×4096 pixels

and a size of 3.5MB. The upload time from the cellphone to the edge server, approximately 826 milliseconds, constitutes the most prolonged segment of the entire querying process. Conversely, the time required to upload the query trapdoor to the cloud server is negligible, given that the query trapdoor is less than 1KB in size. The time expended for search and similarity comparison on the cloud server is approximately 1 millisecond.

The experimental data demonstrate that SHPIR can construct an index in 3 seconds for a dataset comprising 16000 images across 200 categories. While index construction typically occurs during the initial stage of the system setup, subsequent processes such as image feature extraction, encryption, and query trapdoor generation take only milliseconds. These results indicate that SHPIR is highly suitable for IoT environments characterized by high data collection volumes. In the image query process, the most time-consuming component, as observed in the experimental tests, is the transmission of the image from the mobile device to the edge server. In practical IoT applications, query efficiency can be enhanced by accelerating transmission speed, which can be achieved by reducing the resolution of the query image.

VIII. Conclusion

This paper presents a novel encrypted image retrieval scheme, SHPIR, designed to facilitate accurate and efficient search and retrieval of large-scale images within IoT network applications. The proposed image feature extraction model simultaneously acquires feature vectors and image categories, achieving accurate feature representation and image classification with low-dimensional vectors, thereby enhancing feasibility for the resource-constrained IoT devices. By constructing a hierarchical retrieval structure, SHPIR enhances retrieval efficiency while maintaining high retrieval accuracy, making it suitable for large-scale image database retrieval. Formal security analysis verifies that SHPIR protects the privacy of images, feature vectors, and file access patterns. Comparative experiments demonstrate the effectiveness and accuracy of the proposed scheme outperforms other state-of-the-art studies. To further assess the practical effectiveness and applicability of SHPIR, a real IoT environment is constructed to evaluate system performance in terms of initialization and image retrieval. The experiments demonstrate that SHPIR is a secure and viable solution for practical image retrieval applications in IoT environment. Future work may extend this approach to other media types, such as video and audio, to further enhance retrieval efficiency and accuracy.

References

[1] Z. Xia, L. Jiang, X. Ma, W. Yang, P. Ji, and N. N. Xiong, "A privacy-preserving outsourcing scheme for image local binary pattern in secure industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 629–638, 2020.

[2] J. Hao, W. Tang, C. Huang, J. Liu, H. Wang, and M. Xian, "Secure data sharing with flexible user access privilege update in cloud-assisted iomt," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 933–947, 2022.

[3] Y. Li, J. Ma, Y. Miao, H. Li, Q. Yan, Y. Wang, X. Liu, and K.-K. R. Choo, "Dvrei: Dynamic verifiable retrieval over encrypted images," *IEEE Transactions on Computers*, vol. 71, no. 8, pp. 1755–1769, 2022.

[4] S. K. Panda, M. Lin, and T. Zhou, "Energy efficient computation offloading with dvfs using deep reinforcement learning for time-critical iot applications in edge computing," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[5] L. Ahmad, "A survey on ddos attacks in edge servers," Master's thesis, University of Taxes at Arlington, Arlington.

[6] I. S. Community, "Cost of a data breach report," 2022.

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

[8] Y. Bai, L. Zhuo, B. Cheng, and Y. F. Peng, "Surf feature extraction in encrypted domain," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, 2014, pp. 1–6.

[9] R. Bellafqira, G. Coatrieux, D. Bouslimi, and G. Quellec, "Content-based image retrieval in homomorphic encryption domain," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 2944–2947.

[10] B. Ferreira, J. Rodrigues, J. Leitao, and H. Domingos, "Privacy-preserving content-based image retrieval in the cloud," in *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, 2015, pp. 11–20.

[11] Z. Xia, L. Jiang, D. Liu, L. Lu, and B. Jeon, "Boew: A content-based image retrieval scheme using bag-of-encrypted-words in cloud computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 202–214, 2022.

[12] Z. Xia, L. Wang, J. Tang, N. N. Xiong, and J. Weng, "A privacy-preserving image retrieval scheme using secure local binary pattern in cloud computing," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 318–330, 2021.

[13] M. Kitayama and H. Kiya, "Hog feature extraction from encrypted images for privacy-preserving machine learning," in *2019 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, 2019, pp. 80–82.

[14] W. Lu, A. Swaminathan, A. Varna, and M. Wu, "Enabling search over encrypted multimedia databases," *SPIE/IS&T Media Forensics and Security*, pp. 7254 – 18, 2009.

[15] J. Yuan, S. Yu, and L. Guo, "Seisa: Secure and efficient encrypted image search with access control," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 2083–2091.

[16] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Trans-*

actions on Information Forensics and Security, vol. 11, no. 11, pp. 2594–2608, 2016.

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

[18] 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 Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5348–5359, 2021.

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

[20] Y. Li, J. Ma, Y. Miao, Y. Wang, X. Liu, and K.-K. R. Choo, "Similarity search for encrypted images in secure cloud computing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 1142–1155, 2022.

[21] M. Shen, Y. Deng, L. Zhu, X. Du, and N. Guizani, "Privacy-preserving image retrieval for medical iot systems: A blockchain-based approach," *IEEE Network*, vol. 33, no. 5, pp. 27–33, 2019.

[22] M. A. A. Sibahee, S. Lu, Z. A. Abduljabbar, A. Ibrahim, Z. A. Hussien, K. A.-A. Mutlaq, and M. A. Hussain, "Efficient encrypted image retrieval in iot-cloud with multi-user authentication," *International Journal of Distributed Sensor Networks*, vol. 14, 2018.

[23] Y. Wang, F. Liu, Z. Pang, A. Hassan, and W. Lu, "Privacy-preserving content-based image retrieval for mobile computing," *Journal of Information Security and Applications*, vol. 49, pp. 2214–2126, 2019.

[24] Y. Ma, X. Chai, Z. Gan, and Y. Zhang, "Privacy-preserving tpe-based jpeg image retrieval in cloud-assisted internet of things," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 4842–4856, 2024.

[25] L. Song, Y. Miao, J. Weng, K.-K. R. Choo, X. Liu, and R. H. Deng, "Privacy-preserving threshold-based image retrieval in cloud-assisted internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 598–13 611, 2022.

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

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

[28] J. Li, Y. Wu, J. Zhao, and K. Lu, "Multi-manifold sparse graph embedding for multi-modal image classification," *Neurocomputing*, vol. 173, pp. 501–510, 2016.

[29] X. C. J. S. Lei Zhu, Zi Huang and H. T. Shen, "Exploring consistent preferences: Discrete hashing with pair-exemplar for scalable landmark search," in *Proceedings of the 25th ACM international conference on Multimedia*, October 2017, pp. 726–734.

[30] L. Zhu, Z. Huang, Z. Li, L. Xie, and H. T. Shen, "Exploring auxiliary context: Discrete semantic transfer hashing for scalable image retrieval," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5264–5276, 2018.

[31] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1224–1244, 2018.

[32] J. Yuan and Y. Tian, "Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 568–579, 2019.

[33] D. Zhu, H. Zhu, X. Wang, R. Lu, and D. Feng, "An accurate and privacy-preserving retrieval scheme over outsourced medical images," *IEEE Transactions on Services Computing*, pp. 1–1, 2022.

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

[35] F. Song, Z. Qin, J. Zhang, D. Liu, J. Liang, and X. S. Shen, "Efficient and privacy-preserving outsourced image retrieval in public clouds," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[36] C. Guo, S. Su, K.-K. R. Choo, and X. Tang, "A fast nearest neighbor search scheme over outsourced encrypted medical images," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 514–523, 2021.

[37] 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 Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5348–5359, 2021.

[38] J.-S. Li, I.-H. Liu, C.-J. Tsai, Z.-Y. Su, C.-F. Li, and C.-G. Liu, "Secure content-based image retrieval in the cloud with key confidentiality," *IEEE Access*, vol. 8, pp. 114 940–114 952, 2020.

[39] S. Guo, J. Xu, C. Zhang, C. Xu, and T. Xiang, "Imageproof: Enabling authentication for large-scale image retrieval," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 1070–1081.

[40] J. Qin, H. Li, X. Xiang, Y. Tan, W. Pan, W. Ma, and N. N. Xiong, "An encrypted image retrieval method based on harris corner optimization and lsh in cloud computing," *IEEE Access*, vol. 7, pp. 24 626–24 633, 2019.

[41] J. Yuan and Y. Tian, "Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 568–579, 2019.

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

[43] Y. Zhu, J. Yu, and C. Jia, "2009 ninth international conference on hybrid intelligent systems," vol. 1, 2009, pp. 338–343.