

Efficient Privacy-Preserving Similarity Retrieval With Fine-Grained Access Control

Yingying Li¹, Member, IEEE, Feng Li², Fuqun Wang³, Zhiquan Liu⁴, Yan Zhu⁵,
Qi Xie⁶, and Song Han⁷, Member, IEEE

Abstract—Privacy-preserving similarity retrieval for ciphertext images has broad applications in Internet of Things (IoT) areas, including smart healthcare, face recognition, and social networking. However, most existing privacy-preserving schemes suffer from inefficient retrieval and limited security guarantees due to the use of unreasonable encryption methods. In addition, those supporting fine-grained access control often lack scalability or are computationally inefficient. To address these challenges, we propose an efficient similarity retrieval scheme for ciphertext images that ensures both privacy preservation and fine-grained access control. First, we construct an encrypted index tree using clustering to improve retrieval efficiency while preserving high recall. Second, we achieve security against chosen-plaintext attacks (CPAs) and result verification by employing symmetric homomorphic encryption and Merkle hash tree. Third, we realize access control for each image, enabling simultaneous access verification and similarity retrieval via a single inner product operation. Our theoretical and experimental analysis shows that the proposed scheme is CPA-secure and achieves up to 100× faster query processing than existing CPA-secure schemes, with the ability to retrieve 2000 ciphertext images within 0.5 s for 128-dimensional feature vectors.

Index Terms—Access control, homomorphic encryption (HE), privacy-preserving similarity retrieval, result verification.

I. INTRODUCTION

WITH the rapid development of Internet of Things (IoT) technology and the proliferation of terminal devices, vast amounts of image data are being generated. A wide range

Received 30 April 2025; revised 14 June 2025; accepted 7 July 2025. Date of publication 15 July 2025; date of current version 25 September 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62402155 and Grant 62072403; in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LQ24F020013; and in part by the Hangzhou Joint Fund of the Zhejiang Provincial Natural Science Foundation of China under Grant LHZSZ24F020002. (Corresponding author: Feng Li.)

Yingying Li, Fuqun Wang, Yan Zhu, and Qi Xie are with the School of Mathematics, Hangzhou Normal University, Hangzhou 311121, China (e-mail: lyylyyingying@163.com; fqwang@hznu.edu.cn; zyooo058@163.com; qixie68@126.com).

Feng Li is with the School of Computing and Information Systems, Singapore Management University, Singapore 178902 (e-mail: fengli@smu.edu.sg).

Zhiquan Liu is with the College of Cyber Security, Jinan University, Guangzhou 510632, China (e-mail: zqliu@vip.qq.com).

Song Han is with the School of Computer and Computing Science, Hangzhou City University, Hangzhou 310015, China, also with the School of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou 310018, China, and also with the Zhejiang Panshine Information Technology Company Ltd., Hangzhou 311100, China (e-mail: hans@hzcu.edu.cn).

Digital Object Identifier 10.1109/IIOT.2025.3589237

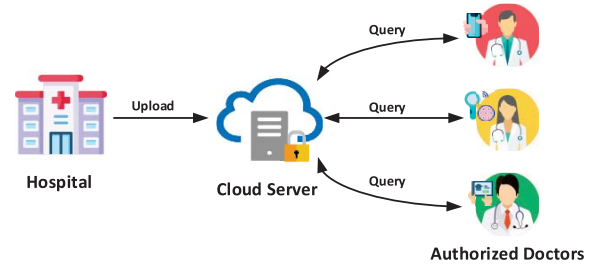


Fig. 1. IoT scenario for image similarity retrieval.

of image-based applications, such as smart healthcare, face recognition, and social networking, are now deployed in cloud-assisted IoT scenarios. Offloading computation to the cloud alleviates the burden on resource-constrained IoT devices, but also introduces new requirements for high efficiency, strong data security, and flexible functionality.

As illustrated in Fig. 1, a representative scenario arises in smart healthcare. The hospital collects medical images, encrypts them, and uploads them to the cloud server for secure storage and retrieval. Authorized doctors use their IoT devices to send similarity queries to the cloud, and the returned retrieval results assist in diagnosis and treatment planning. In this context, efficient retrieval ensures timely medical responses, strong security guarantees the confidentiality of sensitive patient data, and access control ensures that doctors from different departments can only access images to which they are authorized. Therefore, achieving efficiency, privacy preservation, and access control in image similarity retrieval is both a critical and urgent challenge in IoT-based applications.

Although many privacy-preserving image retrieval schemes have been proposed [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], they still face some limitations. First, current schemes fail to achieve a balance between efficiency and security. Many privacy-preserving similarity retrieval schemes [3], [4], [5], [8], [9], [10], [11] adopt the secure k -nearest neighbor (kNN) technique [21] to enable efficient similarity computation over encrypted features. The basic secure kNN scheme [21] protects data privacy by adding random numbers and applying random matrix transformations. However, as shown in [22], it is even vulnerable to ciphertext-only attacks (COAs). Enhanced secure kNN variants [3], [4], [5], [8], [9], [10], [11] have been developed by introducing additional randomization, and

have been proven to resist both COAs and known-background attacks (KBAs). Nevertheless, KBA security is insufficient to meet the strong security requirements of practical applications. Additionally, most secure kNN-based schemes are designed for ranked retrieval, which results in similarity order leakage and helps attackers infer correlations between query tokens [12], [13]. To strengthen privacy preservation, some schemes [14], [15], [16], [17], [18], [19], [20] have instead adopted homomorphic encryption (HE), such as the Paillier cryptosystem [23] to compute similarity. While these schemes provide semantic security, simple linear index structures and complex ciphertext operations result in expensive overhead during image retrieval.

Second, existing access control mechanisms in privacy-preserving image retrieval schemes are either inflexible or computationally expensive. In the polynomial-based access control mechanism [2], [4], [8], [9], access policy is represented as the roots of the polynomial, and the user's role is evaluated by calculating the polynomial's value. If the value is zero, the user's role satisfies the access policy, thereby granting access to this image. However, the degree of the polynomial increases super-linearly with the number of roles, which leads to high-computational overhead. In addition, the polynomial-based access control mechanism does not support wildcard roles, which limits its practical use in IoT settings. Another widely used approach for achieving fine-grained access control is ciphertext-policy attribute-based encryption (CP-ABE). In CP-ABE, the symmetric key used for image encryption is encrypted under an access policy. Only users whose attributes satisfy the policy can recover the symmetric key and decrypt the corresponding image ciphertext [18]. However, CP-ABE is generally less efficient than polynomial-based access control mechanisms due to the computational overhead of bilinear pairing operations.

In light of the above observations, existing schemes are either efficient but provide limited security, or highly secure but suffer from low efficiency. Moreover, access control mechanisms in these schemes either do not support wildcard attributes or incur significant performance overhead, which limits their practical deployment in IoT environments. Motivated by these limitations, we propose PPSR, an efficient and privacy-preserving image similarity retrieval scheme with fine-grained access control. The main contributions of this work are summarized as follows.

- 1) *Efficient Retrieval*: PPSR is the first to combine clustering with HE to construct a two-layer encrypted index tree, which significantly reduces retrieval overhead. It achieves query processing speeds up to 100× faster than existing schemes that provide comparable security guarantees.
- 2) *High Security*: PPSR realizes threshold retrieval over encrypted images based on the homomorphic ciphertext symbol computation protocol. It returns only results that meet the similarity threshold and access control requirements, avoiding similarity ranking and thus preventing order leakage. Compared to KBA-secure schemes based on secure kNN, PPSR has stronger security and can resist chosen-plaintext attack (CPA). Additionally, it

designs a simplified verification mechanism to ensure the correctness of retrieval results.

- 3) *Fine-Grained Access Control*: PPSR supports wildcard attributes and enforces access control at the individual image level rather than at the broader category level, enabling more flexible access control mechanism. It embeds both access policies and user attributes into image feature vectors, enabling efficient and simultaneous retrieval and access control. Access is denied when the inner product of the corresponding vectors exceeds zero.

The remainder of this article is organized as follows. Section II reviews the related work on retrieval. Section III introduces the preliminaries used in our scheme, and Section IV presents the system model, threat model and design goals. Section V provides the main idea and the details of our scheme. The theoretical analysis and performance evaluation are presented in Sections VI and VII, respectively. Finally, Section VIII concludes this article.

II. RELATED WORK

In this section, we briefly review some existing related work on ciphertext images similarity retrieval.

Retrieval Based on Secure kNN: Since the first ciphertext images retrieval scheme was proposed by Lu et al. [1], researchers have proposed various schemes to improve efficiency and security. Yuan et al. [2] constructed an index tree based on the K -means clustering and encrypted feature vectors with the secure kNN, which achieves high retrieval efficiency. Building upon this, Li et al. [3] extended the approach by incorporating a dynamic verification tree based on the chameleon hash, which supports dynamic updates and results verification. More recently, Mao et al. [4] proposed a verifiable encrypted image retrieval scheme based on the chameleon hash tree. To further speed up the retrieval process, Li et al. [5] redesigned the hierarchical index tree by merging several similar clusters into one cluster. The adapted index tree has fewer levels. In addition to the index tree, building an index with hash function is also a common method. In [6], [7], Xia et al. constructed prefilter tables using locality-sensitive hashing (LSH) and encrypted the Multimedia Content Description Interface (MPEG-7) features in the tables using secure kNN. Moreover, they proposed a watermark-based protocol to trace the illegal query user who distributed the retrieved images in [6]. Similarly, scheme [8] implemented user tracking through watermarking technology. In addition, a lightweight encrypted image retrieval with access control based on polynomial-based access policy and secure kNN was proposed in [8]. Considering retrieval in multiowner multiuser settings, Tong et al. [9] employed the polynomial-based access policy and secure kNN, while Khan et al. [10] employed attribute-based encryption and secure kNN. In [11], secure kNN and watermarking are used to achieve secure and traceable image retrieval. These schemes have been proven to resist COA and KBA by their authors. In the KBA model, the attacker is assumed to have access to a limited amount of

plaintext, which is insufficient to construct a linearly independent system of equations for breaking the ciphertext. However, KBA security is not sufficient for practical applications. In addition, these ranked search schemes rely on ranking similarity scores to produce retrieval results, which can leak similarity order information and help attackers infer correlations between query tokens. To enhance security, two threshold retrieval schemes that introduce more random numbers were proposed in [12] and [13], and have been proven to be CPA-secure.

Retrieval Based on HE: Compared with secure kNN, HE performs better in terms of security. Zhang et al. [14] proposed a secure image retrieval scheme, which encrypts the feature vectors using the Paillier HE algorithm. Li et al. [15] designed a new data structure subsimhash and then constructed an inverted index to achieve a secure dynamic threshold retrieval scheme. However, this scheme revealed the Hamming distance and threshold to the cloud server. Instead of encrypting feature vectors, Guo et al. [16] encrypted the mean and standard deviation of feature vector using Paillier HE to compute a lower bound of Euclidean distance, thereby significantly reducing the computational cost of Euclidean distance. Based on [16], Yang et al. [17] proposed an encrypted image retrieval in multiuser settings through encrypting the lower bound on the squared Euclidean distance with the distributed two trapdoors public-key cryptosystem [24]. In addition to Paillier HE algorithm, Zhang et al. [18] employed a multilevel HE algorithm [25] that supports both homomorphic computation and key conversion, enabling a privacy-preserving image search system with access control in multiuser settings. With the Cheon-Kim-Kim-Song (CKKS) and private information retrieval (PIR), Chen et al. [19] proposed a secure content-based image retrieval scheme in cloud-assisted IoT, which achieves indistinguishability against CPA. Li et al. [20] designed an additional homomorphic bitmap index to improve search efficiency and accuracy and employed a reversible data hiding technique to protect the privacy of verification information. Although the HE techniques can protect data privacy while supporting ciphertext computation, their computational and communication overhead is still high.

To improve efficiency and security, we will propose an efficient privacy-preserving similarity retrieval scheme with fine-grained access control for ciphertext images. Compared with previous schemes, our scheme has the following advantages shown in Table I.

III. PRELIMINARIES

In this section, we will introduce some related preliminaries, including symmetric HE (SHE), sign computation protocol and Merkle hash tree (MHT).

A. Symmetric Homomorphic Encryption

The SHE is more lightweight than the traditional public key HE Paillier [23], and has been proved to be semantically secure in [26] and [27]. It can support leveled homomorphic addition and multiplication. Specifically, the SHE consists of three steps, namely, key generation, message encryption, and ciphertext decryption, which are defined as follows.

TABLE I
COMPARISON OF SIMILARITY RETRIEVAL SCHEMES

Schemes	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$	$\mathbb{F}_6$
[2]	Tree	SkNN	Rank	KBA	Polynomial*	✗
[3]	Tree	SkNN	Rank	KBA	✗	CHT
[4]	Linear	SkNN	Rank	KBA	Polynomial*	CHT
[8]	Linear	SkNN	Rank	KBA	Polynomial*	✗
[9]	Linear	SkNN	Rank	KBA	Polynomial*	HomMAC
[10]	Tree	SkNN	Rank	KBA	CP-ABE	MHT
[12]	Linear	Matrix	Threshold	CPA	✗	MHT
[13]	Linear	Matrix	Threshold	CPA	✗	✗
[18]	Linear	HE	Rank	CPA	CP-ABE	✗
[20]	Bitmap	HE	Rank	CPA	✗	✗
PPSR	Tree	HE	Threshold	CPA	Polynomial**	MHT

- $\mathbb{F}_1$: Index structure; $\mathbb{F}_2$: Index encryption method; $\mathbb{F}_3$: Retrieval type; $\mathbb{F}_4$: The strongest attack model that the scheme claims to resist; $\mathbb{F}_5$: Access control method; $\mathbb{F}_6$: Verification method.
- SkNN: Secure kNN; CHT: Chameleon hash tree; MHT: Merkle hash tree; HomMAC: Homomorphic MAC.
- The difference between Polynomial* and Polynomial** is that Polynomial** additionally implements wildcard attribute matching.

- 1) $(PP, SK) \leftarrow \text{SHE.KeyGen}(k_0, k_1, k_2)$: Given the security parameters (k_0, k_1, k_2) with $k_1 < k_2 < (k_0/2)$, generate the secret key $SK = (p, L)$, where p, q are two large prime numbers with $|p| = |q| = k_0$ and L is a random number with $|L| = k_2$. Then, compute $N = pq$ and set the public parameter $PP = (k_0, k_1, k_2, N)$, and the basic message space $M = \{m | m \in [-2^{k_1-1}, 2^{k_1-1}]\}$.
- 2) $E(m) \leftarrow \text{SHE.Enc}(m, PP, SK)$: Encrypt a plaintext message $m \in M$ with the secret key $SK = (p, L)$ as

$$E(m) = (rL + m)(1 + r'p) \bmod N \quad (1)$$

where r and r' are two random numbers with the bit length $|r| = k_2$, $|r'| = k_0$.

- 3) $m \leftarrow \text{SHE.Dec}(E(m), SK)$: Decrypt a ciphertext $E(m)$ with the secret key $SK = (p, L)$ as

$$m' = (E(m) \bmod p) \bmod L. \quad (2)$$

If $m' < (L/2)$, it means that $m \geq 0$ and $m = m'$, otherwise $m < 0$ and $m = m' - L$.

The SHE method has the following homomorphic addition and multiplication properties.

- 1) *Homomorphic Addition-I*: $E(m_1) + E(m_2) \rightarrow E(m_1 + m_2)$.
- 2) *Homomorphic Multiplication-I*: $E(m_1) * E(m_2) \rightarrow E(m_1 * m_2)$.
- 3) *Homomorphic Addition-II*: $E(m_1) + m_2 \rightarrow E(m_1 + m_2)$.
- 4) *Homomorphic Multiplication-II*: when $m_2 > 0$, $E(m_1) * m_2 \rightarrow E(m_1 * m_2)$.

B. Sign Computation Protocol

We give a sign computation protocol [26] to determine whether a plaintext message $m \in M$ is greater than 0 based on its ciphertext. The sign of m is defined as

$$\text{sign}(m) = \begin{cases} 1, & m > 0 \\ -1, & m \leq 0. \end{cases} \quad (3)$$

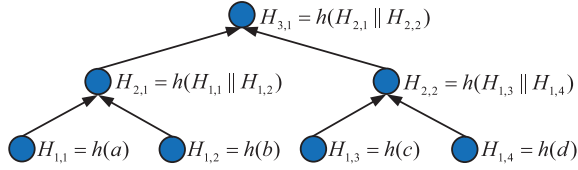


Fig. 2. Construction of MHT.

Given the ciphertext $E(m) \leftarrow \text{SHE.Enc}(m)$ of message m , two noncollusive cloud servers (i.e., CS_1 and CS_2 in our scheme) cooperate on computing the sign of m while keeping m secret from both CS_1 and CS_2 . CS_1 holds two ciphertexts $E(m)$ and $E(-1)$, and CS_2 holds a secret key SK . The sign computation protocol has the following steps.

- 1) *Step 1*: CS_1 selects two random numbers $r_1, r_2 \in \{0, 1\}^{k_1}$ where $r_1 > r_2 > 0$. Then, CS_1 computes

$$r_1 * E(m) + r_2 * E(-1) \bmod N \quad (4)$$

with the SHE's homomorphic properties, which equals to $E(r_1 * m - r_2)$. Finally, CS_1 sends $E(r_1 * m - r_2)$ to CS_2 .

- 2) *Step 2*: CS_2 first decrypts $E(r_1 * m - r_2)$ with SHE's decryption algorithm and secret key SK to recover $r_1 * m - r_2$. Then, CS_2 can deduce that the sign of m as

$$\text{sign}(m) = \begin{cases} 1, & r_1 * m - r_2 > 0 \\ -1, & r_1 * m - r_2 < 0. \end{cases} \quad (5)$$

Correctness: The sign computation protocol is correct iff $m > 0 \Leftrightarrow \text{sign}(m) = 1$ and $m \leq 0 \Leftrightarrow \text{sign}(m) = -1$. Since these two propositions are equivalent, we only prove $m > 0 \Leftrightarrow \text{sign}(m) = 1$ holds.

Proof: We prove the proposition $m > 0 \Leftrightarrow \text{sign}(m) = 1$ from sufficiency and necessity.

- 1) *Sufficiency*: Since $r_1 > r_2 > 0$, if $m > 0$, $r_1 * m - r_2 > 0$ holds. CS_2 gets $\text{sign}(m) = 1$.

- 2) *Necessity*: If CS_2 gets $\text{sign}(m) = 1$, $r_1 * m - r_2 > 0$ holds. Since $r_1 > r_2 > 0$, $m > (r_2/r_1) > 0$ holds. ■

C. Merkle Hash Tree

The MHT [28] is a complete binary tree, which can be used to verify whether the data is tampered with and forged, even deleted. In MHT, the hash values of leaf nodes are generated by hashing their data, while the hash values of nonleaf nodes are generated by hashing their child nodes. Specifically, as shown in Fig. 2, a, b, c, d are four data, h is a collision-resistant hash function, and “||” denotes the concatenation of two hash values. The hash values of four data are $H_{1,1} = h(a)$, $H_{1,2} = h(b)$, $H_{1,3} = h(c)$, $H_{1,4} = h(d)$. They are arranged in four leaf nodes, respectively. The hash values of intermediate nodes $H_{2,1}, H_{2,2}$ are computed by hashing the hash values of their child nodes, $H_{2,1} = h(H_{1,1} || H_{1,2})$, $H_{2,2} = h(H_{1,3} || H_{1,4})$. At last, the hash value of root node is $H_{3,1} = h(H_{2,1} || H_{2,2})$. Under the assumption that the hash function $h()$ is collision-resistant, these four data cannot be forged since the root node $H_{3,1}$ is unique.

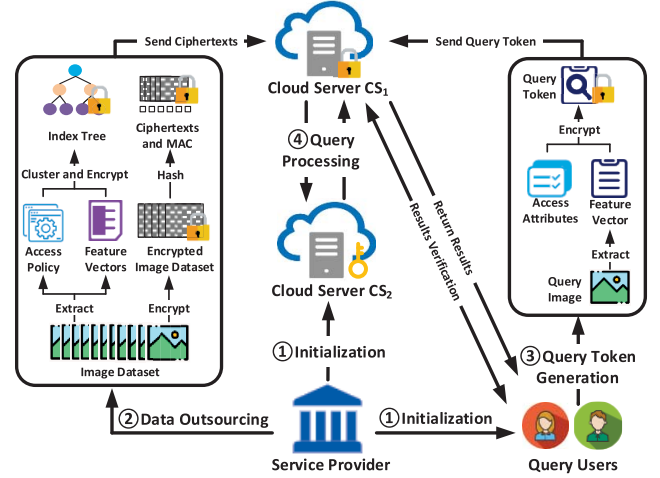


Fig. 3. System model of PPSR.

IV. PROBLEM FORMULATION

In this section, we present the system model, threat model and design goals.

A. System Model

In PPSR, we consider a secure ciphertext images retrieval scenario, which consists of four parties, i.e., a service provider SP, a set of query users $\text{QUs} = \{\text{QU}_1, \text{QU}_2, \dots\}$, and two cloud servers CS_1, CS_2 , as shown in Fig. 3.

- 1) *Service Provider SP*: SP has an image dataset with n images $\text{IMG} = \{\text{img}_1, \text{img}_2, \dots, \text{img}_n\}$. SP outsources IMG to CS_1 and delegates CS_1, CS_2 to provide similarity retrieval services to query users. Specifically, SP first initializes the system, and then generates and distributes the system keys (as shown in step ①). Next, SP encrypts the image dataset and integrates the access control vector into the image's feature vector, resulting in an extended vector stored in the encrypted index tree. In addition, SP computes the message authentication code (MAC) values based on ciphertext images for results verification. Finally, SP sends the encrypted image dataset, encrypted index tree, and MAC values to CS_1 (as shown in step ②).

- 2) *Query Users QUs*: When a QU issues a query to the cloud server, he/she extracts the feature vector of query image img_q and defines two thresholds for index. Then, QU generates the query token and sends it to CS_1 (as shown in step ③). The returned retrieval images satisfy both the similarity threshold and the access control policy. When QU verifies the correctness of retrieval results, he/she sends the challenging information to CS_1 for verification proof.

- 3) *Cloud Servers CS_1, CS_2* : CS_1 stores the encrypted dataset and index tree, while CS_2 stores the secret key used to encrypt the index tree. Once the query token is received, CS_1 and CS_2 cooperate to complete the retrieval task and return the retrieval results to query users (as shown in step ④). Besides, CS_1 answers the verification proof from QU.

B. Threat Model

In PPSR, SP is *honest* since he/she owns the image dataset. Moreover, QUs are assumed to be *honest*, meaning that they will honestly follow our scheme to issue reasonable queries to cloud servers and will not collude with either cloud servers or unauthorized query users. As for the cloud servers, CS₂ is assumed to be *honest-but-curious*, which will honestly follow our scheme to process queries. But CS₁ is considered to be *semi-honest-but-curious*, which is also applied in [29] and [30]. Different from *honest-but-curious* assumption, *semi-honest-but-curious* CS₁ may tamper with or forge retrieval results for the sake of its commercial interests or conceal data loss accidents. In addition, we assume that CS₁ and CS₂ will not collude. The noncollusion assumption is reasonable since cloud servers from different companies compete for interest. Note that our scheme may face other active attacks, such as side-channel attacks. Since we are mainly concerned with data privacy preservation, those attacks are beyond the scope of our work and will be discussed in future work.

C. Design Goals

We propose an efficient privacy-preserving similarity retrieval scheme for ciphertext images, which is required to achieve the following goals.

- 1) *Efficient Retrieval*: PPSR should support efficient ciphertext similarity retrieval, enabling users to quickly obtain relevant results.
- 2) *High Security*: PPSR should ensure the privacy of images and their feature vectors, as well as the correctness of retrieval results, preventing cloud servers from accessing plaintext or tampering with the results.
- 3) *Fine-Grained Access Control*: PPSR should allow an access control policy to be defined for each image, ensuring that only users who meet the policy can access the image.

V. OUR PROPOSED SCHEME

In this section, we first introduce the main idea of PPSR, then present its detailed constructions, and finally extend the scheme by enabling both service providers and query users to enforce access control policies, thereby achieving bilateral access control. Table II gives some important notations used in PPSR.

A. Main Idea of PPSR

The similarity between images is typically measured by the Euclidean distance between their feature vectors. We consider a plaintext similarity retrieval scenario, where SP holds an image dataset $IMG = \{img_1, img_2, \dots, img_n\}$ consisting of n images. For each image $img_i \in IMG$, SP extracts a CNN feature $\vec{f}_i$, resulting in a feature vector set $F = \{\vec{f}_1, \vec{f}_2, \dots, \vec{f}_n\}$. SP sends IMG (referred to as data images) and F (referred to as data vectors) to CS₁. When a QU wishes to retrieve images similar to a query image img_q , he/she first extracts its CNN feature $\vec{f}_q$ as the query vector and selects a threshold

TABLE II
NOTATIONS

Notations	Descriptions
IMG	the image dataset $\{img_1, img_2, \dots, img_n\}$
n	the number of images in IMG
k_{img}	the key of AES
$cimg_i, \vec{f}_i$	the ciphertext and feature vector of img_i
d	the length of feature vector $\vec{f}_i$
$\vec{a}_i, l$	the access control vector of img_i and its length
$\{C_t\}_{t=1}^K$	the clustering centers
K	the number of clustering centers
$\vec{c}_t$	the clustering center vector of C_t
$\vec{r}$	the root node value of index
$T, E(T)$	The index and its ciphertext
img_q	the query image of QU
$\vec{f}_q$	the feature vector of img_q
$\vec{v}$	the attribute vector of QU
τ_{leaf}, τ_{med}	the thresholds selected by QU
PP, SK	the public parameter and secret key of SHE
MAC_i	the MAC value of img_i
k_{MAC}	the key of MAC()
R	the identifiers of returned ciphertext images

$\tau > 0$. QU then submits $\vec{f}_q$ and τ to CS₁, which computes the Euclidean distance $\|\vec{f}_i - \vec{f}_q\|$ for each $i \in [1, n]$. If $\|\vec{f}_i - \vec{f}_q\| \leq \tau$ holds, the image img_i is considered a match. Finally, CS₁ returns all matched data images to QU.

To protect data privacy while enabling efficient computation in the ciphertext similarity retrieval scenario, we adopt a SHE algorithm to encrypt feature vectors. Each data vector $\vec{f}_i$ is expanded to $\vec{f}'_i = (\vec{f}_i, 1, \|\vec{f}_i\|^2)$, and the query vector $\vec{f}_q$ is expanded to $\vec{f}'_q = (-2\vec{f}_q, \|\vec{f}_q\|^2 - \tau^2, 1)$. The comparison between the Euclidean distance and the threshold is transformed into the computation of the inner product $\vec{f}'_i \cdot \vec{f}'_q$. This inner product is computed under encryption by leveraging the homomorphic properties and a secure sign computation protocol is used to determine the comparison result. To improve retrieval efficiency, we further process the feature vectors using a clustering algorithm and construct an index tree. During query processing, CS₁ and CS₂ first identify the clusters most similar to the query and then perform a refined search within those clusters. This approach avoids traversing the entire dataset and significantly improves efficiency. Meanwhile, the recall rate is enhanced by allowing QU to specify different thresholds, enabling more flexible and comprehensive retrieval.

To support fine-grained access control within the ciphertext similarity retrieval process, we integrate access policy evaluation directly into the vector space. Specifically, each data image img_i is associated with an l -dimensional access vector $\vec{a}_i = (a_{i,1}, a_{i,2}, \dots, a_{i,l})$ defined by SP, while each QU holds an attribute vector $\vec{v} = (v_1, v_2, \dots, v_l)$ representing his/her access privileges. To simultaneously perform access control and similarity computation, we extend the vectors as follows. $\vec{a}_i$ is expanded to $\vec{a}'_i = (\vec{a}_i, -2a_{i,1}^2, -2a_{i,2}^2, \dots, -2a_{i,l}^2, \sum_{t=1}^l a_{i,t}^3)$, resulting in the extended data vector $\vec{f}''_i = (\vec{f}'_i, \vec{a}'_i)$. Similarly, $\vec{v}$ is extended to $\vec{v}' = \gamma^2(v_1^2, v_2^2, \dots, v_l^2, \vec{v}, 1)$, and the query vector becomes $\vec{f}''_q = (\vec{f}'_q, \vec{v}')$, where γ is a random number greater than the threshold τ . As a result, the inner product $\vec{f}''_i \cdot \vec{f}''_q \leq 0$ holds

iff two conditions are simultaneously satisfied: 1) the feature vectors meet the similarity condition $\|\vec{f}_i - \vec{f}_q\|^2 \leq \tau^2$, and 2) the QU satisfies the access policy defined by SP. This enables access control and similarity retrieval to be performed simultaneously through a single inner product operation.

B. Detailed Constructions of PPSR

Our PPSR involves four entities, namely, SP, QU, CS₁, and CS₂, and consists of four phases: system initialization and data outsourcing on the side of SP, query token generation on the side of QU, and query processing on the sides of CS₁ and CS₂. The details are presented as follows.

1) *System Initialization*: In this phase, SP is responsible for initializing the system. First, SP selects three security parameters (k_0, k_1, k_2) . Then, SP runs $\text{SHE.KeyGen}()$ to generate a pair of public parameters and secret key as $(PP, SK) \leftarrow \text{SHE.KeyGen}(k_0, k_1, k_2)$, and runs $E(-1) \leftarrow \text{SHE.Enc}(-1)$ to generate the ciphertext of -1 . After that, SP generates a key k_{MAC} of MAC algorithm, a key k_{img} of advanced encryption standard (AES) algorithm, and a collision-resistant hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ where κ is the security parameter. At last, SP publishes $\{PP, E(-1), h\}$ and sends $\{SK, k_{\text{MAC}}, k_{\text{img}}\}$ to QUs and SK to CS₂.

2) *Data Outsourcing*: In this phase, SP outsources his/her dataset $\text{IMG} = \{\text{img}_1, \text{img}_2, \dots, \text{img}_n\}$ to CS₁. First, SP constructs an encrypted index tree, then encrypts the image dataset and generates the MAC values, and finally sends them to CS₁.

1) *Build the Encrypted Index*: As shown in Algorithm 1, SP first extracts the feature vectors $\{\vec{f}_1, \vec{f}_2, \dots, \vec{f}_n\}$ using the pretrained CNN model for IMG, and then clusters all features by using clustering method (e.g., K-means). Suppose the dataset are clustered into K clusters $\{C_1, C_2, \dots, C_K\}$ and their corresponding clustering center vectors are $\{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_K\}$. SP constructs the index tree by storing $\{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_K\}$ into intermediate nodes and $\{\vec{f}_1, \vec{f}_2, \dots, \vec{f}_n\}$ into leaf nodes. The root node of index is the average value of clustering centers, $\vec{r} = (\sum_{i=1}^K \vec{c}_i / K)$. The child node values of C_i ($i \in [1, K]$) are the feature vectors that belong to the same cluster as the C_i . SP encrypts leaf nodes $\{\vec{f}_1, \vec{f}_2, \dots, \vec{f}_n\}$ (lines 9 to 16), intermediate nodes $\{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_K\}$ (lines 17 to 19) and root node $\vec{r}$ (lines 20 to 21). Finally, the encrypted index $E(T)$ is constructed.

2) *Encrypt the Image Dataset*: SP encrypts each image img_i ($i \in [1, n]$) in IMG with the AES algorithm. The encrypted images are $\{\text{cimg}_1, \text{cimg}_2, \dots, \text{cimg}_n\}$, where $\text{cimg}_i \leftarrow \text{AES.Enc}(k_{\text{img}}, \text{img}_i)$.

3) *Generate the MAC Values*: SP generates an MAC value for cimg_i ($i \in [1, n]$) using SHA256. The MAC values are $\{\text{MAC}_1, \text{MAC}_2, \dots, \text{MAC}_n\}$, where $\text{MAC}_i \leftarrow \text{MAC}(k_{\text{MAC}}, \text{cimg}_i \| i)$, “ $\|$ ” denotes the concatenation.

4) *Send Ciphertexts*: The encrypted index $E(T)$ and $\{i, \text{cimg}_i, \text{MAC}_i\}_{i=1}^n$ are outsourced to CS₁.

3) *Query Token Generation*: For the query image img_q , QU extracts the feature vector $\vec{f}_q$ with the pretrained CNN model. Then, QU selects two thresholds to generate two

Algorithm 1: Build the Encrypted Index

Input: The image dataset IMG .

Output: The encrypted index $E(T)$.

```

1 for  $i \in [1, n]$  do
2   Extract feature vector  $\vec{f}_i$  from  $\text{img}_i$  using the
   pre-trained CNN model;
3 Set  $F = \{\vec{f}_i\}_{i=1}^n$  where  $\vec{f}_i = (f_{i,1}, f_{i,2}, \dots, f_{i,d})$ ;
4 Cluster  $F$  using the clustering algorithm to get  $K$  centers
 $C_1, \dots, C_t, \dots, C_K$  and their values are
 $\vec{c}_1, \dots, \vec{c}_t, \dots, \vec{c}_K$ , where  $\vec{c}_t = (c_{t,1}, c_{t,2}, \dots, c_{t,d})$ ;
5 Set  $\{C_i\}_{i=1}^K$  as  $K$  intermediate nodes, and the node values
are  $\{\vec{c}_i\}_{i=1}^K$ ;
6 for each intermediate node  $C_i$  do
7   Set feature vectors which are clustered into  $C_i$  as the
   child node values of  $C_i$ ;
8 Set  $\vec{r} = \frac{\sum_{i=1}^K \vec{c}_i}{K}$  as the value of root node;
9 for  $i \in [1, n]$  do
10   $\vec{f}'_i = (\vec{f}_i, 1, \|\vec{f}_i\|^2)$ ;
11  Select  $a_{i,1}, a_{i,2}, \dots, a_{i,l}$  from  $\{\mathcal{U} \cup 0\}$ ;
12  //  $\mathcal{U}$  denotes the attribute universal
   set, 0 denotes the wildcard
   attribute that SP doesn't care
13   $\vec{a}_i = (a_{i,1}, a_{i,2}, \dots, a_{i,l})$ ;
14   $\vec{a}'_i = (\vec{a}_i, -2a_{i,1}^2, -2a_{i,2}^2, \dots, -2a_{i,l}^2, \sum_{t=1}^l a_{i,t}^3)$ ;
15   $\vec{f}''_i = (\vec{f}'_i, \vec{a}'_i)$ ;
16  Encrypt each element of  $\vec{f}''_i$  with  $\text{SHE.Enc}()$ ;
17 for  $t \in [1, K]$  do
18   $\vec{c}'_t = (\vec{c}_t, 1, \|\vec{c}_t\|^2)$ ;
19  Encrypt each element of  $\vec{c}'_t$  with  $\text{SHE.Enc}()$ ;
20  $\vec{r}' = (\vec{r}, 1, \|\vec{r}\|^2)$ ;
21 Encrypt each element of  $\vec{r}'$  with  $\text{SHE.Enc}()$ .
```

queries for retrieving intermediate and leaf nodes, respectively. To improve the retrieval recall rate, the threshold for intermediate nodes is larger than that for leaf nodes so that more matched leaf nodes are obtained. Specifically, QU selects a smaller threshold $\tau_{\text{leaf}} > 0$ and extends $\vec{f}_q$ to

$$\vec{f}'_q = (-2\vec{f}_q, \|\vec{f}_q\|^2 - \tau_{\text{leaf}}^2, 1). \quad (6)$$

To access SP's data, QU generates an attribute vector based on his/her attributes as $\vec{v} = (v_1, v_2, \dots, v_l) \in \mathcal{U}^l$, then extends $\vec{v}$ to

$$\vec{v}' = \gamma^2 (v_1^2, v_2^2, \dots, v_l^2, \vec{v}, 1) \quad (7)$$

where $\gamma > \tau_{\text{leaf}} > 0$ is a random number selected by QU. After that, QU appends the extended attribute vector to $\vec{f}'_q$ to obtain

$$\vec{f}''_{q_{\text{leaf}}} = (\vec{f}'_q, \vec{v}'). \quad (8)$$

Then, QU selects a larger threshold $\tau_{\text{med}} > 0$ to extend $\vec{f}'_q$ to

$$\vec{f}'_{q_{\text{med}}} = (-2\vec{f}_q, \|\vec{f}_q\|^2 - \tau_{\text{med}}^2, 1). \quad (9)$$

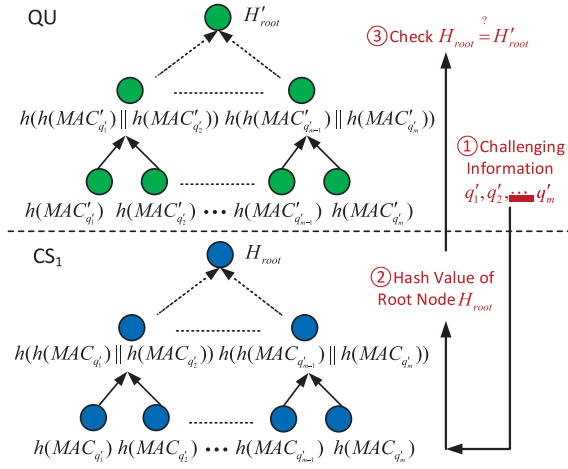


Fig. 4. Results verification.

Next, QU uses SHE.Enc() to encrypt $\vec{f}_{q_{leaf}}''$ to $E(\vec{f}_{q_{leaf}}'')$ and $\vec{f}_{q_{med}}'$ to $E(\vec{f}_{q_{med}}')$. Finally, QU sends $E(\vec{f}_{q_{leaf}}'')$ and $E(\vec{f}_{q_{med}}')$ as the query token to CS₁.

4) *Query Processing*: Upon receiving the query token from QU, CS₁ and CS₂ first retrieve the encrypted index tree and the encrypted image dataset, then return the matched encrypted images to QU for decryption. After that, QU verifies and decrypts the retrieval results.

1) *Retrieve the Encrypted Index*: As shown in Algorithm 2, CS₁ first computes the inner product $E(IP_{t,q})$ ($t \in [1, K]$) of intermediate node value and query token (lines 1 to 7), then computes the sign of $IP_{t,q}$ based on $E(IP_{t,q})$ by using the sign computation protocol (shown in Section III-B) together with CS₂ (lines 8 to 13). If $\text{sign}(IP_{t,q}) = -1$ holds, CS₂ returns t to CS₁ and CS₁ continues to retrieve the child nodes of C_t . Specifically, CS₁ computes the inner product $E(IP_{i,q})$ ($t \in [1, K]$) of child node and query token (lines 14 to 21). Then, CS₁ and CS₂ compute the sign of $IP_{i,q}$ by using the sign computation protocol (lines 22 to 27). If $\text{sign}(IP_{i,q}) = -1$ holds, CS₂ adds the identifier i to the result set R and returns R to CS₁.

2) *Retrieve the Encrypted Image Dataset*: According to the identifiers in R , CS₁ finds out the corresponding encrypted images and returns them to QU.

3) *Verify and Decrypt the Retrieval Results*: Assume that c ciphertext images $\{\text{cimg}_{q_1}, \text{cimg}_{q_2}, \dots, \text{cimg}_{q_c}\}$ are returned as retrieval results. QU randomly selects $m \in [0, c]$ ciphertext images $\{\text{cimg}_{q'_1}, \text{cimg}_{q'_2}, \dots, \text{cimg}_{q'_m}\}$ from results. Then, QU sends the challenging information $\{q'_1, q'_2, \dots, q'_m\}$ to CS₁. CS₁ constructs an MHT based on challenging information. In this MHT, the hash values of leaf nodes are computed as $h(\text{MAC}_i)$, where $i \in \{q'_1, \dots, q'_m\}$. Note that the leaf nodes are arranged from left to right according to i from smallest to largest. As shown in Fig. 4, the intermediate node is generated by two child nodes, and its value is computed by hashing the concatenation of the hash values of child nodes. If only one child node exists, the hash

Algorithm 2: Retrieve the Encrypted Index

Input: The query token and the encrypted index.

Output: The retrieval results.

//CS₁:

1 Initialize an empty set $IP = \{\}$;

2 **for** $t \in [1, K]$ **do**

3 Compute the inner product

$$E(IP_{t,q}) = E(\vec{c}_t') * E(\vec{f}_{q_{med}}');$$

4 Select two random numbers $r_1, r_2 \in \{0, 1\}^{k_1}$ where $r_1 > r_2 > 0$;

5 Compute $E(IP_{t,q})' = r_1 * E(IP_{t,q}) + r_2 * E(-1) \bmod N$;

6 Add $E(IP_{t,q})'$ to IP ;

7 Send IP to CS₂;

//CS₂:

8 Initialize an empty set $R = \{\}$;

9 **for each** $E(IP_{t,q})'$ in IP **do**

10 Decrypt $E(IP_{t,q})'$ with SHE.Dec() and obtain

$$r_1 * IP_{t,q} - r_2;$$

11 **if** $r_1 * IP_{t,q} - r_2 < 0$ **then**

12 Add t to R ;

13 Send R to CS₁;

//CS₁:

14 Initialize an empty set $IP = \{\}$;

15 **for** $t \in R$ **do**

16 **for each** child node value $\vec{f}_i$ of C_t **do**

17 Compute the inner product

$$E(IP_{i,q}) = E(\vec{f}_i'') * E(\vec{f}_{q_{leaf}}');$$

18 Select two random numbers $r_1, r_2 \in \{0, 1\}^{k_1}$ where $r_1 > r_2 > 0$;

19 Compute $E(IP_{i,q})' = r_1 * E(IP_{i,q}) + r_2 * E(-1) \bmod N$;

20 Add $E(IP_{i,q})'$ to IP ;

21 Send IP to CS₂;

//CS₂:

22 Initialize an empty set $R = \{\}$;

23 **for each** $E(IP_{i,q})'$ in IP **do**

24 Decrypt $E(IP_{i,q})'$ with SHE.Dec() and obtain

$$r_1 * IP_{i,q} - r_2;$$

25 **if** $r_1 * IP_{i,q} - r_2 < 0$ **then**

26 Add i to R ;

27 Send R to CS₁;

value of intermediate node is computed by hashing the hash value of this child node. In this way, CS₁ finally obtains the hash value H_{root} of the root node of MHT and sends H_{root} to QU. QU computes the MAC value $\text{MAC}_i' \leftarrow \text{MAC}(k_{\text{MAC}}, \text{cimg}_i \| i)$ for $i \in \{q'_1, \dots, q'_m\}$. Then, QU builds an MHT with the same method as CS₁ and obtains the hash value H'_{root} of root node. Finally, QU checks whether $H_{\text{root}} = H'_{\text{root}}$ holds. If the above equation holds, the returned results are correct and QU can decrypt the images with the key k_{img} by

using $\text{AES.Dec}()$, otherwise the returned ciphertext images have been tampered with.

Remark 1: For each encrypted image cim_i ($i \in [1, n]$), the MAC value is computed as

$$\text{MAC}_i \leftarrow \text{MAC}(k_{\text{MAC}}, \text{cim}_i \| i). \quad (10)$$

Here, $\text{MAC}()$ can be SHA256. Thus, CS_1 cannot tamper with or forge the MAC values without the key k_{MAC} . Further, the root node of MHT generated during the results verification phase is unique and cannot be tampered with or forged.

Remark 2: A key challenge in many privacy-preserving schemes is the tradeoff between efficiency and security. The time-consuming verification process will become a bottleneck for the schemes to be applied to large-scale retrieval scenarios. To achieve optimal performance, the query user can request proofs as infrequently as possible while the cloud server still stays honest [31]. We design an incentive mechanism. Specifically, if CS_1 returns correct results, it gets paid from QU. On the contrary, if the results are wrong, CS_1 will be punished seriously. With the view of game theory, CS_1 is motivated to provide honest results to QU. Assume that the reward received by CS_1 for answering a query is r , the penalty received by CS_1 for being malicious is p , the value of results (*i.e.*, the profit of being malicious without being detected) is v and the probability that QU requests a proof is q . CS_1 will be honest when the expected profit for being honest is larger than the expected profit for being malicious, which is

$$r > (1 - q)v - qp. \quad (11)$$

Therefore, for a query with a small value compared with the penalty, the rational CS_1 will be honest even when QU rarely requests the proof; while for the important query, QU can request the proof with a high probability to make CS_1 to be honest. In particular, when $q = 1$, *i.e.*, QU always requests the proof for each query, the rational CS_1 will be honest in order not to be punished. We conclude that CS_1 will be honest as long as the reward is larger than the maintenance cost and the penalty is large enough to hold (11).

C. Bilateral Access Control

In practical applications, access control is not only enforced by the service provider over the query users, but query users may also impose constraints on the service provider [32], [33]. For instance, in an electronic medical record system, a service provider managing patient data from multiple hospitals offers query services to authorized doctors. A doctor may prefer to retrieve data from a specific hospital rather than querying the entire dataset. To accommodate such scenarios, a bilateral access control mechanism is required. Our PPSR can be easily extended to bilateral access control. SP adds his/her attributes on each feature vector and QU adds an access control policy to the query vector. In this way, only images satisfying the bilateral access control policy will be returned.

Example: Suppose $\mathcal{U} = \{1, 2, 3\}$, the attribute sets for SP_1 , SP_2 , QU_1 and QU_2 are $\vec{w}_1 = \{1, 3\}$, $\vec{w}_2 = \{2, 3\}$, $\vec{v}_1 = \{1, 3\}$ and $\vec{v}_2 = \{2, 1\}$, respectively. SP_1 sets the access policy for image img_1 to $\vec{a}_1 = (1, 0)$, while SP_2 sets $\vec{a}_2 = (2, 3)$ for

image img_2 . QU_1 sets the access policy for his/her query token to $\vec{b}_1 = (1, 0)$, while QU_2 sets $\vec{b}_2 = (2, 3)$. As described in Algorithm 1 line 14 and (7), we have

$$\begin{aligned} \vec{w}'_1 &= \gamma_{\text{SP}_1}^2(1, 9, 1, 3, 1), \vec{w}'_2 = \gamma_{\text{SP}_2}^2(4, 9, 2, 3, 1) \\ \vec{v}'_1 &= \gamma_{\text{QU}_1}^2(1, 9, 1, 3, 1), \vec{v}'_2 = \gamma_{\text{QU}_2}^2(4, 1, 2, 1, 1) \\ \vec{a}'_1 &= (1, 0, -2, 0, 1), \vec{a}'_2 = (2, 3, -8, -18, 35) \\ \vec{b}'_1 &= (1, 0, -2, 0, 1), \vec{b}'_2 = (2, 3, -8, -18, 35). \end{aligned}$$

By computing the inner products, we have

$$\begin{aligned} (\vec{w}'_1, \vec{a}'_1) \cdot (\vec{b}'_1, \vec{v}'_1)^\top &= 0, (\vec{w}'_2, \vec{a}'_2) \cdot (\vec{b}'_1, \vec{v}'_1)^\top \neq 0 \\ (\vec{w}'_1, \vec{a}'_1) \cdot (\vec{b}'_2, \vec{v}'_2)^\top &\neq 0, (\vec{w}'_2, \vec{a}'_2) \cdot (\vec{b}'_2, \vec{v}'_2)^\top \neq 0. \end{aligned}$$

It is obvious that SP_1 and QU_1 satisfy each other's access control policies.

VI. THEORETIC ANALYSIS

In this section, we theoretically analyze the correctness and security of PPSR.

A. Correctness Analysis

PPSR allows both access control and similarity retrieval to be simultaneously performed through a single inner product operation. For simplicity, we analyze its correctness in the plaintext scenario. $\vec{f}'_i \cdot \vec{f}'_q^\top$ can be regard as the sum of $\vec{f}'_i \cdot \vec{f}'_q^\top$ and $\vec{a}'_i \cdot \vec{v}'_q^\top$

$$\begin{aligned} IP_{i,q} &= \vec{f}'_i \cdot \vec{f}'_q^\top \\ &= (\vec{f}'_i, \vec{a}'_i) \cdot (\vec{f}'_q, \vec{v}'_q)^\top \\ &= \vec{f}'_i \cdot \vec{f}'_q^\top + \vec{a}'_i \cdot \vec{v}'_q^\top. \end{aligned} \quad (12)$$

For feature vectors, CS_1 computes the inner product of $\vec{f}'_i$ and $\vec{f}'_q$ as

$$\begin{aligned} \vec{f}'_i \cdot \vec{f}'_q^\top &= (\vec{f}_i, 1, \|\vec{f}_i\|^2) \cdot (-2\vec{f}_q, \|\vec{f}_q\|^2 - \tau^2, 1)^\top \\ &= \|\vec{f}_i\|^2 + \|\vec{f}_q\|^2 - 2\vec{f}_i \cdot \vec{f}_q - \tau^2 \\ &= \|\vec{f}_i - \vec{f}_q\|^2 - \tau^2. \end{aligned} \quad (13)$$

If $\vec{f}'_i \cdot \vec{f}'_q^\top \leq 0$, $\|\vec{f}_i - \vec{f}_q\|^2 \leq \tau^2$ holds. Thus, the image img_i matches the query request.

For attribute vector, CS_1 computes the inner product of $\vec{a}'_i$ and $\vec{v}'_q$ as

$$\begin{aligned} \vec{a}'_i \cdot \vec{v}'_q^\top &= (a_{i,1}, a_{i,2}, \dots, a_{i,l}, -2a_{i,1}^2, -2a_{i,2}^2, \dots, -2a_{i,l}^2) \\ &\quad \cdot \gamma^2(v_1^2, v_2^2, \dots, v_l^2, v_1, v_2, \dots, v_l, 1)^\top \\ &= \gamma^2 \left(\sum_{t=1}^l a_{i,t} \cdot v_t^2 - 2 \sum_{t=1}^l a_{i,t}^2 \cdot v_t + \sum_{t=1}^l a_{i,t}^3 \right) \\ &= \gamma^2 \left(\sum_{t=1}^l a_{i,t} \cdot (v_t - a_{i,t})^2 \right). \end{aligned} \quad (14)$$

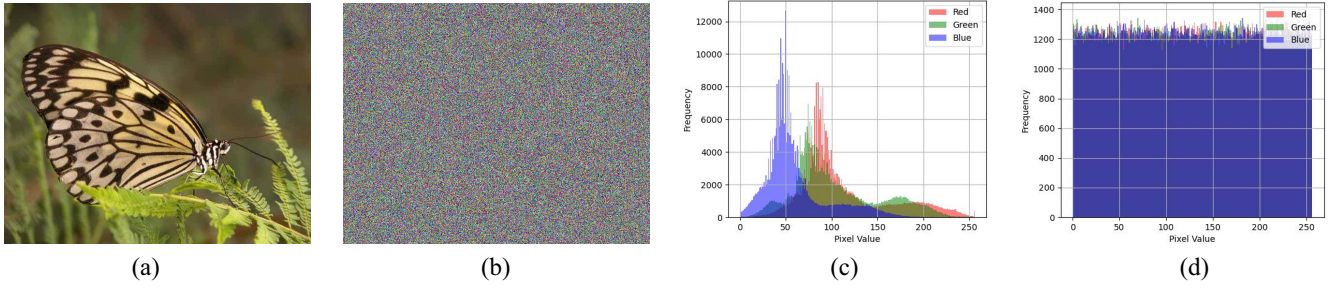


Fig. 5. Plaintext and ciphertext comparison. (a) Plaintext image. (b) Ciphertext image. (c) Plaintext image histogram. (d) Ciphertext image histogram.

If $\vec{a}'_i \cdot \vec{v}'^\top = 0$, $a_{i,t} = 0$ or $a_{i,t} = v_t$ holds. It indicates that QU's attributes match the SP's access control policy.

Note that $\|\vec{f}_i - \vec{f}_q\|^2 - \tau^2$ is either positive or nonpositive, and $\vec{a}'_i \cdot \vec{v}'^\top$ is either 0 or at least γ^2 (with $\gamma^2 > \tau^2 > 0$). Therefore, $IP_{i,q} \leq 0$ holds iff both conditions $\|\vec{f}_i - \vec{f}_q\|^2 - \tau^2 \leq 0$ and $\vec{a}'_i \cdot \vec{v}'^\top = 0$ are satisfied. It shows that the Euclidean distance between SP's image img_i and QU's query image img_q is less than or equal the threshold τ and QU's attributes match the SP's access control policy.

B. Security Analysis

As described in the design goals, we aim to protect the privacy of the image dataset, index, and query token from two cloud servers.

First, we prove that the image dataset is privacy preserving. In PPSR, each image img_i in $\text{IMG} = \{\text{img}_1, \text{img}_2, \dots, \text{img}_n\}$ is encrypted with AES algorithm in CBC mode. As shown in Fig. 5, we use a butterfly image as an example. Fig. 5(a) presents the plaintext image, while Fig. 5(b) shows the corresponding ciphertext obtained by applying AES encryption. The information entropy of the ciphertext image in the R, G, and B channels reaches 7.9994, which is close to the theoretical maximum of 8 for an 8-bit RGB images. This high entropy implies that the ciphertext image appears random and reveals no useful information. Fig. 5(c) and (d) depict the histograms of the plaintext and ciphertext images, respectively. It can be observed that the histogram of the ciphertext image exhibits a nearly uniform distribution, indicating that AES encryption effectively conceals the statistical characteristics of the plaintext image. Therefore, as long as the AES encryption key k_{img} remains confidential, the plaintext image cannot be recovered by an adversary.

Next, We prove that our index and query token are selectively secure in the real world and ideal world. We define two leakage functions of CS_1 and CS_2 .

- 1) $\mathcal{L}(\text{CS}_1)$: The information leaked to CS_1 contains the public parameter PP , encrypted index tree $E(T)$, query token $\{E(\vec{f}_{q,\text{leaf}}''), E(\vec{f}_{q,\text{med}}'')\}$, access pattern of $E(T)$, a ciphertext $E(-1)$, retrieval result R .
 - 2) $\mathcal{L}(\text{CS}_2)$: The information leaked to CS_2 contains the public parameter PP , secret key SK , the identifiers $t \in [1, K]$, $i \in [1, n]$ of matched intermediate and leaf nodes.
- Based on $\mathcal{L}(\text{CS}_1)$ and $\mathcal{L}(\text{CS}_2)$, we define the real world and ideal world as follows.

Real World: In the real world, there are two noncollusive probabilistic polynomial time (PPT) adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, and a challenger $\mathcal{B}$.

- 1) **System Initialization and Data Outsourcing Phases:** $\mathcal{A}_1$ sends an image dataset $\text{IMG} = \{\text{img}_1, \text{img}_2, \dots, \text{img}_n\}$ to $\mathcal{B}$. Then, $\mathcal{B}$ runs the system initialization algorithm to generate the keys $\{PP, SK\}$. Next, $\mathcal{B}$ publishes PP and sends SK to $\mathcal{A}_2$. Finally, with the image dataset IMG , $\mathcal{B}$ runs the data outsourcing algorithm to generate the encrypted index tree $E(T)$.
- 2) **Query Token Generation Phase 1:** $\mathcal{A}_1$ sends t_1 query vectors $\{\vec{f}_{q_1}, \vec{f}_{q_2}, \dots, \vec{f}_{q_{t_1}}\}$ to $\mathcal{B}$. $\mathcal{B}$ runs the query token generation algorithm to generate the query token $\{E(\vec{f}_{q,\text{leaf}}''), E(\vec{f}_{q,\text{med}}'')\}_{i=1}^{t_1}$ and returns them to $\mathcal{A}_1$.
- 3) **Challenge Phase:** $\mathcal{B}$ sends $E(T)$ to $\mathcal{A}_1$.
- 4) **Query Token Generation Phase 2:** $\mathcal{A}_1$ sends $t_2 - t_1$ query vectors $\{\vec{f}_{q_{t_1+1}}, \vec{f}_{q_{t_1+2}}, \dots, \vec{f}_{q_{t_2}}\}$ to $\mathcal{B}$. $\mathcal{B}$ runs the query token generation algorithm to generate the query token $\{E(\vec{f}_{q,\text{leaf}}''), E(\vec{f}_{q,\text{med}}'')\}_{i=t_1+1}^{t_2}$ and returns them to $\mathcal{A}_1$.
- 5) **Query Processing Phase:** When $\mathcal{A}_1$ receives the t_2 query tokens $\{E(\vec{f}_{q,\text{leaf}}''), E(\vec{f}_{q,\text{med}}'')\}_{i=1}^{t_2}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ cooperatively run the query processing algorithm to retrieve $E(T)$ and obtain the t_2 retrieval results $\{R_i\}_{i=1}^{t_2}$, where R_i contains the identifiers of matched images.

In the real world, $\mathcal{A}_1$'s view $\mathcal{V}_{\mathcal{A}_1, \text{Real}}$ contains $E(T)$, $\{E(\vec{f}_{q,\text{leaf}}''), E(\vec{f}_{q,\text{med}}'')\}_{i=1}^{t_2}$, $\{R_i\}_{i=1}^{t_2}$, and $\mathcal{A}_2$'s view $\mathcal{V}_{\mathcal{A}_2, \text{Real}}$ contains the sign of inner products as well as $\{R_i\}_{i=1}^{t_2}$.

Ideal World: The ideal world has two PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, and a simulator $\mathcal{S}$ with leakage $\mathcal{L}(\text{CS}_1)$ and $\mathcal{L}(\text{CS}_2)$.

- 1) **System Initialization and Data Outsourcing Phases:** $\mathcal{A}_1$ sends the image dataset $\text{IMG} = \{\text{img}_1, \text{img}_2, \dots, \text{img}_n\}$ to $\mathcal{S}$. Then, $\mathcal{S}$ first publishes PP and sends $SK \in \mathcal{L}(\text{CS}_2)$. Next, with the leakage $\mathcal{L}(\text{CS}_1)$ and $\mathcal{L}(\text{CS}_2)$, $\mathcal{S}$ generates an encrypted index tree $E(T)'$ by self-blinding nodes values. Specifically, $\mathcal{S}$ uses $E(-1) \in \mathcal{L}(\text{CS}_1)$ to self-bind each element $E(f_{i,s}'')$ in $E(\vec{f}_i'')$ as $E(f_{i,s}'')' = (E(f_{i,s}'') + \sum_{\eta, \sigma} (E(-1)^{2\eta} + E(-1)^{2\sigma-1})) \bmod N$. According to SHE's homomorphic properties, $E(f_{i,s}'')'$ is also a ciphertext of $f_{i,s}''$. Similarly, $\mathcal{S}$ self-blinds $E(\vec{c}_i')$, $E(\vec{r}')$ to $E(\vec{c}_i')'$, $E(\vec{r}')'$.
- 2) **Query Token Generation Phase 1:** $\mathcal{A}_1$ sends t_1 query vectors $\{\vec{f}_{q_1}, \vec{f}_{q_2}, \dots, \vec{f}_{q_{t_1}}\}$ to $\mathcal{S}$. Then, $\mathcal{S}$ uses $E(-1)$ to self-blind $\{E(\vec{f}_{q,\text{leaf}}''), E(\vec{f}_{q,\text{med}}'')\}$ and generates the corresponding query tokens $\{E(\vec{f}_{q,\text{leaf}}'')', E(\vec{f}_{q,\text{med}}'')'\}$ for $i \in [1, t_1]$.

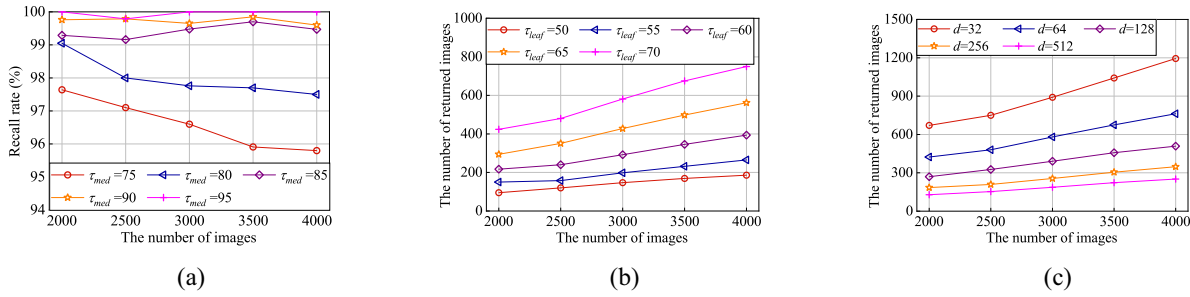


Fig. 6. Recall rate. (a) $d = 64$, $\tau_{\text{leaf}} = 70$. (b) $d = 64$, $\tau_{\text{med}} = 110$, recall rate=100%. (c) $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 110$, recall rate = 100%.

- 3) *Challenge Phase*: $\mathcal{S}$ returns $E(T)'$ to $\mathcal{A}_1$.
- 4) *Query Token Generation Phase 2*: $\mathcal{A}_1$ sends $t_2 - t_1$ query vectors $\{\vec{f}_{q_{t_1+1}}, \vec{f}_{q_{t_1+2}}, \dots, \vec{f}_{q_{t_2}}\}$ to $\mathcal{S}$. $\mathcal{S}$ returns query tokens $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}'$ by self-blinding $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}'$ for $i \in [t_1, t_2]$.
- 5) *Query Processing Phase*: When $\mathcal{A}_1$ obtains the t_2 query tokens $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ cooperatively run the query processing algorithm to retrieve $E(T)'$ and obtain the t_2 retrieval results $\{R'_i\}_{i=1}^{t_2}$, where R'_i contains the identifiers of matched images.

In the ideal world, $\mathcal{A}_1$'s view $\mathcal{V}_{\mathcal{A}_1, \text{Ideal}}$ contains $E(T)'$, $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$, $\{R'_i\}_{i=1}^{t_2}$, and $\mathcal{A}_2$'s view $\mathcal{V}_{\mathcal{A}_2, \text{Ideal}}$ contains the sign of inner products as well as $\{R'_i\}_{i=1}^{t_2}$.

Theorem 1: Index and query are secure with the leakage $\mathcal{L}(\text{CS}_1)$ and $\mathcal{L}(\text{CS}_2)$ iff the SHE algorithm is CPA-secure.

Proof: We prove that any PPT adversary, e.g., $\mathcal{A}_1$ and $\mathcal{A}_2$, cannot distinguish the view of the real and ideal model.

$\mathcal{A}_1$: For the encrypted index tree $E(T)$ and $E(T)'$, both intermediate nodes and leaf nodes are encrypted with $\text{SHE.Enc}()$. Since SHE is semantically secure against CPA as proved in [26], $\mathcal{A}_1$ cannot distinguish the $E(T)$ and $E(T)'$. Similarly, query tokens are encrypted with $\text{SHE.Enc}()$. Thus, $\mathcal{A}_1$ also cannot distinguish $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$ and $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$. Since the encrypted tree $E(T)$ and $E(T)'$, the query tokens $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$ and $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$ have same plaintexts in real world and ideal world, the retrieval results $\{R_i\}_{i=1}^{t_2}$ and $\{R'_i\}_{i=1}^{t_2}$ are same.

$\mathcal{A}_2$: $\mathcal{A}_2$ can view the sign of inner products and $\{R_i\}_{i=1}^{t_2}$, $\{R'_i\}_{i=1}^{t_2}$. Since $E(T)$, $E(T)'$ have the same plaintexts, $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$ and $\{E(\vec{f}_{q,i_{\text{leaf}}}''), E(\vec{f}_{q,i_{\text{med}}}')\}_{i=1}^{t_2}$ also have the same plaintexts, the retrieval results are same in real world and ideal world. For the encrypted inner product $E(IP_{t,q})$, $\mathcal{A}_2$ obtains $\text{sign}(IP_{t,q})$ from the sign computation protocol. In this protocol, $\mathcal{A}_1$ adds two random numbers $r_1, r_2 \in \{0, 1\}^{k_1}$ into $E(IP_{t,q})$ with SHE's homomorphic properties. After that, $\mathcal{A}_2$ decrypts it and obtains $r_1 * IP_{t,q} + r_2$. It is obviously that $\mathcal{A}_2$ cannot deduce $IP_{t,q}$ without r_1, r_2 . Similarly, $\mathcal{A}_2$ cannot deduce $IP_{i,q}$ with $E(IP_{i,q})$ and $\text{sign}(IP_{i,q})$. Although $\mathcal{A}_2$ gets $\{R_i\}_{i=1}^{t_2}$, he has no way to access corresponding images since $\mathcal{A}_1$ and $\mathcal{A}_2$ are noncollusive. Thus, $\{R_i\}_{i=1}^{t_2}$ are useless for $\mathcal{A}_2$.

Therefore, $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot distinguish the view of real and ideal model, which proves that index and query token is selectively secure with the leakage $\mathcal{L}(\text{CS}_1)$ and $\mathcal{L}(\text{CS}_2)$. ■

VII. PERFORMANCE EVALUATION

In this section, we evaluate performance of PPSR and compare it with two CPA-secure baselines schemes. The first is FNNS [16], a fast nearest neighbor search scheme based on Paillier HE designed for Healthcare Industry 4.0. The second is TIR [13], a threshold image retrieval scheme based on matrix encryption in cloud-assisted IoT environments. We implement PPSR, FNNS and TIR in C++ and conduct experiments on a machine with a 16-core Intel Core i7-1360P and 32GB RAM, running the Ubuntu 22.04 operating system. The parameters of SHE are set as $k_0 = 1024$, $k_1 = 40$, $k_2 = 60$. The security parameter of Paillier HE is set as $\kappa = 512$. Since there is no real dataset containing both images and attributes, we evaluate the accuracy performance on a real image dataset Caltech256 [34] and without attributes. We randomly select 40, 50, 60, 70, 80 images from the first 50 categories and obtain 2000, 2500, 3000, 3500, 4000 images. The pretrained CNN model ResNet50 and principal component analysis (PCA) technology are used to extract features and reduce dimension. To test the performance of access control, we randomly generate access and attribute vectors by selecting values between 0 and 10 for each element.

A. Recall Rate

We define the recall rate as $(|R \cap R_t|/|R_t|)$, where R denotes the set of results retrieved under PPSR (including encryption and indexing), and R_t represents the ground truth results obtained by directly performing similarity retrieval on plaintext images without any encryption, indexing, or access control. Fig. 6 shows the recall rate under different thresholds τ_{leaf} , τ_{med} , feature vector dimension d , and total number of images n . In Fig. 6(a), when τ_{med} and n increase, the recall rate remains above 95%. In Fig. 6(b) and (c), since the recall rate reaches 100%, we give the total number of returned results. Fig. 6(b) shows that increasing the threshold leads to more returned results, particularly as n grows. Fig. 6(c) displayed the total number of returned results decreases as d increases when we set $\tau_{\text{leaf}} = 70$ and $\tau_{\text{med}} = 110$. This is because when the dimension increases, the distance between feature vectors

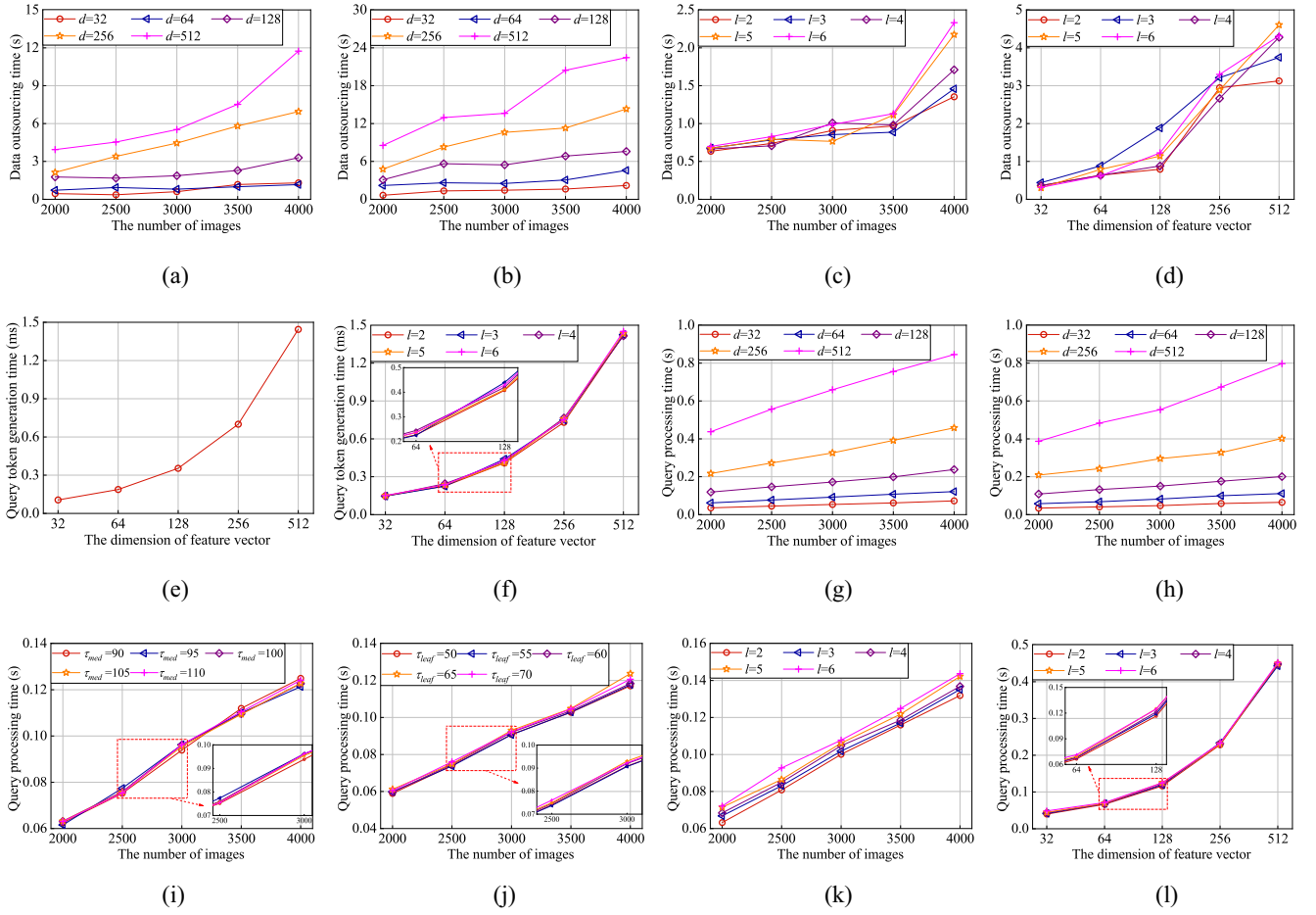


Fig. 7. Computational costs of data outsourcing, query token generation and query processing. (a) Without access control, $K = 5$. (b) Without access control, $K = 15$. (c) With access control, $d = 64$, $K = 5$. (d) With access control, $n = 2000$, $K = 5$. (e) Without access control. (f) With access control. (g) Without access control, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (h) Without access control, $K = 15$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (i) Without access control, $d = 64$, $K = 5$, $\tau_{\text{leaf}} = 70$. (j) Without access control, $d = 64$, $K = 5$, $\tau_{\text{med}} = 95$. (k) With access control, $d = 64$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (l) With access control, $n = 2000$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$.

will also increase, resulting in a decrease in the total number of distances less than the threshold.

B. Computational Cost of Dataset Outsourcing

Fig. 7(a)–(d) show the data outsourcing time under varying total number of images n , total number of clusters K , feature vector dimension d , and access vector dimension l . The theoretical computational cost includes one clustering operation, $(n + K + 1)(d + 2) + n(2l + 1)$ HEs, n AES encryptions, and n MAC operations. The data outsourcing time increases with the growth of n , K , d , and l . Comparing Fig. 7(a) and (b), we observe that the data outsourcing time rises significantly when K increases, primarily because the clustering operation accounts for approximately 75% of the total time cost. After incorporating the l -dimensional access vector for each image, the data outsourcing time is shown in Fig. 7(c) and (d). When $n = 2000$ and $d = 64$, and l increases from 2 to 3, an additional 2000 HE operations are required theoretically, resulting in a 0.25s increase in practice. Nevertheless, the increase in l does not lead to a sharp increase in data outsourcing time, which indicates that the proposed access control mechanism imposes little additional burden.

C. Computational Cost of Query Token Generation

Fig. 7(e) and (f) display the query token generation time as a function of feature vector dimension d and access vector dimension l , respectively. The theoretical computational cost of this phase consists of $2d + 2l + 5$ HE operations. In Fig. 7(e), the query token generation time grows with the increase of d . After incorporating attribute vectors of varying dimensions, Fig. 7(f) shows that the query token generation time increases slightly as l grows. For example, when l increases from 2 to 3, the additional cost corresponds to only two extra HE operations, resulting in the curves in Fig. 7(f) being almost overlapping.

D. Computational Cost of Query Processing

Fig. 7(g)–(l) illustrate the variation of query processing time with respect to total number of images n , total number of clusters K , feature vector dimension d , thresholds τ_{leaf} , τ_{med} , and attribute vector dimension l . We set $q'_m = (|R|/2)$ and denote the number of images in the candidate clusters by $|C|$. The theoretical computational cost of query processing includes $(K + |C|)(d + 2l + 3)$ homomorphic ciphertext multiplications, $K + |C|$ homomorphic decryptions, $2(K + |C|)$

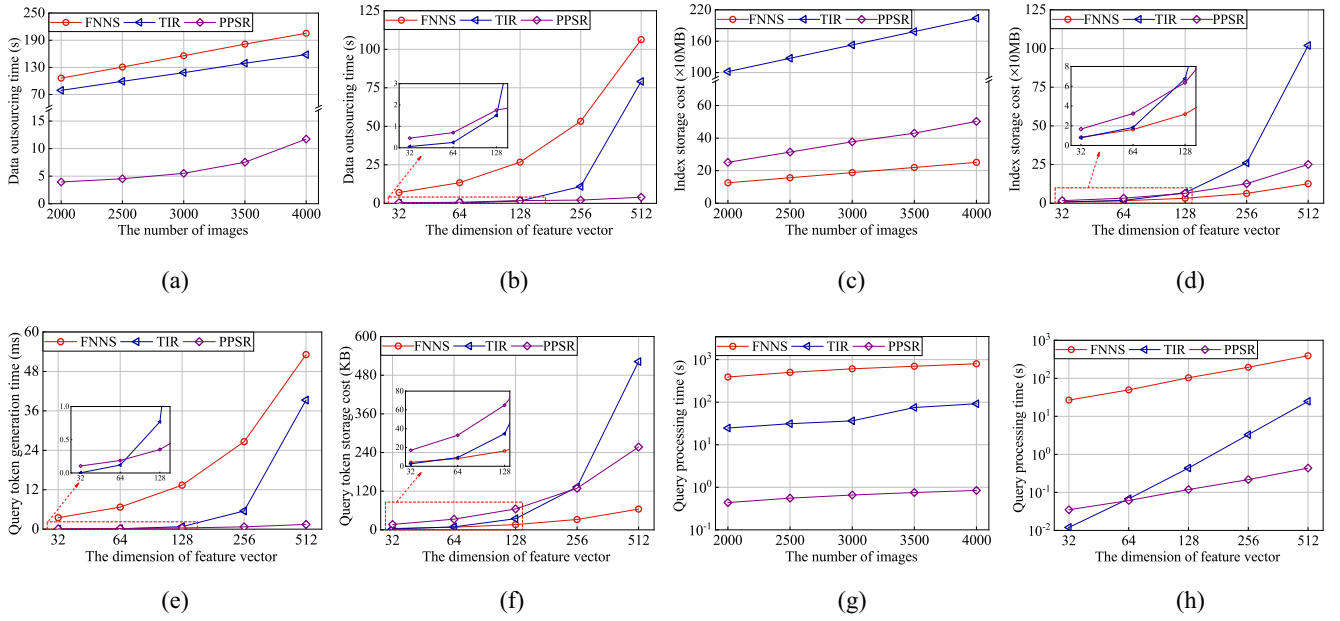


Fig. 8. Performance in different schemes. (a) $d = 512$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (b) $n = 2000$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (c) $d = 512$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (d) $n = 2000$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (e) $n = 2000$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (f) $n = 2000$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (g) $d = 512$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$. (h) $n = 2000$, $K = 5$, $\tau_{\text{leaf}} = 70$, $\tau_{\text{med}} = 95$.

homomorphic number multiplications, q'_m MACs, $4q'_m - 2$ hashes, and $|R|$ AES decryptions. An increase in the total number of images n leads to more images in each cluster, thereby increasing the number of images in the candidate clusters $|C|$. Fig. 7(g) and (h) show that query processing time grows with n and d . In contrast, a larger K reduces the time required for query processing. Fig. 7(i) and (j) show that, for a fixed n , the query processing time remains almost unchanged across different thresholds. This is because varying the thresholds only affects the size of R , while the time spent on hashing and AES decryption is negligible in the overall query processing time, making the impact of threshold changes insignificant. When access control is incorporated, the number of homomorphic multiplications increases, and the number of retrieved results may decrease. Therefore, the query processing time in Fig. 7(k) is slightly higher than that in Fig. 7(i) and (j). Fig. 7(l) shows that the query processing time remains almost constant as l varies for a given d . In fact, as d increases from 32 to 512, the influence of a slight increase in l on the overall query processing time becomes negligible, resulting in overlapping curves in Fig. 7(l).

E. Comparison

Since image encryption and retrieval are independent, we assume that PPSR, FNNS, and TIR adopt the same image encryption method, and we omit the image encryption step in our comparative analysis. In the data outsourcing phase, PPSR mainly performs one clustering operation and $(n + K + 1)(d + 2) + n(2l + 1)$ SHE HEs, whereas FNNS primarily executes $n(d + 2)$ Paillier HEs, and TIR involves approximately $3n(d + 5)^3$ multiplications. As shown in Fig. 8(a), when $d = 512$ and n increases from 2000 to 4000, the data outsourcing time of PPSR remains below 12s. In comparison, the time

consumption of FNNS increases from 106.3s to 205.45s, and that of TIR grows from 70.09s to 158.13s. In Fig. 8(b), increasing the feature dimension can significantly raises the data outsourcing time for FNNS and TIR, while PPSR remains almost unaffected.

For the index storage cost, PPSR consists of $(n + K + 1)(d + 1)$ SHE ciphertexts, while that of FNNS consists of $n(d + 2)$ Paillier ciphertexts, and TIR generates n matrices of size $(d + 5) \times (d + 5)$. We set the SHE key length to 1024 bits, the Paillier key length to 512 bits, and each short integer in the matrices to 2 bytes. As shown in Fig. 8(c), when $d = 512$ and n increases from 2000 to 4000, the index storage cost of PPSR grows from 251MB to 502.7MB. In contrast, FNNS increases from 125.5MB to 250.9MB, and TIR increases sharply from 1019.6MB to 2039.2MB. In Fig. 8(d), after d increases to 512, TIR incurs significantly higher storage costs compared to FNNS and PPSR, mainly because storing high-dimensional matrices is more expensive than storing large integers. To enhance the index encryption and retrieval efficiency of FNNS, we set the Paillier key length to be half of the SHE key length, resulting in FNNS having a smaller index storage cost compared to PPSR. If the key lengths were the same, the storage costs of the two schemes would be comparable.

In the query token generation phase, PPSR performs $2(d + 2)$ SHE HEs, FNNS performs $d + 2$ Paillier HEs, and TIR performs approximately $3(d + 5)^3$ multiplications. When $d = 32$ and $d = 64$ in Fig. 8(e), TIR takes the least time to generate a query token. As d increases, the query token generation time for FNNS and TIR increases significantly more than that for PPSR. Specifically, when d increases from 128 to 512, the query token generation time for PPSR increases from 0.35ms to 1.44 ms, while the time for FNNS increases from 13.38 ms to 53.15 ms, and the time for TIR increases from 0.76ms to 39.28 ms. Regarding the storage cost of query token, PPSR

stores $2(d+2)$ SHE ciphertexts, FNNS stores $d+2$ Paillier ciphertexts, and TIR stores a matrix of size $(d+5) \times (d+5)$. Since the key length of PPSR is twice that of FNNS, the storage cost of PPSR is approximately four times that of FNNS in Fig. 8(f). Furthermore, the storage cost of TIR increases significantly with d , far exceeding both PPSR and FNNS when $d = 512$. Therefore, PPSR outperforms FNNS and TIR in both time and storage cost, indicating its potential applicability in IoT scenarios.

In the query processing phase, PPSR mainly performs $(K+|C|)(d+2)$ SHE homomorphic ciphertext multiplications and $K+|C|$ SHE homomorphic decryptions. In contrast, FNNS requires at least $3n+5d$ Paillier homomorphic ciphertext multiplications, $6(n+d)$ Paillier HEs, and $2(n+d)$ Paillier homomorphic decryptions. TIR conducts n matrix multiplications and trace computations on matrices of size $(d+5) \times (d+5)$. Since PPSR employs more efficient SHE for data protection and builds an index tree based on clustering, it avoids traversing the entire dataset, resulting in a much slower growth rate in query processing time. Specifically, as shown in Fig. 8(g), PPSR requires less than 1s to process queries over 2000 to 4000 images, whereas FNNS requires at least 392.18s and TIR at least 24.72s. In Fig. 8(h), as d increases, the query processing time of PPSR remains consistently lower than that of FNNS, with a slower growth rate compared to TIR. While achieving the same level of security, the proposed PPSR demonstrates significant performance advantages over both FNNS and TIR, achieving approximately $100\times$ and $4\times$ efficiency improvements when $n = 2000$ and $d = 128$, respectively.

VIII. CONCLUSION

In this article, we proposed an efficient privacy-preserving similarity retrieval scheme with fine-grained access control for ciphertext images. Leveraging the SHE and clustering techniques, PPSR builds an encrypted index tree to improve the retrieval efficiency and provide security guarantee. Moreover, by integrating access control and retrieval into a single inner product operation, PPSR eliminates redundant policy checks and significantly reduces computational cost. Finally, security analysis demonstrated that PPSR is privacy-preserving and performance evaluation validated its efficiency. In future work, we plan to optimize the tree index structure for large-scale deployment on resource-constrained devices.

REFERENCES

- [1] W. Lu, A. Swaminathan, A. L. Varna, and M. Wu, "Enabling search over encrypted multimedia databases," in *Proc. Media Forensics Security*, San Jose, CA, USA, 2009, pp. 404–414.
- [2] J. Yuan, S. Yu, and L. Guo, "SEISA: Secure and efficient encrypted image search with access control," in *Proc. IEEE Conf. Comput. Commun.*, Hong Kong, China, 2015, pp. 2083–2091.
- [3] Y. Li et al., "DVREI: Dynamic verifiable retrieval over encrypted images," *IEEE Trans. Comput.*, vol. 71, no. 8, pp. 1755–1769, Aug. 2022.
- [4] C. Mao, Z. Shen, K. Chen, Y. Liu, Q. Meng, and F. Wang, "DCIRM: Dynamic and controllable image retrieval scheme in multi-owner multi-user settings," *IEEE Trans. Services Comput.*, vol. 17, no. 4, pp. 1435–1448, Jul./Aug. 2024.
- [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.*, Atlanta, GA, USA, 2017, pp. 1–9.
- [6] 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. Inf. Forensics Security*, vol. 11, pp. 2594–2608, 2016.
- [7] 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, May 2017.
- [8] Y. Li et al., "Traceable and controllable encrypted cloud image search in multi-user settings," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2936–2948, Oct.–Dec. 2020.
- [9] Q. Tong et al., "VFIRM: Verifiable fine-grained encrypted image retrieval in multi-owner multi-user settings," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3606–3619, Nov./Dec. 2021.
- [10] S. Khan, H. Abbas, and W. Iqbal, "Verifiable privacy-preserving image retrieval in multi-owner multi-user settings," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 8, no. 2, pp. 1640–1655, Apr. 2024.
- [11] T. Yang et al., "Secure and traceable Multikey image retrieval in cloud-assisted Internet of Things," *IEEE Internet Things J.*, vol. 11, no. 24, pp. 40875–40887, Dec. 2024.
- [12] Y. Li, J. Ma, Y. Miao, X. Liu, and Q. Jiang, "Blockchain-based encrypted image storage and search in cloud computing," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Hyderabad, India, 2022, pp. 413–421.
- [13] 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 Things J.*, vol. 9, no. 15, pp. 13598–13611, Aug. 2022.
- [14] Y. Zhang, L. Zhuo, Y. Peng, and J. Zhang, "A secure image retrieval method based on homomorphic encryption for cloud computing," in *Proc. IEEE Int. Conf. Digit. Signal Process.*, Hong Kong, China, 2014, pp. 269–274.
- [15] M. Li et al., "InstantCryptoGram: Secure image retrieval service," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 2222–2230.
- [16] C. Guo, S. Su, K.-K. R. Choo, and X. Tang, "A fast nearest neighbor search scheme over outsourced encrypted medical images," *IEEE Trans. Ind. Inform.*, vol. 17, no. 1, pp. 514–523, Jan. 2021.
- [17] T. Yang et al., "MU-TEIR: Traceable encrypted image retrieval in the multi-user setting," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1282–1295, Mar./Apr. 2022.
- [18] 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.
- [19] L. Chen et al., "Efficient and secure content-based image retrieval in cloud-assisted Internet of Things," *IEEE Internet Things J.*, vol. 12, no. 5, pp. 6001–6013, Mar. 2025.
- [20] M. Li, Y. Zhu, R. Du, and C. Jia, "Verifiable encrypted image retrieval with reversible data hiding in cloud environment," *IEEE Trans. Cloud Comput.*, vol. 13, no. 1, pp. 397–410, Jan.–Mar. 2025.
- [21] W. K. Wong, D. W.-I. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Providence, RI, USA, 2009, pp. 139–152.
- [22] R. Li, A. X. Liu, Y. Liu, H. Xu, and H. Yuan, "Insecurity and hardness of nearest Neighbor queries over encrypted data," in *Proc. IEEE Int. Conf. Data Eng.*, Macao, China, 2019, pp. 1614–1617.
- [23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, Prague, Czech Republic, 1999, pp. 223–238.
- [24] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 2401–2414, 2016.
- [25] L. Xiao, O. Bastani, and I.-L. Yen, "An efficient homomorphic encryption protocol for multi-user systems," *Cryptol. ePrint Arch.*, IACR, Bellevue, WA, USA, Rep. 2012/193, 2012. [Online]. Available: <https://eprint.iacr.org/2012/193>
- [26] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 4, pp. 2501–2516, Jul./Aug. 2021.
- [27] Y. Zheng, R. Lu, Y. Guan, S. Zhang, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity query with access control in eHealthcare," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 880–893, 2022.
- [28] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine, "Authentic data publication over the Internet," *J. Comput. Secur.*, vol. 11, no. 3, pp. 291–314, 2003.

- [29] Y. Miao, J. Weng, X. Liu, K.-K. R. Choo, Z. Liu, and H. Li, "Enabling verifiable multiple keywords search over encrypted cloud data," *Inf. Sci.*, vol. 465, pp. 21–37, Oct. 2018.
- [30] H. Cheng, H. Wang, X. Liu, Y. Fang, M. Wang, and X. Zhang, "Person re-identification over encrypted outsourced surveillance videos," *IEEE Trans. Depend. Secure Comput.*, vol. 18, no. 3, pp. 1456–1473, May/Jun. 2019.
- [31] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, "FalconDB: Blockchain-based collaborative database," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Portland, OR, USA, 2020, pp. 637–652.
- [32] S. Xu et al., "Match in my way: Fine-grained bilateral access control for secure cloud-fog computing," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 2, pp. 1064–1077, Mar./Apr. 2022.
- [33] M. Zhao, C. Zhang, T. Wu, J. Ni, X. Liu, and L. Zhu, "Revocable and privacy-preserving bilateral access control for cloud data sharing," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 5389–5404, 2024.
- [34] G. Griffin, A. Holub, and P. Perona, *Caltech-256 Object Category Dataset*, California Inst. Technol., Pasadena, CA, USA, 2007.

Yingying Li (Member, IEEE) received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 2017, and the Ph.D. degree from the School of Cyber Engineering, Xidian University, Xi'an, in 2022.

She is currently a Lecture with the School of Mathematics, Hangzhou Normal University, Hangzhou, China. Her research interests include data security and applied cryptography.

Feng Li received the Ph.D. degree from the School of Cyber Engineering, Xidian University, Xi'an, China, in 2024.

He is currently a Postdoctoral Fellow with Singapore Management University, Singapore. His research interests include information security and applied cryptography.

Fuqun Wang received the B.S. degree in information and computing science and the M.S. degree in pure mathematics from Zhengzhou University, Zhengzhou, Henan, China, in 2003 and 2006, respectively, and the Ph.D. 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, Hangzhou, China. His research interests include fully homomorphic cryptography, lattices-based cryptography, and public key cryptography.

Zhiqian Liu received the B.S. degree from the School of Science, Xidian University, Xi'an, China, in 2012, and the Ph.D. degree from the School of Computer Science and Technology, Xidian University, Xi'an, in 2017.

He is currently a Full Professor with the College of Cyber Security, Jinan University, Guangzhou, China. His current research focuses on security, trust, privacy, and intelligence in vehicular networks.

Prof. Liu currently serves as the Area Editor, an Associate Editor, or an Academic Editor for more than 10 SCI-index journals, such as IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE INTERNET OF THINGS JOURNAL, *Information Fusion*, IEEE NETWORK, *Computer Networks*, and *Journal of Systems Architecture*. His homepage is <https://www.zqliu.com>.

Yan Zhu is currently pursuing the B.E. degree with the School of Mathematics, Hangzhou Normal University, Hangzhou, China.

Her research interests include data privacy and operations research.

Qi Xie received the Ph.D. degree in applied mathematics from Zhejiang University, Hangzhou, China, in 2005.

He was a Visiting Scholar with the Department of Computer Science, University of Birmingham, Birmingham, U.K., from 2009 to 2010, and a Visiting Scholar with the Department of Computer Science, The City University of Hong Kong, Hong Kong, in 2012. He is a Professor with Hangzhou Normal University, Hangzhou, and the Director of the Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University. He has published over 100 research papers in international journals and conferences, such as IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE INTERNET OF THINGS JOURNAL. His research interests include applied cryptography, including digital signatures, authentication, and key agreement protocols.

Prof. Xie served as a reviewer for over 40 international journals and served as the General Co-Chair for ISPEC2012 and ACM ASIACCS2013.

Song Han (Member, IEEE) received the Ph.D. degree in mathematics from Peking University, Beijing, China.

He is currently a Full Professor with the School of Computer and Computing Science, Hangzhou City University, Hangzhou, China. He is also with Zhejiang Ponshine Information Technology Company Ltd., Hangzhou. His current research interests include data security, privacy protection, IoT, artificial intelligence, and Industrial Internet of Things.

Prof. Han is a Qian Jiang Scholar Distinguished Professor with Zhejiang Province. His research has been published in international conferences and journals, including IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.