EAI Endorsed Transactions
on Security and Safety

Research Article **EAI.EU**

CPAR: Cloud-Assisted Privacy-preserving Image Annotation with Randomized k-d Forest

Yifan Tian[1], Jiawei Yuan[2,*], Yantian Hou[3]

[1]Agari Data, Inc., Foster City, CA 94404, USA
[2]University of Massachusetts Dartmouth, North Dartmouth, MA 02747, USA
[3]Boise State University, Boise, ID 83725, USA

Abstract

With the explosive growth in the number of pictures taken by smartphones, organizing and searching pictures has become important tasks. To efficiently fulfill these tasks, the key enabler is annotating images with proper keywords, with which keyword-based searching and organizing become available for images. Currently, smartphones usually synchronize photo albums with cloud storage platforms, and have their images annotated with the help of cloud computing. However, the "offloading-to-cloud" solution may cause privacy breach, since photos from smart photos contain various sensitive information. For privacy protection, existing research made effort to support cloud-based image annotation on encrypted images by utilizing cryptographic primitives. Nevertheless, for each annotation, it requires the cloud to perform linear checking on the large-scale encrypted dataset with high computational cost.

This paper proposes a cloud-assisted privacy-preserving image annotation with randomized k-d forest, namely CPAR. With CPAR, users are able to automatically assign keywords to their images by leveraging the power of cloud with privacy protected. CPAR proposes a novel privacy-preserving randomized k-d forest structure, which significantly improves the annotation performance compared with existing research. Thorough analysis is carried out to demonstrate the security of CPAR. Experimental evaluation on the well-known IAPR TC-12 dataset validates the efficiency and effectiveness of CPAR.

Received on 26 November 2019; accepted on 20 April 2020; published on 21 April 2020

Keywords: Image Annotation, Privacy-preserving, Cloud Computing

doi:10.4108/eai.13-7-2018.163999

1. Introduction

The widespread use of smartphones causes photography boom in recent years. According to a recent report from Forever's Strategy & Business Development team [1], the number of photos taken by smartphone is estimated to be 8.8 trillion in 2018. To facilitate the storage of photos, majority of smartphones today are synchronizing their photo albums with cloud storage, such as Apple's iCloud, Samsung Cloud, and Google Photos. Besides the storage service, these cloud storage platforms also help annotate users' photos with proper keywords, which is the key enabler for users to perform popular keyword-based search and organization over their photos. Although the cloud storage offers a set of decent features, it also raises privacy concerns since many users' photos may contain sensitive information, such as personal identities, locations, and financial information [2]. To protect the privacy of photos, encrypting them with standard encryption algorithms, e.g., AES, is still the major approach for privacy protection in cloud storage [3, 4]. However, this kind of encryption also sacrifices many other attractive functionalities of cloud storage, especially for keyword-based search and management for imagery files.

In order to enable keyword-based search and management on encrypted data in cloud, keyword-based searchable encryption (SE) has been widely investigated in recent years [5–9]. An SE scheme typically provides

*Corresponding author. Email: jyuan@umassd.edu encrypted search indexes constructed based on proper keywords assigned to data files. With these encrypted indexes, the data owner can submit encrypted keyword-based search request to search their data over ciphertexts. Unfortunately, these SE schemes all assume that keywords are already available for files to be processed, which is hard to be true for photos taken by smartphones. Specifically, unlike text files that support automatic keyword extraction from their contents, keywords assignment for imagery files relies on manual description or automatic annotation based on a large-scale pre-annotated image dataset. From the perspective of user experience, manually annotating each image from users' devices is clearly an impractical choice. Meanwhile, automatic image annotation that involves large-scale image datasets is too resource-consuming to be developed on smartphones. Although currently several cloud storage platforms offer image annotation services [10, 11], these platforms require access to unencrypted images. Therefore, how to provide efficient and privacy-preserving automatic annotation for smartphones' photos becomes the foundation of SE schemes applications on smartphones. To address this problem, our preliminary research proposes a scheme called CAPIA [12]. By tailoring homomorphic encryption over vector space, CAPIA offloads the image annotation process to the public cloud in privacy-preserving manner. Nevertheless, for every single annotation request, CAPIA requires the linear processing of all encrypted records in a large-scale dataset, which hence becomes its performance bottleneck for practical usage.

This paper proposes a cloud-based privacy-preserving image annotation scheme using the power of cloud computing with significantly enhanced efficiency, namely CPAR. To turbocharge the annotation efficiency with privacy protected, CPAR designs a novel privacy-preserving randomized k-d forest structure. Specifically, CPAR first integrates operations for image annotation with the data search using randomized k-d forest [13]. Then, by proposing a set of privacy-preserving comparison schemes, CPAR enables the cloud server to perform image annotation directly over an encrypted randomized k-d forest structure. Compared with the existing solution - CAPIA, CPAR offers an adjustable speedup rate from 4× to 43.1× while achieving 97.7% to 80.3% accuracy of CAPIA. Our privacy-preserving randomized k-d forest design can also be used as independent tools for other related fields, especially for these requiring similarity measurement on encrypted data. Moreover, considering the same keyword may have different importance for the semantic description of different images, CPAR also proposes a privacy-preserving design for real-time keywords ranking. To evaluate CPAR, thorough security analysis and numerical analysis are carried out first. Then, we implement a prototype of CPAR and conduct an extensive experimental evaluation using the well-known IAPR TC-12 dataset [14]. Our evaluation results demonstrate the practical performance of CPAR in terms of efficiency and accuracy.

The rest of this paper is organized as follows: In Section 2: we present the system model and threat model of CPAR. Section 3 introduces backgrounds of automatic image annotation and technical preliminaries for CPAR. The detailed construction of CPAR is provided in Section 4. We analyze the security of CPAR in Section 5. Section 6 evaluates the performance of CPAR. We review and discuss related works in Section 7 and conclude this paper in Section 8.

2. Models

2.1. System Model

As shown in Fig.1, CAPR is composed of two entities: a *Cloud Server* and a *User*. The user stores his/her images on cloud, and the cloud helps the user to annotate his/her images without learning the contents and keywords of images. In CPAR, the user first performs a one-time system setup that constructs an encrypted randomized k-d forest with a pre-annotated image datasets. This encrypted randomized k-d forest is offloaded to the cloud server to assist future privacy-preserving image annotation. For resource-constrained mobile devices, this one-time setup process can be performed using desktops. Later on, when the user has a new image to annotate, he/she generates an encrypted request and sends it to the cloud. After processing the encrypted request, the cloud returns ciphertexts of top related keywords and auxiliary information to the user. Finally, the user decrypts all keywords and ranks them based on their real-time weights to select final keywords.

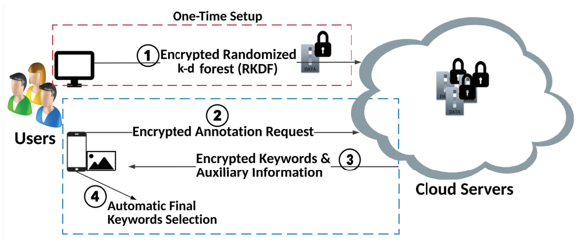

One-Time Setup
1 Encrypted Randomized k-d forest (RKDF)
DATA
Users
2 Encrypted Annotation Request
Encrypted Keywords & Auxiliary Information 3
4 Automatic Final Keywords Selection
Cloud Servers

Figure 1. System Model of CPAR

2.2. Threat Model

In CPAR, we consider the cloud server to be "curious-but-honest", i.e., the cloud server will follow our scheme to perform storage and annotation services correctly, but it may try to learn sensitive information in user's data. The cloud server has access to all encrypted images, encrypted image features, encrypted keywords, encrypted RKDF, the user's encrypted requests, and encrypted annotation results. We also assume the user's devices are fully trusted and will not be compromised. The research on protecting user devices is orthogonal to this work. These assumptions are consistent with major research works that focus on search over encrypted data on public cloud [7–9]. CPAR focuses on preventing the cloud server from learning following information: 1) contents of the user's images; 2) features extracted and keywords annotated for each image; 3) request linkability, i.e., tell whether multiple annotation requests are from the same image.

3. Preliminaries

3.1. Image Feature Extraction

In this paper, we adopt global low-level image features as that are utilized in the baseline image annotation technique [15], because it can be applied to general images without complex models and subsequent training. Color features of an image are extracted in three different color spaces: RGB, HSV, and LAB. In particular, RGB feature is computed as a normalized 3D histogram of RGB pixel, in which each channel (R,G,B) has 16 bins that divide the color space values from 0 to 255. The HSV and LAB features can be processed similarly as RGB, and thus we can construct three feature vectors for RGB, HSV and LAB respectively as $\mathbf{V}_{RGB}$, $\mathbf{V}_{HSV}$, and $\mathbf{V}_{LAB}$. Texture features of an image are extracted using Gabor and Haar wavelets. Specifically, an image is first filtered with Gabor wavelets at three scales and four orientations, resulting in twelve response images. Each response image is then divided into non-overlapping rectangle blocks. Finally, mean filter response magnitudes from each block over all response images are concatenated into a feature vector, denoted as $\mathbf{V}_G$. Meanwhile, a quantized Gabor feature of an image is generated using the mean Gabor response phase angle in non-overlapping blocks in each response image. These quantized values are concatenated into a feature vector, denoted as $\mathbf{V}_{GQ}$. The Haar feature of an image is extracted similarly as Gabor, but based on differently configured Haar wavelets. HaarQ stands for the quantized version of Haar feature, which quantizes Haar features into [0,-1,1] if the signs of Haar response values are zero, negative, and positive respectively. We denote feature vectors of Haar and HaarQ as $\mathbf{V}_H$ and $\mathbf{V}_{HQ}$ respectively. Therefore, given an image, seven feature vectors will be extracted as $[\mathbf{V}_{RGB}, \mathbf{V}_{HSV}, \mathbf{V}_{LAB}, \mathbf{V}_G, \mathbf{V}_{GQ}, \mathbf{V}_H, \mathbf{V}_{HQ}]$. For more details about the adopted image feature extraction, please refer to ref [15].

3.2. Integer Vector Encryption (IVE)

In this section, we describe a homomorphic encryption scheme designed for integer vectors [16], which will be tailored in our construction to achieve privacy-preserving image annotation. For expression simplicity, following definitions will be used in the rest of this paper:

- For a vector $\mathbf{V}$ (or a matrix $\mathbf{M}$), define $|max(\mathbf{V})|$ (or $|max(\mathbf{M})|$) to be the maximum absolute value of its elements.
- For $a \in \mathbb{R}$, define $\lceil a \rfloor$ to be the nearest integer of a, $\lceil a \rfloor_q$ to be the nearest integer of a with modulus q.
- For matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$, define $vec(\mathbf{M})$ to be a nm-dimensional vector by concatenating the transpose of each column of $\mathbf{M}$.

Encryption: Given a m-dimensional vector $\mathbf{V} \in \mathbb{Z}_p^m$ and the secret key matrix $\mathbf{S} \in \mathbb{Z}_q^{m\times m}$, output the ciphertext of $\mathbf{V}$ as

$$\mathbf{C}(\mathbf{V}) = \mathbf{S}^{-1}(w\mathbf{V} + \mathbf{e})^T \qquad (1)$$

where $\mathbf{S}^{-1}$ is the inverse matrix of $\mathbf{S}$, T is the transpose operator, $\mathbf{e}$ is a random error vector, w is an integer parameter, $q >> p$, $w > 2|max(\mathbf{e})|$.
Decryption: Given the ciphertext $\mathbf{C}(\mathbf{V})$, it can be decrypted using $\mathbf{S}$ and w as $\mathbf{V} = \lceil \frac{(\mathbf{SC}(\mathbf{V}))^T}{w} \rfloor_q$.
Inner Product: Given two ciphertexts $\mathbf{C}(\mathbf{V}_1), \mathbf{C}(\mathbf{V}_2)$ of $\mathbf{V}_1, \mathbf{V}_2$, and their corresponding secret keys $\mathbf{S}_1$ and $\mathbf{S}_2$, the inner product operation of $\mathbf{V}_1$ and $\mathbf{V}_2$ over ciphertexts can be performed as

$$vec(\mathbf{S}_1^T\mathbf{S}_2)\lceil \frac{vec(\mathbf{C}(\mathbf{V}_1)\mathbf{C}(\mathbf{V}_2)^T)}{w} \rfloor_q = w\mathbf{V}_1\mathbf{V}_2^T + \mathbf{e} \qquad (2)$$

To this end, $vec(\mathbf{S}_1^T\mathbf{S}_2)$ becomes the new secret key and $\lceil \frac{vec(\mathbf{C}(\mathbf{V}_1)\mathbf{C}(\mathbf{V}_2)^T)}{w} \rfloor_q$ becomes the new ciphertext of $\mathbf{V}_1\mathbf{V}_2^T$.

More details about this IVE encryption algorithm and its security proof are available in ref [16].

3.3. Order-Preserving Encryption (OPE)

Order-preserving symmetric encryption (OPE) is a deterministic encryption scheme whose encryption function preserves numerical ordering of the plaintexts. Given two integers a and b in which $a < b$, by encrypting with OPE, the order of a and b is preserved as $OPE(a) < OPE(b)$. More details about this OPE encryption scheme and its security proof are available in ref [17, 18].

4. Construction of CPAR

4.1. Scheme Overview

The core idea of automatic image annotation is built on the hypothesis that images contain similar objects are likely to share keywords. The distance between the feature vectors of two images is used to measure the probability that they contain similar objects [15]. Given a large-scale pre-annotated image dataset, the annotation process for a new image can be treated as a process of finding a set of images with shared objects and transferring keywords from those images. As a result, the annotation efficiency becomes heavily dependent on the performance of finding the image with shared objects. To boost the search efficiency, CPAR adopts randomized k-d forest as the searching index [13]. In addition, novel privacy-preserving schemes are designed to address the privacy concerns when integrating the randomized k-d forest into CPAR. Different from many other index structures that are only efficient for low-dimensional data, Randomized k-d forest (RKDF) is featured by its performance in handling high-dimensional data. In CPAR, data vectors are over 1300-dimension and thus making RKDF an effective selection.

As depicted in Figure 2, a RKDF is composed of a set of parallel k-d trees. For each $Node_i$ in a k-d tree [19], it stores a feature vector $\mathbf{V}_i$ of dataset image I_i. In addition, each non-leaf node also stores a *split* field s_i to generate a hyperplane that divides the vector space into two parts. Each $Node_j$ in left sub-tree of $Node_i$ has $Node_j[s_i] \le Node_i[s_i]$ and vice versa, as described in ref [19]. To search nodes that store vectors with top-smallest distances to a request vector $\mathbf{V}_{req}$, a parallel search among all trees in the forest is performed. Specifically, each tree is traversed in a top-down manner by comparing the *split* field values of $\mathbf{V}_{req}$ and the vector $\mathbf{V}_i$ stored in each $Node_i$ as an example shown in Fig.2(a). The traversal selects the left branch to continue if $\mathbf{V}_{req}[s_i] \le \mathbf{V}_i[s_i]$ and vice versa. Once the traversal reaches a leaf node, the vector stored in that leaf node is pushed into a priority queue $Queue$ as a current close candidate to $\mathbf{V}_{req}$. Note that during the search process, this $Queue$ keeps updating to hold L closest vectors to $\mathbf{V}_{req}$ and is shared by all trees in the forest. After that, a back trace search starts by iterating all the nodes in the path from the parent of the current node to the root node as an example shown in Fig.2(b). When reaching a $Node_i$ during the back trace, a same queue push is executed to judge whether to add $Node_i$ to $Queue$ as illustrated in Algorithm 1. For each $Node_i$ in this path, a distance comparison between $Dis(\mathbf{V}_{req}, \mathbf{H}_i)$ and $Dis(\mathbf{V}_{req}, \mathbf{V}_{qL})$ is compared, where $Dis(\mathbf{V}_{req}, \mathbf{H}_i)$ is the distance between $\mathbf{V}_{req}$ and a $Node_i$'s hyperplane. $\mathbf{H}_i$ can be considered as the projection vector of $\mathbf{V}_{req}$ on $Node_i$'s hyperplane. $\mathbf{V}_{qL}$ is the Lth vector in $Queue$ which meets $Dis(\mathbf{V}_{req}, \mathbf{V}_{qi}) \le Dis(\mathbf{V}_{req}, \mathbf{V}_{qL}), \forall \mathbf{V}_{qi} \in Queue$. If $Dis(\mathbf{V}_{req}, \mathbf{H}_i) > Dis(\mathbf{V}_{req}, \mathbf{V}_{qL})$, the back trace continues to the next node in this path. Otherwise, the sibling branch of $Node_i$ needs to be searched using the top-down traversal. In RKDF, once a node has been searched in one k-d tree, it will be marked and does not need to be checked again in the other trees. To further enhance the search efficiency of a RKDF, approximated search strategy can be adopted. In particular, based on the hypothesis that feature vectors of similar images are likely to be grouped in the same branch, there is a high probability that the targeted optimal top similar vectors will be visited well before visiting all nodes in each k-d tree. In Section 6, we will evaluate the relationship among the approximation strength, accuracy, and efficiency. The detailed search of a RKDF is provided in Algorithm 1. For more details about the RKDF, please refer to ref [13].

To protect the privacy of user's data during the cloud-based annotation, the image data associated with the RKDF need to be encrypted. Furthermore, these encrypted data shall support corresponding search operations in RKDF, which include:

- The comparison between $\mathbf{V}_{req}[s_i]$ and $\mathbf{V}_i[s_i]$ in the top-down traversal for path selection.
- The comparison between $Dis(\mathbf{V}_{req}, \mathbf{H}_i)$ and $Dis(\mathbf{V}_{req}, \mathbf{V}_{qL})$ during the back trace process.
- The comparison between $Dis(\mathbf{V}_{req}, \mathbf{V}_a)$ and $Dis(\mathbf{V}_{req}, \mathbf{V}_b)$, i.e., distances from the request vector to two different images' feature vectors, which is used in the queue push process.

The distance $Dis(\cdot)$ between two vectors is calculated with a combination of L_1 distance and KL-Divergence [15]. Specifically, the distance Dis_{ab} of two vectors is computed as

$$\begin{aligned} Dis_{ab} =& DL1_{ab}^{RGB} + DL1_{ab}^{HSV} + DL1_{ab}^{G} + DL1_{ab}^{GQ} \\ &+ DL1_{ab}^{H} + DL1_{ab}^{HQ} + DKL_{ab}^{LAB} \end{aligned}$$

where each vector has seven low-level color and texture feature vectors as discussed in Section 3.1, and $DL1$ and DKL denote L_1 distance and KL-Divergence of two vectors after data normalization.

In order to address the privacy challenges while utilizing RKDF for cloud-assisted automatic image annotation, a challenge needs to be resolved: The original privacy-preserving comparison scheme for L_1 distance ($PL1C$) and KL-Divergence ($PKLC$) in CAPIA cannot be simply re-used in CPAR. That's because $PL1C$ and $PKLC$ can only support the privacy-preserving distance comparison between two vectors. However, while searching in a RKDF, the distance comparison between a vector and a hyperplane needs to be supported in the back trace process and queue push process of RKDF. In order to resolve this issue, we re-design $PL1C$ and $PKLC$ to get $PL1C-RF$ and $PKLC-$

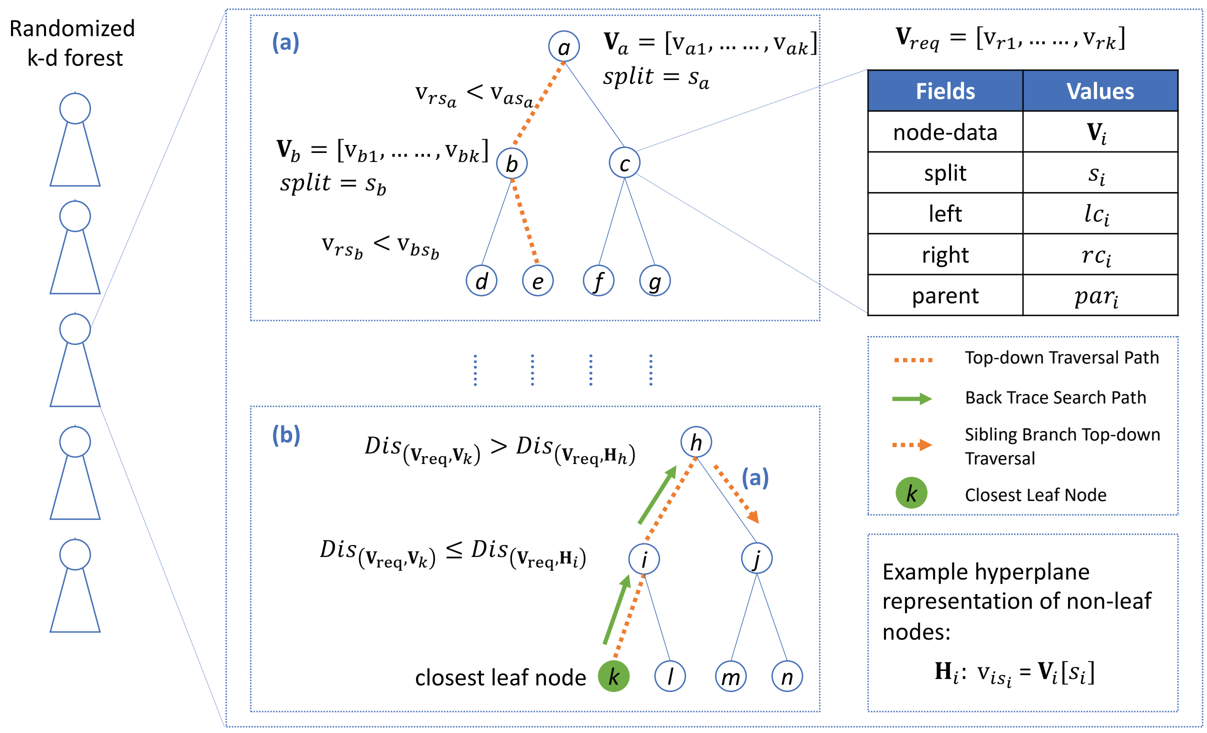

Randomized k-d forest
(a)
$\mathbf{V}_a = [v_{a1}, \dots\dots, v_{ak}]$
$split = s_a$
$v_{rs_a} < v_{as_a}$
$\mathbf{V}_b = [v_{b1}, \dots\dots, v_{bk}]$
$split = s_b$
$v_{rs_b} < v_{bs_b}$
a b c d e f g
$\mathbf{V}_{req} = [v_{r1}, \dots\dots, v_{rk}]$
Fields | Values
node-data | $\mathbf{V}_i$
split | s_i
left | lc_i
right | rc_i
parent | par_i
(b)
$Dis_{(\mathbf{V}_{req}, \mathbf{V}_k)} > Dis_{(\mathbf{V}_{req}, \mathbf{H}_h)}$
$Dis_{(\mathbf{V}_{req}, \mathbf{V}_k)} \leq Dis_{(\mathbf{V}_{req}, \mathbf{H}_i)}$
(a)
h i j k l m n
closest leaf node
Top-down Traversal Path
Back Trace Search Path
Sibling Branch Top-down Traversal
Closest Leaf Node
Example hyperplane representation of non-leaf nodes:
$\mathbf{H}_i$: $v_{is_i} = \mathbf{V}_i[s_i]$

Figure 2. $\mathbf{V}_{req}$ is the request vector and each $\mathbf{V}_i$ is stored in each tree node i. $Dis(\cdot)$ is an arbitrary distance calculation function and $Dis(\mathbf{V}_{req}, \mathbf{H}_i)$ is the distance between the request vector $\mathbf{V}_{req}$ and $Node_i$'s hyperplane. $\mathbf{V}_{qL}$ is the least closest vector to $\mathbf{V}_{req}$ in priority queue $Queue$. (a) represents top-down traversal and (b) represents back trace search.

RF, standing for $PL1C$ and $PKLC$ for RKDF. $PL1C - RF$ and $PKLC - RF$ enable the aforementioned privacy-preserving distance comparison between two vectors as well as between one vector and one hyperplane. In addition, we integrate order-preserving encryption [17, 18] into CPAR to protect the comparison of *split* field values in the top-down traversal of RKDF.

4.2. PL1C-RF: Privacy-preserving L_1 Distance Comparison for Randomized k-d Forest

In $PL1C - RF$, we consider two types of L_1 distance comparison that are required in the queue push and back trace process of RKDF: 1) $DL1_{ac}$ and $DL1_{bc}$ for three image feature vectors $\mathbf{V}_i, i \in \{a, b, c\}$; 2) $DL1_{hc}$ and $DL1_{bc}$ for a hyperplane projected vector $\mathbf{H}_a$ and two image feature vectors $\mathbf{V}_b, \mathbf{V}_c$. $DL1_{hc}$ is measured by the L_1 distance between $\mathbf{H}_a[s_a]$ and $\mathbf{V}_c[s_a]$, where s_a is the *split* field of the $Node_a$. To be more specific, $DL1_{hc}$ is calculated by projecting $\mathbf{V}_c$ on $Node_a$'s hyperplane and then calculating the L_1 distance between $\mathbf{V}_c$ and the projected vector $\mathbf{H}_a$.

Data Preparation: Given an image feature vector $\mathbf{V}_i = [v_{i1}, \cdots, v_{im}]$, the user first converts it to a $m\beta$-dimensional binary vector $\tilde{\mathbf{V}}_i = [F(v_{i1}), \cdots, F(v_{im})]$, where $\beta = |qL(\mathbf{V}_i)|$, and $F(v_{ij}) = [1, 1, \cdots, 1, 0, \cdots, 0]$ such that the first v_{ij} terms are 1 and the rest $\beta - v_{ij}$ terms are 0. The L_1 distance between $\mathbf{V}_a$ and $\mathbf{V}_b$ now can be calculated as

$$DL1_{ab} = \sum_{j=1}^{m} |v_{aj} - v_{bj}| = \sum_{j=1}^{m\beta} (\tilde{v}_{aj} - \tilde{v}_{bj})^2$$

Then, the approximation introduced in ref [20] is applied to $\tilde{\mathbf{V}}_i$ to update its dimension from $m\beta$ to $\hat{m} = \alpha m \log_{\gamma}^{\beta+1}$ based on the Johnson Lindenstrauss (JL) Lemma [21]. By denoting the approximated vector as $\hat{\mathbf{V}}_i$, we have $DL1_{ab} = \sum_{j=1}^{m\beta} (\tilde{v}_{aj} - \tilde{v}_{bj})^2 \approx \sum_{j=1}^{\hat{m}} (\hat{v}_{aj} - \hat{v}_{bj})^2$. The correctness and accuracy of such an approximation have been proved in ref [20]. According to our experimental evaluation in Section 6, we sets $\alpha = 1$ and $\gamma = 100$ in CPAR to balance accuracy and efficiency.

The detailed construction of the rest stages in *PL1C-RF* is presented in Fig.3. The user first encrypts the image feature vectors and its corresponding hyperplane projected vector (if exists), and then stores them in the cloud. Later on the user can generate encrypted L_1 distance comparison request and ask the cloud to conduct privacy-preserving comparison. On receiving the request, the cloud can conduct two types of L_1 distance comparison using ciphertext only according to user's request.

It is worth to note that $PL1C - RF$ is only interested in which distance is smaller during the comparison. Therefore, instead of letting the cloud get exact L_1 distances for comparison, $PL1C - RF$ adopts

Construction of PL1C-RF

Data Encryption:

1. Append 3 elements to an approximated $\hat{\mathbf{V}}_i$ as $\hat{\mathbf{V}}_i = [\hat{v}_{i1}, \hat{v}_{i2}, \cdots, \hat{v}_{i\hat{m}}, r - \frac{1}{2}\sum_{j=1}^{\hat{m}} \hat{v}_{ij}^2, \epsilon_i, -1], i \in \{a, b\}$, where r is a random number and ϵ_i is a small random noise.
2. If $\hat{\mathbf{V}}_i$ is stored in a non-leaf node, generate a $(2\hat{m}+2)$-dimensional hyperplane projected vector as $\hat{\mathbf{H}}_i = [0, \cdots, \hat{v}_{is_i}, \cdots, 0, r - \frac{1}{2}\hat{v}_{is_i}^2, \epsilon_i^{'}, 0, \cdots, 0 - 1, 0 \cdots, 0]$, where $r - \frac{1}{2}\hat{v}_{is_i}^2$ is the $(\hat{m}+1)$th element, -1 is the $(\hat{m}+2+s_i)$th element, and s_i is the *split* field of node i.
3. Encrypt $\hat{\mathbf{V}}_i$ and $\hat{\mathbf{H}}_i$ using the **Encryption** algorithm of IVE as $\mathbf{C}(\hat{\mathbf{V}}_i) = \mathbf{S}^{-1}(w\hat{\mathbf{V}}_i + \mathbf{e}_i)^T$ and $\mathbf{C}(\hat{\mathbf{H}}_i) = \mathbf{S}^{'-1}(w\hat{\mathbf{H}}_i + \mathbf{e}_i^{'})^T$. $\mathbf{C}(\hat{\mathbf{V}}_i), \mathbf{C}(\hat{\mathbf{H}}_i)$, and w are outsourced to the cloud.

Request Generation:

1. Append approximated request vector $\hat{\mathbf{V}}_c$ as $\hat{\mathbf{V}}_c = [r_c\hat{v}_{c1}, \cdots, r_c\hat{v}_{c\hat{m}}, r_c, 1, \frac{1}{2}r_c\sum_{j=1}^{\hat{m}} \hat{v}_{cj}^2]$, in which r_c is a positive random number.
2. Generate $\hat{\mathbf{H}}_c = [r_c\hat{v}_{c1}, \cdots, r_c\hat{v}_{c\hat{m}}, r_c, 1, \frac{1}{2}r_c\hat{v}_{c1}^2, \cdots, \frac{1}{2}r_c\hat{v}_{c\hat{m}}^2]$ as hyperplane projected vector.
3. $\hat{\mathbf{V}}_c$ and $\hat{\mathbf{H}}_c$ are encrypted as $\mathbf{C}(\hat{\mathbf{V}}_c) = \mathbf{S}_c^{-1}(w\hat{\mathbf{V}}_c + \mathbf{e}_c)^T$ and $\mathbf{C}(\hat{\mathbf{H}}_c) = \mathbf{S}_c^{'-1}(w\hat{\mathbf{H}}_c + \mathbf{e}_c^{'})^T$. $\mathbf{C}(\hat{\mathbf{V}}_c), \mathbf{C}(\hat{\mathbf{H}}_c), \mathbf{S}^T\mathbf{S}_c$ and $\mathbf{S}^{'T}\mathbf{S}_c^{'}$ are sent to the cloud as request.

Distance Comparison:
Type-1: Compare $DL1_{ac}, DL1_{bc}$

1. Given $\mathbf{C}(\hat{\mathbf{V}}_a)$, $\mathbf{C}(\hat{\mathbf{V}}_b)$ and $\mathbf{C}(\hat{\mathbf{V}}_c)$, compute $\lceil\frac{vec(\mathbf{C}(\hat{\mathbf{V}}_a)\mathbf{C}(\hat{\mathbf{V}}_c)^T)}{w}\rfloor_q$, $\lceil\frac{vec(\mathbf{C}(\hat{\mathbf{V}}_b)\mathbf{C}(\hat{\mathbf{V}}_c)^T)}{w}\rfloor_q$ and decrypt them as $\hat{\mathbf{V}}_a\hat{\mathbf{V}}_c^T$ and $\hat{\mathbf{V}}_b\hat{\mathbf{V}}_c^T$ as Eq.2.
2. Compare the approximated L_1 distance comparison as $\hat{\mathbf{V}}_b\hat{\mathbf{V}}_c^T - \hat{\mathbf{V}}_a\hat{\mathbf{V}}_c^T \approx \frac{r_c}{2}(DL1_{ac} - DL1_{bc}) + (\epsilon_b - \epsilon_a)$.

Type-2: Compare $DL1_{hc}, DL1_{bc}$,

1. Given $\mathbf{C}(\hat{\mathbf{H}}_a)$, $\mathbf{C}(\hat{\mathbf{V}}_b)$, $\mathbf{C}(\hat{\mathbf{V}}_c)$ and $\mathbf{C}(\hat{\mathbf{H}}_c)$, compute $\lceil\frac{vec(\mathbf{C}(\hat{\mathbf{H}}_a)\mathbf{C}(\hat{\mathbf{H}}_c)^T)}{w}\rfloor_q$, $\lceil\frac{vec(\mathbf{C}(\hat{\mathbf{V}}_b)\mathbf{C}(\hat{\mathbf{V}}_c)^T)}{w}\rfloor_q$ and decrypt them as $\hat{\mathbf{H}}_a\hat{\mathbf{H}}_c^T$ and $\hat{\mathbf{V}}_b\hat{\mathbf{V}}_c^T$ as Eq.2.
2. Compare the approximated L_1 distance comparison as $\hat{\mathbf{V}}_b\hat{\mathbf{V}}_c^T - \hat{\mathbf{H}}_a\hat{\mathbf{H}}_c^T \approx \frac{r_c}{2}(DL1_{hc} - DL1_{bc}) + (\epsilon_b - \epsilon_a^{'})$.

Figure 3. Construction of $PL1C - RF$

approximated distance comparison result scaled and obfuscated by r_c, $\epsilon_b - \epsilon_a$ and $\epsilon_b - \epsilon_a^{'}$ as shown in 4.2. As r_c is a positive random number, the sign of $\frac{r_c}{2}(DL1_{ac} - DL1_{bc})$ and $\frac{r_c}{2}(DL1_{hc} - DL1_{bc})$ are consistent with $DL1_{ac} - DL1_{bc}$ and $DL1_{hc} - DL1_{bc}$ respectively. Meanwhile, since $r_c >> \epsilon_b - \epsilon_a$ and $r_c >> \epsilon_b - \epsilon_a^{'}$, the added noise term has negligible influence to the sign of $DL1_{ac} - DL1_{bc}$ or $DL1_{hc} - DL1_{bc}$ unless these two distances are very close to each other. Fortunately, instead of finding the most related one, our CPAR design will utilize $PL1C - RF$ to figure out top 10 related candidates during the comparison. Such a design makes important candidates (say top 5 out of top 10) not be bypassed by the error introduced in $\epsilon_b - \epsilon_a$ and $\epsilon_b - \epsilon_a^{'}$. This hypothesis is further validated by our experimental results in Section 6.

4.3. PKLC-RF: Privacy-preserving KL-Divergence Comparison for Randomized k-d Forest

In $PKLC - RF$, we also consider two types of KL-Divergence comparison similar to $PL1C - RF$: 1) DKL_{ac} and DKL_{bc} for three image feature vectors $\mathbf{V}_i, i \in \{a, b, c\}$; 2) DKL_{hc} and DKL_{bc} for a hyperplane projected vector $\mathbf{H}_a$ and two image feature vectors $\mathbf{V}_b, \mathbf{V}_c$. Given two m-dimensional vectors $\mathbf{V}_i, i \in \{a, b\}$, their KL-Divergence DKL_{ab} is calculated as

$$\begin{aligned} DKL_{ab} &= \sum_{j=1}^{m} v_{aj} \times log(\frac{v_{aj}}{v_{bj}}) \\ &= \sum_{j=1}^{m} v_{aj} \times log(v_{aj}) - \sum_{j=1}^{m} v_{aj} \times log(v_{bj}) \end{aligned} \qquad (3)$$

where $log(\frac{v_{aj}}{v_{bj}}) = log(v_{aj}) = log(v_{bj}) = 0$ if $v_{aj} = 0$ or $v_{bj} = 0$. In addition, the KL-Divergence DKL_{hc} between a image feature vector and a hyperplane is measured by the KL-Divergence between $\mathbf{H}_a[s_a]$ and $\mathbf{V}_c[s_a]$, where s_a is the *split* field of $Node_a$. Similar with $PL1C - RF$, $PKLC - RF$ is also calculated by projecting $\mathbf{V}_c$ on $Node_a$'s hyperplane and then calculating the KL-Divergence between $\mathbf{V}_c$ and the projected vector $\mathbf{H}_a$.

The detailed construction of $PKLC - RF$ is presented in Fig.4. In the data encryption stage, the image feature vectors and corresponding hyperplane projected vector (if exists) are encrypted and stored in the cloud. On receiving the encrypted KL-Divergence comparison request from the user, the cloud conducts two types of privacy-preserving KL-Divergence comparison using ciphertext only according to user's request. Similar to our $PL1C$ construction, we have $r_c > 0$ and $r_c >> (\epsilon_b - \epsilon_a)$. Therefore, the cloud can figure out which KL-Divergence is smaller based on the scaled and obfuscated comparison result.

4.4. Detailed Construction of CPAR

CPAR consists of five major procedures. In the *System Setup*, the user selects system parameters, extracts, pre-processes feature vectors of images in a pre-annotated dataset and uses these feature vectors to build a RKDF. Then, the user executes the *RKDF Encryption* procedure to encrypt all data associated with nodes in the RKDF. Both the *System Setup* procedure and the *RKDF Encryption* procedure are one-time cost in CPAR.

Construction of PKLC-RF

Data Encryption:

1. Given an image feature vector $\mathbf{V}_i$, append $m+2$ elements as $\mathbf{V}_i = [v_{i1}, v_{i2}, \cdots, v_{im}, v_{i1} \times log(v_{i1}), \cdots, v_{im} \times log(v_{im}), r, \epsilon_i]$, where r is a random number and ϵ_i is a small random noise. If $\mathbf{V}_i$ is stored in a non-leaf node in RKDF, its corresponding hyperplane projected vector is processed as $\mathbf{H}_i = [0, \cdots, v_{is_i}, \cdots, 0, \cdots, v_{is_i} \times log(v_{is_i}), \cdots, 0, r, \epsilon_i^{'}]$, where s_i is the *split* field of the node, v_{is_i}, $v_{is_i} \times log(v_{is_i})$ and r are the s_ith, $(m+s_i)$th and $(2m+1)th$ elements respectively.
2. Encrypt $\mathbf{V}_i$ and $\mathbf{H}_i$ with the **Encryption** algorithm of IVE as $\mathbf{C}(\mathbf{V}_i) = \mathbf{S}^{-1}(w\mathbf{V}_i + \mathbf{e}_i)^T$ and $\mathbf{C}(\mathbf{H}_i) = \mathbf{S}^{-1}(w\mathbf{H}_i + \mathbf{e}_i^{'})^T$.

Request Generation:

1. Given request image feature vector $\mathbf{V}_c$, replace its elements v_{cj} with $-r_c \times log(v_{cj})$ and append $m+2$ elements to it as $\mathbf{V}_c = [-r_c \times log(v_{c1}), \cdots, -r_c \times log(v_{cm}), G(v_{c1}), \cdots, G(v_{cm}), r_c, -1]$, where $G(v_{cj}) = \begin{cases} r_c, v_{cj} \neq 0 \\ 0, v_{cj} = 0 \end{cases}$, r_c is a positive random number changing for every request.
2. Using the **Encryption** algorithm of IVE to encrypt $\mathbf{V}_c$ as $\mathbf{C}(\mathbf{V}_c) = \mathbf{S}_c^{-1}(w\mathbf{V}_c + \mathbf{e}_c)^T$. $\mathbf{C}(\mathbf{V}_c)$ and $\mathbf{S}^T\mathbf{S}_c$ are sent to the cloud as request.

KL-Divergence Comparison:
Type-1: Compare DKL_{ac}, DKL_{bc}

1. Compute $\lceil \frac{vec(\mathbf{C}(\mathbf{V}_a)\mathbf{C}(\mathbf{V}_c)^T)}{w} \rfloor_q, \lceil \frac{vec(\mathbf{C}(\mathbf{V}_b)\mathbf{C}(\mathbf{V}_c)^T)}{w} \rfloor_q$ and decrypts them as $\mathbf{V}_a\mathbf{V}_c^T$ and $\mathbf{V}_b\mathbf{V}_c^T$ using the **Decryption** of IVE in Section 3.2.
2. Compare KL divergence as $\mathbf{V}_a\mathbf{V}_c^T - \mathbf{V}_b\mathbf{V}_c^T = r_c(DKL_{ac} - DKL_{bc}) + (\epsilon_b - \epsilon_a)$.

Type-2: Compare DKL_{hc}, DKL_{bc}

1. Compute $\lceil \frac{vec(\mathbf{C}(\mathbf{H}_a)\mathbf{C}(\mathbf{V}_c)^T)}{w} \rfloor_q, \lceil \frac{vec(\mathbf{C}(\mathbf{V}_b)\mathbf{C}(\mathbf{V}_c)^T)}{w} \rfloor_q$ and decrypts as $\mathbf{H}_a\mathbf{V}_c^T$ and $\mathbf{V}_b\mathbf{V}_c^T$ using the **Decryption** of IVE as Eq.2.
2. Compare KL divergence as $\mathbf{H}_a\mathbf{V}_c^T - \mathbf{V}_b\mathbf{V}_c^T = r_c(DKL_{hc} - DKL_{bc}) + (\epsilon_b - \epsilon_a^{'})$.

Figure 4. Construction of PKLC-RF

Later on, the user can use the *Secure Annotation Request* procedure to generate an encrypted annotation request. On receiving the request, the cloud server performs the *Privacy-preserving Annotation on Cloud* procedure to return encrypted keywords for the requested image. At the end, the user obtains final keywords by executing the *Final Keyword Selection* procedure.

System Setup. To perform the one-time setup of CPAR system, the user first prepares a pre-annotated image dataset with n images, which can be obtained from public sources, such as IAPR TC-12 [14], LabelMe [22], etc. For each image I_i in the dataset, the user extracts seven feature vectors $[\mathbf{V}_{i,RGB}, \mathbf{V}_{i,HSV}, \mathbf{V}_{i,LAB}, \mathbf{V}_{i,G}, \mathbf{V}_{i,GQ}, \mathbf{V}_{i,H}, \mathbf{V}_{i,HQ}]$. Compared with other five feature vectors that have dimension up to 256, $\mathbf{V}_{i,H}$ and $\mathbf{V}_{i,HQ}$ have a high dimension as 4096. To guarantee the efficiency while processing feature vectors, Principal Component Analysis (PCA) [23] is utilized to reduce the dimension of $\mathbf{V}_{i,H}$ and $\mathbf{V}_{i,HQ}$. According to our experimental evaluation in Section 6.3, PCA-based dimension reduction with proper setting can significantly improve the efficiency of CPAR with slight accuracy loss. After that, L_1 normalization will be performed for each feature vector, which normalizes elements in these vectors to [-1,1]. Besides $\mathbf{V}_{i,LAB}$, the user also increases each element in $\mathbf{V}_{i,k}, k \in \{RGB, HSV, G, GQ, H, HQ\}$ as $v_{i,k,j} = v_{i,k,j} + 1$ to avoid negative values. Six feature vectors that use L_1 distance for similarity measurement are concatenated as a m_{L1}-dimensional vector $\mathbf{V}_{i,L1}$. $\mathbf{V}_{i,LAB}$ is denoted as a m_{KL}-dimensional vector $\mathbf{V}_{i,KL}$ for expression simplicity. It is easy to verify that $DL1_{ab}^{L1} = DL1_{ab}^{RGB} + DL1_{ab}^{HSV} + DL1_{ab}^{G} + DL1_{ab}^{GQ} + DL1_{ab}^{H} + DL1_{ab}^{HQ}$.

After that, the user constructs a RKDF with feature vector space $\{\mathbf{V}_i\}_{1 \leq i \leq n}$, in which each node in a single tree is associated with one $\mathbf{V}_i$. For each non-leaf node in RKDF, its *split* field element $\mathbf{V}_i[s_i]$ is stored in a set $\mathcal{SF}$. In CPAR, the RKDF contains ten parallel k-d trees.

RKDF Encryption. In this stage, the user is responsible for encrypting the constructed RKDF. Given an image I_i in the pre-annotated dataset, its keywords $\{K_{i,t}\}$ are first encrypted using AES by the user. Then, its processed feature vectors $\mathbf{V}_{i,L1}, \mathbf{V}_{i,KL}$ are encrypted with our $PL1C-RF$ and $PKLC-RF$ schemes as $\mathbf{C}(\mathbf{V}_{i,L1})$ and $\mathbf{C}(\mathbf{V}_{i,KL})$ respectively. $\mathbf{C}(\mathbf{V}_{i,L1})$ and $\mathbf{C}(\mathbf{V}_{i,KL})$ are then stored in the corresponding $Node_i$ of the RKDF. For each non-leaf node, encrypted hyperplane projected vectors $\mathbf{C}(\mathbf{H}_{i,L1}), \mathbf{C}(\mathbf{H}_{i,KL})$ are generated and added into $Node_i$ using the data encryption processes described in our $PL1C-RF$ and $PKLC-RF$. In addition, for the *split* field element $\mathbf{V}_i[s_i]$ of each non-leaf node, an order-preserving encryption is executed and the ciphertext $OPE(\mathbf{V}_i[s_i])$ is stored in $Node_i$. After the encryption, each node in the RKDF only contains encrypted data as

- **Non-leaf Node**: $[\mathbf{C}(\mathbf{V}_{i,L1}), \mathbf{C}(\mathbf{V}_{i,KL}), \mathbf{C}(\mathbf{H}_{i,L1}), \mathbf{C}(\mathbf{H}_{i,KL}), OPE(\mathbf{V}_i[s_i]), AES(\{K_{i,t}\})]$
- **Leaf Node**: $[\mathbf{C}(\mathbf{V}_{i,L1}), \mathbf{C}(\mathbf{V}_{i,KL}), AES(\{K_{i,t}\})]$

During the encryption process, same secret keys $\mathbf{S}_{L1}$, $\mathbf{S}_{L1}^{'}$, $\mathbf{S}_{KL}$, public parameter w, and random number r will be used for all images. However, different error vector $\mathbf{e}_i, \mathbf{e}_i^{'}$ and noise term $\epsilon_i, \epsilon_i^{'}$ are generated for each image I_i correspondingly. The user also computes

$\mathbf{S}_{L1}^T\mathbf{S}_{s,L1}$, $\mathbf{S}_{L1}^{'T}\mathbf{S}_{s,L1}^{'}$ and $\mathbf{S}_{KL}^T\mathbf{S}_{s,KL}$, in which $\mathbf{S}_{s,L1}$, $\mathbf{S}_{s,L1}^{'}$ and $\mathbf{S}_{s,KL}$ are secret keys for the encryption of later annotation requests. The encrypted RKDF, $\mathbf{S}_{L1}^T\mathbf{S}_{s,L1}$, $\mathbf{S}_{L1}^{'T}\mathbf{S}_{s,L1}^{'}$ and $\mathbf{S}_{KL}^T\mathbf{S}_{s,KL}$ are outsourced to the cloud.

Secure Annotation Request. When the user has a new image I_s for annotation, he/she first extracts seven feature vectors as $\mathbf{V}_s, s \in [RGB, HSV, LAB, G, GQ, H, HQ]$. These vectors will be processed to output $\mathbf{V}_{s,L1}$ and $\mathbf{V}_{s,KL}$ as that in the *System Setup* procedure. $\mathbf{V}_{s,L1}$ and $\mathbf{V}_{s,KL}$ are encrypted as $\mathbf{C}(\mathbf{V}_{s,L1})$, $\mathbf{C}(\mathbf{H}_{s,L1})$, and $\mathbf{C}(\mathbf{V}_{s,KL})$ using the *Request Generation* of $PL1C-RF$ and $PKLC-RF$ schemes respectively. For each annotation request, the user generates a new positive random number r_s and new error vectors $\mathbf{e}_s, \mathbf{e}_s^{'}$. Meanwhile, for each element sf_j in the *split* field set $\mathcal{SF}$ generated in *System Setup*, the user encrypts $\mathbf{V}_s[sf_j]$ using order-preserving encryption as $OPE(\mathbf{V}_s[sf_j])$. $\mathbf{C}(\mathbf{V}_{s,L1})$, $\mathbf{C}(\mathbf{H}_{s,L1})$, $\mathbf{C}(\mathbf{V}_{s,KL})$ and $\{OPE(\mathbf{V}_s[sf_j])\}$ are sent to the cloud as the annotation request.

Privacy-preserving Annotation on Cloud. On receiving the encrypted request, the cloud first performs a privacy-preserving search over the encrypted RKDF. As described in Algorithm 1, the cloud conducts parallel search over each encrypted tree in the RKDF. There are three places that require the cloud to conduct privacy-preserving computation over encrypted data:

• During the top-down traversal, as the *split* field element of each non-leaf node is encrypted using order-preserving encryption, the cloud can directly compare their ciphertexts (line 7) to determine which node to be checked next.

• In the back trace process, the cloud needs to perform privacy-preserving comparison to determine whether the current node's sibling branch needs to be searched (line 24 to 29). In particular, given $\mathbf{C}(\mathbf{V}_{s,L1})$, $\mathbf{C}(\mathbf{H}_{s,L1})$, $\mathbf{C}(\mathbf{V}_{qL,L1})$, $\mathbf{C}(\mathbf{H}_{parent,L1})$, $\mathbf{C}(\mathbf{V}_{s,KL})$, $\mathbf{C}(\mathbf{V}_{qL,KL})$, and $\mathbf{C}(\mathbf{H}_{parent,KL})$, the cloud first uses type-2 distance comparison in $PL1C-RF$ and $PKLC-RF$ to compute

$$\mathbf{V}_{qL,L1}\mathbf{V}_{s,L1}^T, \quad \mathbf{V}_{qL,KL}\mathbf{V}_{s,KL}^T,$$
$$\mathbf{H}_{parent,L1}\mathbf{H}_{s,L1}^T, \quad \mathbf{H}_{parent,KL}\mathbf{V}_{s,KL}^T$$

Then, the distance comparison is executed as

$$\begin{aligned} Comp_{qL} &= -2(\mathbf{V}_{qL,L1}\mathbf{V}_{s,L1}^T) + \mathbf{V}_{qL,KL}\mathbf{V}_{s,KL}^T \\ Comp_h &= -2(\mathbf{H}_{parent,L1}\mathbf{H}_{s,L1}^T) + \mathbf{H}_{parent,KL}\mathbf{V}_{s,KL}^T \end{aligned}$$

$$\begin{aligned} & Comp_{qL} - Comp_h \\ &= r_s(DL1_{qL,s}^{L1} - DL1_{parent,s}^{L1}) + 2(\epsilon_{parent}^{'} - \epsilon_{qL}) \\ &+r_s(DKL_{qL,s}^{LAB} - DKL_{parent,s}^{LAB}) + (\epsilon_{parent}^{'} - \epsilon_{qL}) \\ &= r_s(Dis(\mathbf{V}_{qL}, \mathbf{V}_s) - Dis(\mathbf{H}_{parent}, \mathbf{V}_s)) \\ &+3(\epsilon_{parent}^{'} - \epsilon_{qL}) \end{aligned} \tag{4}$$

where $\mathbf{V}_{qL}$ is the least closest vector to $\mathbf{V}_{req}$ in priority queue *Queue*. As r_s is a positive value and $r_s >> (\epsilon_{parent}^{'} - \epsilon_{qL})$, the sign of $Comp_{qL} - Comp_h$ is consistent with $Dis(\mathbf{V}_{qL}, \mathbf{V}_s) - Dis(\mathbf{H}_{parent}, \mathbf{V}_s)$.

• In the *Queue* push process (line 30-37), privacy-preserving distance comparison is needed to determine whether a new node shall be added. Specifically, given $\mathbf{C}(\mathbf{V}_{s,L1})$, $\mathbf{C}(\mathbf{V}_{Node,L1})$, $\mathbf{C}(\mathbf{V}_{qL}, L1)$, $\mathbf{C}(\mathbf{V}_{s,KL})$, $\mathbf{C}(\mathbf{V}_{Node,KL})$, $\mathbf{C}(\mathbf{V}_{qL}, KL)$, the cloud use type-1 distance comparison in $PL1C-RF$ and $PKLC-RF$ to perform distance comparison as

$$\begin{aligned} Comp_{Node} &= -2(\mathbf{V}_{Node,L1}\mathbf{V}_{s,L1}^T) + \mathbf{V}_{Node,KL}\mathbf{V}_{s,KL}^T \\ Comp_{qL} &= -2(\mathbf{V}_{qL,L1}\mathbf{V}_{qL,L1}^T) + \mathbf{V}_{cur,KL}\mathbf{V}_{s,KL}^T \end{aligned}$$

$$\begin{aligned} & Comp_{Node} - Comp_{qL} \\ &= r_s(DL1_{Node,s}^{L1} - DL1_{qL,s}^{L1}) + 2(\epsilon_{qL} - \epsilon_{Node}) \\ &+r_s(DKL_{Node,s}^{LAB} - DKL_{qL,s}^{LAB}) + (\epsilon_{qL} - \epsilon_{Node}) \\ &= r_s(Dis(\mathbf{V}_{Node}, \mathbf{V}_s) - Dis(\mathbf{V}_{qL}, \mathbf{V}_s)) \\ &+3(\epsilon_{qL} - \epsilon_{Node}) \end{aligned} \tag{5}$$

To this end, the cloud is able to perform all operations required by a RKDF search in the privacy-preserving manner, and obtain a *Queue* of nodes that stores data of top related images to the request. The cloud returns distance comparison candidates (type-1 distance) $Comp_i, i \in Queue$ as well as corresponding encrypted keywords back to the user.

Final Keyword Selection. The user first decrypts encrypted keywords and obtains $K_{i,t}, i \in Queue$, where $K_{i,t}$ is the t-th pre-annotated keyword in image I_i. Then, the user computes distances $Dis(\mathbf{V}_i, \mathbf{V}_s), i \in Queue$ as

$$\begin{aligned} Dis(\mathbf{V}_i, \mathbf{V}_s) &= (2r + \textstyle\sum_{j=1}^{m_{L1}} v_{s,L1,j}^2) + \frac{Comp_i}{r_s} \\ &= (2r + \textstyle\sum_{j=1}^{m_{L1}} v_{s,L1,j}^2) + \frac{-2(\mathbf{V}_{i,L1}\mathbf{V}_{s,L1}^T)+\mathbf{V}_{i,KL}\mathbf{V}_{s,KL}^T}{r_s} \end{aligned} \tag{6}$$

To achieve higher accuracy in keywords selection, we consider that keywords in images that have smaller distance to the requested one are more relevant. Thus, we define a real-time weight W_t for each keyword based on distances $Dis(\mathbf{V}_i, \mathbf{V}_s)$ as

$$W_{I_i} = 1 - \frac{Dis(\mathbf{V}_i, \mathbf{V}_s)}{\sum_{i \in RST} Dis(\mathbf{V}_i, \mathbf{V}_s)} \tag{7}$$

$$W_t = \sum W_{I_i}, \ for\ I_i\ contains\ K_{i,t} \tag{8}$$

Specifically, we first figure out the weight W_{I_i} of each image according to their distance-based similarity. As our definition in Eq.7, images with smaller distance will receive a larger weight value. Then, considering the same keyword can appear in multiple images, the final weight W_t of a keyword $K_{i,t}$ is generated by adding weights of images that contain this keyword. Finally, the user selects keywords for his/her image according to their ranking of weight W_t.

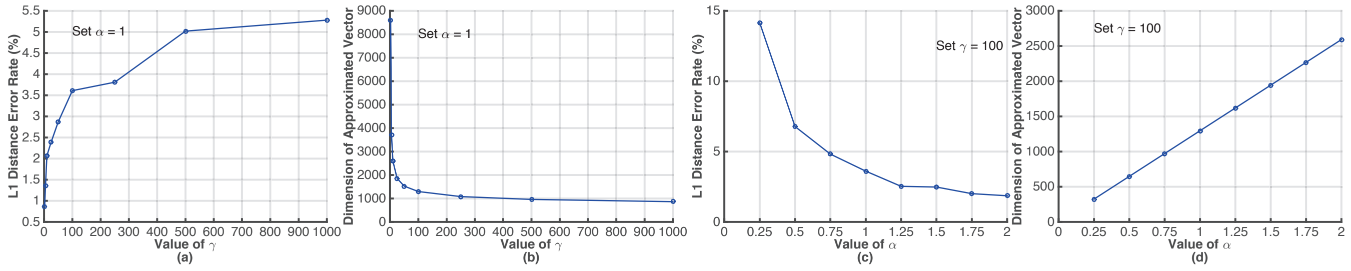


Figure 5. Error rate of Approximation and Dimension of Approximated Vector ($PCA - 32$)

Discussion - Personalization. In real-world use cases, for the purpose of enabling more accurate privacy-preserving cloud-assisted image annotation for personal imagery data, we suggest users to perform one additional initial preparation step. Instead of directly using well-known pre-labeled dataset to build RKDF, the user can select a few images from his/her own photo library, manually label them with personalized keywords and then add those images and keywords to the pre-labeled dataset. For example, user Lisa prefers to annotate her dog as "Mike" instead of "dog". Then during this initial step, Lisa only needs to add a few Mike's photos to the dataset and label them as "Mike" so that when there's a new photo of Mike waiting for annotation, our scheme could be able to target the existing photos of Mike in the dataset as similar images and thus picking the keyword "Mike" to appear in the final keyword selection. Note that this initial step only changes the size of the dataset and keyword set and would not affect the following RKDF building and searching process.

5. Security Analysis

In CPAR, we have the following privacy related data: feature vectors $\{\mathbf{V}_{i,L1}, \mathbf{V}_{i,KL}\}_{1\leq i\leq n}$, hyperplane projected vectors $\mathbf{H}_{i,L1}$, $\mathbf{H}_{i,KL}$ of each non-leaf node associated with $\mathbf{V}_{i,L1}$, $\mathbf{V}_{i,KL}$, the *split* field element of each non-leaf node, keywords of image I_i in the pre-annotated dataset, and feature vectors $\mathbf{V}_{s,L1}$, $\mathbf{H}_{s,L1}$, $\mathbf{V}_{s,KL}$ of the image requested for annotation. As keywords are encrypted using standard AES encryption, we consider them secure against the cloud server as well as outside adversaries. For the *split* field element of each non-leaf node, it is encrypted using the order-preserving encryption [17, 18], which has been proved to be secure. With regards to $\mathbf{V}_{i,L1}$, $\mathbf{H}_{i,L1}$, $\mathbf{V}_{i,KL}$, $\mathbf{H}_{i,KL}$, $\mathbf{V}_{s,L1}$, $\mathbf{H}_{s,L1}$ $\mathbf{V}_{s,KL}$, they are encrypted using the encryption scheme of IVE [16] after pre-processing as presented in our $PL1C-RF$ and $PKLC-RF$ schemes. The IVE scheme [16] has been proved to be secure based on the well-known Learning with Errors (LWE) hard problem [24]. Thus, given the ciphertexts $\mathbf{C}(\mathbf{V}_{i,L1})$, $\mathbf{C}(\mathbf{H}_{i,L1})$, $\mathbf{C}(\mathbf{V}_{i,KL})$, $\mathbf{C}(\mathbf{H}_{i,KL})$, $\mathbf{C}(\mathbf{V}_{s,L1})$, $\mathbf{C}(\mathbf{H}_{s,L1})$, $\mathbf{C}(\mathbf{V}_{s,KL})$ only, it is computational infeasible for the cloud server or outside adversaries to recover the corresponding feature vectors.

5.1. Security of Outsourcing $\mathbf{S}_{L1}^T\mathbf{S}_{s,L1}$, $\mathbf{S}_{L1}^{'T}\mathbf{S}_{s,L1}^{'}$ and $\mathbf{S}_{KL}^T\mathbf{S}_{s,KL}$

As $\mathbf{S}_{L1}^T\mathbf{S}_{s,L1}$, $\mathbf{S}_{L1}^{'T}\mathbf{S}_{s,L1}^{'}$, and $\mathbf{S}_{KL}^T\mathbf{S}_{s,KL}$ are used in the same manner, we use $\mathbf{S}^T\mathbf{S}_s$ to denote them for expression simplicity. Different from the original encryption algorithm of IVE, the user in CPAR also outsources $\mathbf{S}^T\mathbf{S}_s$ to the cloud besides ciphertexts. As all elements in $\mathbf{S}$ and $\mathbf{S}_s$ are randomly selected, elements in their multiplication $\mathbf{S}^T\mathbf{S}_s$ have the same distribution as these elements in $\mathbf{S}$ and $\mathbf{S}_s$ [25]. Thus, given $\mathbf{S}^T\mathbf{S}_s$, the cloud server is not able to extract $\mathbf{S}$ or $\mathbf{S}_s$ directly and use them to decrypt ciphertexts. By combining $\mathbf{S}^T\mathbf{S}_s$ with ciphertexts $\mathbf{C}(\mathbf{V}_{i,L1})$ and $\mathbf{C}(\mathbf{V}_{s,L1})$ (same as that for $\mathbf{C}(\mathbf{H}_{i,L1})$, $\mathbf{C}(\mathbf{H}_{s,L1})$, $\mathbf{C}(\mathbf{V}_{i,KL})$, $\mathbf{C}(\mathbf{H}_{i,KL})$ and $\mathbf{C}(\mathbf{V}_{s,KL})$), the cloud can obtain

$$\begin{aligned}
\mathbf{S}^T\mathbf{S}_s\mathbf{C}(\mathbf{V}_{i,L1}) &= \mathbf{S}^T\mathbf{S}_s\mathbf{S}^{-1}(w\mathbf{V}_{i,L1}+\mathbf{e}_i)^T \\
\mathbf{S}^T\mathbf{S}_s\mathbf{C}(\mathbf{V}_{s,L1}) &= \mathbf{S}^T\mathbf{S}_s\mathbf{S}_s^{-1}(w\mathbf{V}_{s,L1}+\mathbf{e}_s)^T \\
&= \mathbf{S}^T(w\mathbf{V}_{s,L1}+\mathbf{e}_s)^T
\end{aligned}$$

From the above two equations, it is clear that the combination of $\mathbf{S}^T\mathbf{S}_s$, $\mathbf{C}(\mathbf{V}_{i,L1})$ and $\mathbf{S}^T\mathbf{S}_s$, $\mathbf{C}(\mathbf{V}_{s,L1})$ only transfer them to the ciphertexts of $\mathbf{V}_{i,L1}$ and $\mathbf{V}_{s,L1}$ that encrypted using the IVE scheme with new keys $\mathbf{S}^T\mathbf{S}_s\mathbf{S}^{-1}$ and $\mathbf{S}^T$ respectively. As $\mathbf{S}^T\mathbf{S}_s\mathbf{S}^{-1}$ and $\mathbf{S}^T$ are random keys and unknown to the cloud, recovering $\mathbf{V}_{i,L1}$, $\mathbf{V}_{s,L1}$ from $\mathbf{S}^T\mathbf{S}_s\mathbf{C}(\mathbf{V}_{i,L1})$, $\mathbf{S}^T\mathbf{S}_s\mathbf{C}(\mathbf{V}_{s,L1})$ still become the LWE problem as proved in ref [16]. To this end, $\mathbf{S}^T\mathbf{S}_s$ only helps the cloud perform distance comparison in CPAR, but does not bring additional advantages to recover feature vectors compared with the given ciphertexts only scenario.

5.2. Request Unlinkability

The request unlinkability in CPAR is guaranteed by the randomization for each request. Specifically, each query request $\{\mathbf{V}_{s,L1}, \mathbf{H}_{s,L1}, \mathbf{V}_{s,KL}\}$ is element-wise obfuscated with different random error terms $\mathbf{e}_s$, $\mathbf{e}_s^{'}$ and random number r_s during the encryption, which makes the obfuscated $\mathbf{V}_{s,L1}, \mathbf{H}_{s,L1}, \mathbf{V}_{s,KL}$ have the same distribution

Algorithm 1: Privacy-preserving RKDF Search

```
Input  : Encrypted Search Request (Req) for $\mathbf{V}_s$,
         Encrypted RKDF with a set of Trees $\{T_k\}$,
         approximation power $\mathcal{AP}-\mathcal{X}$
Output: Encrypted Nodes Associated with Top Related
         Images to the Request.
1  Initialization $Queue$ = [], $Path$ = [] (Searched Path), $Vis$
     = [] (Visited Nodes), $Node_k$=$T_k$.root;
2  Each tree $T_k$ executes topDownTraversal() and
     backTraceSearch() in parallel, $Queue$ and $Vis$ are
     shared among all trees;
3  Function topDownTraversal($Req$, $Node_k$):
4      if $Node_k$ is not null then
5          return;
6      $\mathbf{V}_i \leftarrow Node_k.\mathbf{V}_i$;
7      if $OPE(\mathbf{V}_s[s_i]) \leq OPE(\mathbf{V}_i[s_i])$ then
8          $Node_k$ = topDownTraversal($Node_k$.left-child);
9      else
10         $Node_k$ =
             topDownTraversal($Node_k$.right-child);
11     if $Node_k \notin Vis$ then
12         $Vis.push(Node_k)$;
13         $Queue.push(Node_k)$;
14     $Path.push(Node_k)$;
15     return $Node_k$;
16 Function backTraceSearch($Req$, $Node_k$):
17     if $Vis.length() > \mathcal{AP}-\mathcal{X}\times$ Nodes Number then
18         return $Queue$;
19     if $Path$ is not null then
20         $parent \leftarrow Path.pop()$;
21     if $parent \notin Vis$ then
22         $Vis.push(parent)$;
23         $Queue.push(parent)$;
24     //Privacy-preserving distance comparison is
       achieved by $PL1C-RF$ and $PKLC-RF$, $\mathbf{V}_{qL}$ is the
       least closest vector to $\mathbf{V}_s$ in $Queue$
25     if $Dis(\mathbf{V}_{qL}, \mathbf{V}_s) < Dis(\mathbf{H}_{parent}, \mathbf{V}_s)$ then
26         backTraceSearch(Req, $parent$);
27     else
28         $Node_k$ =
             topDownTraversal($Req, Node_k.sibling$);
29     return $Queue$;
30 Function Queue.push($Node$):
31     //Each $Node_q$ in $Queue$ are ordered by
        $Dis(\mathbf{V}_s, \mathbf{V}_{Node_q})$
32     if $Queue.length()$ < Defined Size $L$ then
33         Add $Node$ into $Queue$ by order;
34     else
35         if $Node_{qL}$ in $Queue$ has
             $Dis(\mathbf{V}_s, \mathbf{V}_{Node}) < Dis(\mathbf{V}_s, \mathbf{V}_{qL})$ then
36             Remove $Node_{qL}$ from $Queue$;
37             Add $Node$ into $Queue$ by order;
```

as in these random values $\mathbf{e}_s$, $\mathbf{e}_s^{'}$ and r_c [25]. Thus, by changing $\mathbf{e}_s$, $\mathbf{e}_s^{'}$ and r_c during the encryption of different requests, CPAR outputs different random ciphertexts, even for requests generated from the same image.

6. Evaluation

To evaluate the performance of CPAR, we implemented a prototype using Python 2.7. In our implementation, Numpy [26] is used to support efficient multi-dimension array operations. OpenCV [27] is used to extract the color-space features of the images and build the filter kernels to generate the Gabor filter results. Pywt [28] is adopted to perform Haar wavelet and get the corresponding Haar results. Sklearn [29] is used to perform the PCA transformation. FLANN library [13] is used to act as the non-privacy randomized k-d forest for comparison. We use the well-known IAPR TC-12 [14] as the pre-annotated dataset, which contains 20,000 annotated images and the average number of keywords for each image is 5.7. All tests are performed on a 3.1 GHz Intel Core i7 Macbook Pro with OS X 10.14.2 installed as *User* and a Microsoft Azure cloud E4-v3 VM with Ubuntu 18.04 LTS installed as *Cloud Server*.

In the rest of this section, n is the total number of images in the pre-annotated dataset, m_{L1} is the dimension of pre-processed feature vectors $\mathbf{V}_{i,L1}$, m_{KL} is the dimension of pre-processed feature vectors $\mathbf{V}_{i,KL}$ and their corresponding hyperplane projected vectors $\mathbf{H}_{i,KL}$, $m_{L1}^{'}$ is the dimensions of hyperplane projected vector $\mathbf{H}_{i,L1}$. We also use DOT_m to denote a dot product operation between to two m-dimensional vectors. $\mathcal{AP}-X$ is used to denote the approximation power during the RKDF search, which indicates $X\%$ of the nodes will be checked in each tree of RKDF. $PCA-X$ is used to denote the strength of PCA transformation applied to $\mathbf{V}_{i,H}$ and $\mathbf{V}_{i,HQ}$ in $\mathbf{V}_{i,L1}$, which compresses their dimensions from 4096 to $\frac{4096}{X}$. $PCA-128$, $PCA-64$, $PCA-32$, $PCA-16$, and $PCA-8$ are evaluated in our experiments to balance the efficiency and accuracy of CAPIA.

In our evaluation, we first provide numerical analysis as well as experimental evaluation for each stage of CPAR. Then, we compare CPAR with CAPIA proposed in ref [12] in terms of efficiency and accuracy.

6.1. System Parameter Selection

To perform the one-time setup in CPAR, the user pre-processes feature vectors of each image in the pre-annotated image dataset. Specifically, the user first performs JL-Lemma based approximation over $\mathbf{V}_{i,L1}$ to make them compatible with our $PL1C-RF$. As discussed in Section 4.2, there is a trade-off between the approximation accuracy of L_1 distance and length of the approximated vector that determines efficiency of follow up privacy-preserving operations. To balance such a trade-off, we evaluate different parameters for approximation as shown in Fig.5 (a)-(d). According to our results, we suggest to set $\alpha = 1$ and $\gamma = 100$ which introduces 3.61% error rate for L_1 distance computation, and extends the dimension of $\mathbf{V}_{i,L1}$ from 864 to 1296 under the setting of $PCA-32$. Specifically, the error rate drops fast when $\alpha < 1$ and becomes relative stable when $\alpha > 1$. Meanwhile, the dimension of the approximated vector increases linearly to the value of α. With regards to γ, the dimension of the approximated vector becomes relative stable when $\gamma > 100$, however, the error rate still increases when $\gamma > 100$.

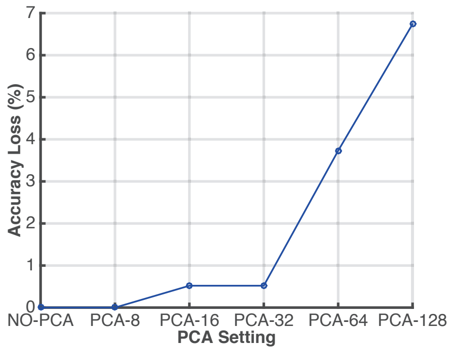


Figure 6. Accuracy Loss with Different PCA Settings

With regards to the selection of PCA parameter, it is clear that better efficiency of CPAR will be achieved by increasing the strength of PCA. However, the stronger PCA setting will also cause accuracy loss due to the loss of information during the compression. To balance the efficiency and accuracy, we evaluate of accuracy loss of annotation with different PCA setting. Compared with the $No-PCA$ setting, Fig.6 shows the accuracy loss for $PCA-8$, $PCA-16$, and $PCA-32$ are stable and bounded in 0.5%. Differently, $PCA-64$ and $PCA-128$ rapidly raise the accuracy loss. Therefore, $PCA-32$ is adopted by CPAR.

6.2. RKDF Construction and Encryption

To construct an encrypted RKDF, the user first constructs an unencrypted RKDF using 20,000 pre-annotated images, and then replaces data of each node in the RKDF with their corresponding ciphertexts. The construction of an unencrypted RKDF with 10 k-d trees costs 28.56 seconds. Then, for the pre-processed feature vectors $\mathbf{V}_{i,L1}$ and $\mathbf{V}_{i,KL}$ of each image, the user can encrypt them using $PL1C-RF$ and $PKLC-RF$ with $(m_{L1})DOT_{m_{L1}}$ and $(m_{KL})DOT_{m_{KL}}$ operations respectively, which costs 8.4ms in total in our implementation. If an image is associated with a non-leaf node in any tree of the RKDF, encryption for the hyperplane projected vectors $\mathbf{H}_{i,L1}$ and $\mathbf{H}_{i,KL}$ with $(m'_{L1})DOT_{m'_{L1}}$ and $(m_{KL})DOT_{m_{KL}}$ operations respectively, which costs 54.7ms in total. In addition, for each non-leaf node, an order-preserving encryption is needed for the *split* field, each of which costs 1.4ms. Therefore, to build a 10-tree encrypted RKDF with a 20,000 pre-annotated image dataset, it takes 74.78 minutes in our implementation. It is noteworthy that the encrypted RKDF construction is one-time offline cost, which does not impact the performance of later on real-time privacy-preserving image annotation.

6.3. Real-time Image Annotation

Request Generation: To annotate a new image in a privacy-preserving manner, the user pre-processes and encrypts its feature vectors $\mathbf{V}_{s,L1}$ and $\mathbf{V}_{s,KL}$ using $PL1C-RF$ and $PKLC-RF$. Specifically, the encryption of $\mathbf{V}_{s,L1}$ requires $(m_{L1})DOT_{m_{L1}} + (m'_{L1})DOT_{m'_{L1}}$ for shown in Fig.3, and the encryption of $\mathbf{V}_{s,KL}$ requires $(m_{KL})DOT_{m_{KL}}$ operations as shown in Fig.4. In addition, for each element sf_j in the *split* field element set $\mathcal{SF}$ with size of 348 in our implementation, order-preserving encryption are executed for $\mathbf{V}_s[sf_j]$. As a result, the encrypted request can be efficiently generated with only 534.16ms.

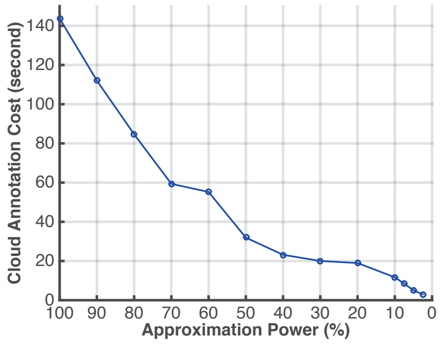


Figure 7. Privacy-preserving Annotation Cost on Cloud with Different Approximation Power

Privacy-preserving Annotation on Cloud: On receiving the encrypted request, the cloud performs privacy-preserving RKDF search with top-down traversal, back trace search, and queue push processes. The top-down traversal only requires a direct comparison between the ciphertexts under order-preserving encryption, whose cost is negligible compared with the other two processes. In the back trace search, privacy-preserving type-2 distance comparison needs to the executed using $PL1C-RF$ and $PKLC-RF$. In particular, two

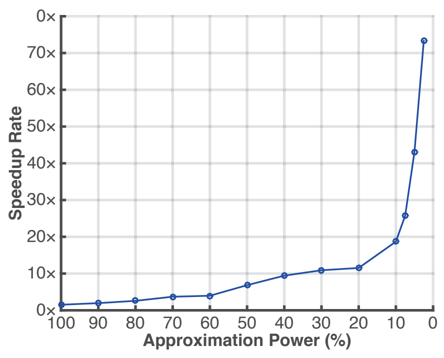


Figure 8. Speedup Rate with Different Approximation Power

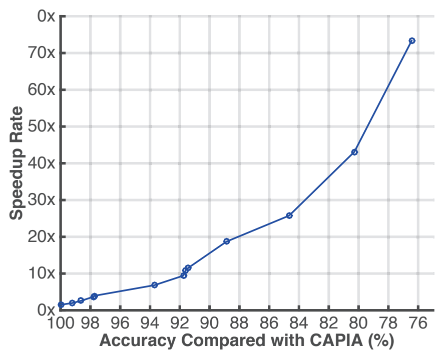


Figure 10. Speedup rate of CPAR with Different Accuracy Compared with CAPIA

comparison candidates $Comp_{qL}$ and $Comp_h$ are computed with $(m_{L1}+1)DOT_{m_{L1}} + (m_{KL}+1)DOT_{m_{KL}}$ operations and $(m'_{L1}+1)DOT_{m'_{L1}} + (m_{KL}+1)DOT_{m_{KL}}$ operations respectively. With regards to the queue push process, privacy-preserving type-1 distance comparison are executed using $PL1C-RF$ and $PKLC-RF$, which requires $2(m_{L1}+1)DOT_{m_{L1}} + 2(m_{KL}+1)DOT_{m_{KL}}$ operations in total. Another important parameter that affects the search efficiency is the selection of approximation power $\mathcal{AP}-\mathcal{X}$. As depicted in Fig.7, by increasing the approximation power from $\mathcal{AP}-100$ to $\mathcal{AP}$–2.5, the privacy-preserving annotation using encrypted RKDF reduces from 143.72 seconds to 2.98 seconds. Compared with CAPIA [12] that requires 218.46 seconds for one privacy-preserving annotation on cloud and does not support approximate dataset checking, CPAR can significantly speed it up as depicted in Fig.8.

Final Keyword Selection: This process only involves AES decryption and the weights generation that only requires a small number of additions. As a result, the final keyword selection can be completed by the user within 318ms.

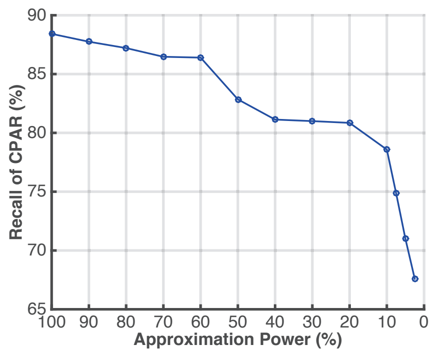


Figure 9. Accuracy (Recall) of CPAR with Different Approximation Power

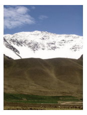
Accuracy: To evaluate the accuracy of CPAR, we use the standard average *recall* rates to measure the accuracy of keywords annotation. To be specific, by using $[K_1, K_2, \cdots, ...K_y]$ to denote distinct keywords annotated with CPAR for a set of image annotation requests, the recall rate for each keyword K_j and the average accuracy are defined as

- $recall_{K_j} = \frac{\#\ of\ images\ assigned\ K_j\ correctly\ by\ CPAR}{\#\ of\ images\ assigned\ K_j\ in\ the\ ground-truth}$
- $Accuracy = \frac{\sum_{j=1}^{y} recall_{K_j}}{y}$

In our evaluation, annotation requests for 50 different images are submitted, in which each requested image has two or more related images in the pre-annotated dataset. As shown in Fig.9, the accuracy of CPAR reduces from 88.42% to 67.59% when the approximation power increases from $\mathcal{AP}-100$ to $\mathcal{AP}$–2.5. Compared with CAPIA [12] our scheme achieves the same accuracy by setting the approximation power as $\mathcal{AP}-100$. While the increasing of approximation power reduces the accuracy of CPAR to some extent, it also boosts the efficiency significantly as shown in Fig.7. Compared with CAPIA, Fig.10 shows that CPAR can speed up CAPIA by 4×, 11.5×, 18.7×, 25.8×, 43.1× when achieving 97.7%, 91.4%, 88.9%, 84.7%, 80.3% accuracy of CAPIA respectively. Therefore, CPAR can greatly promote the efficiency the of CAPIA while retaining comparable accuracy. To balance the efficiency speedup and annotation accuracy of CPAR, we suggest to set the approximation power as $\mathcal{AP}-10$, i.e. achieves 88.9% accuracy of CAPIA with 18.7× speedup. Note that in real world applications, we can opt to launch our scheme for image annotaiton tasks during device's idle time, for instance, when device is charging during user's bed time.

In Table 1, we present samples of automatically annotated images using CAPIA and CPAR with approximation power as $\mathcal{AP}-10$. In the last column

Table 1. Sample Annotation Results

Image	CPAR Annotation	CAPIA Annotation	Human Annotation
	floor-tennis-court, **man**, grass	**floor-tennis-court**, **man**, woman	floor-tennis-court, man
	highway, sky-blue, **trees, vegetation**, ground, ship, sky, ocean, bush	**sky-blue, highway**, **vegetation**, ground, bush, **trees**, lake, ocean	highway, sky-blue, trees, vegetation
	group-of-persons, **ground, cloud**, man, sky-light, **mountain**, door, chair, floor-other, column	**cloud, sky-blue**, **ground, mountain**, horse man, road, **grass**	ground, cloud, sky-blue, mountain, snow, grass
	group-of-persons, hat, hill, cloud, **sky-blue**, **ground**, sky, fabric, couple-of-persons, grass	**group-of-persons**, **sky-blue, ground**, **trees**, mountain, ruin-archeological, hat, cloud, hill	trees, ground, man, sky-blue, group-of-persons

In each cell of CPAR and CAPIA annotation results, ground-truth human annotation results are underlined and bold out.

we list the human annotation results (ground-truth) for comparison. On one hand, CPAR is highly possible to assign correct keywords to images compared with human annotation. This observation also confirms the high average recall rate of CPAR, since these ground-truth annotations are likely to be covered in CPAR. On the other hand, CPAR also introduces additional keywords that frequently appear together with these accurate keywords in top related images. These additional keywords are typically not directly included in human annotations, but are potentially related to correct keywords. Compared with CAPIA, CPAR only misses a small portion of ground-truth keywords due to the approximation strategy, which is consistence with our evaluation result in Fig.9 and Fig.10. Overall, our evaluation results demonstrate that although CPAR cannot provide perfect keywords selection all the time compared with human annotation, it can still maintain comparable accuracy as CAPIA and is promising for automatically assigning keywords to images.

Communication Cost: The communication cost in CPAR comes from two major parts: annotation request and encrypted results returned from the cloud server. The encrypted request consists of a m_{L1}-dimensional vector $\mathbf{C}(\mathbf{V}_{s,L1})$, a m'_{L1}-dimensional vector $\mathbf{C}(\mathbf{H}_{s,L1})$, a m_{KL}-dimensional vector $\mathbf{C}(\mathbf{V}_{s,KL})$ and a set of encrypted *split* field elements $\mathcal{SF}$. In the $PCA-32$ setting, the total communication cost for a request is 80KB, in which 26KB for $\mathbf{C}(\mathbf{V}_s)$, 48KB for $\mathbf{C}(\mathbf{H}_s)$ and 4KB for $\mathcal{SF}$. Meanwhile, the returned results contain encrypted keywords and distance comparison candidates of top 10 related images. Using AES-256 for keywords encryption, the total size for the returned result is 488 Bytes with the average number of keywords for each pre-annotated image as 5.7. Therefore, the communication cost for each privacy-preserving annotation can be efficiently handled in today's Internet environment.

7. Related Works

To solve the problem of how to search over encrypted data, the idea of keyword-based searchable encryption (SE) was first introduced by Song et.al in ref [5]. Later on, with the widespread use of cloud storage services, the idea of SE received increasing attention from researchers. In ref [6, 7], search efficiency enhanced SE schemes are proposed based on novel index constructions. After that, SE schemes with the support of multiple keywords and conjunctive keywords are investigated in ref [8], and thus making the search more accurate and flexible. Recently, fuzzy keyword is considered in ref [9], which enables SE schemes to tolerate misspelled keyword during the search process. While these SE schemes offer decent features for keyword-based search, their application to images are limited given the question that how keywords of images can be efficiently extracted with privacy protection. It is impractical for cloud storage users to manually annotate their images.

To automate the keywords extraction process for images, a number of research works have been proposed with the concept of "automatic image annotation" [15, 30–32]. Chapelle et al. [33] trained support vector machine (SVM) classifiers to achieve high annotation accuracy where the only available image features are high dimensional histograms. In ref [34, 35], SVM was used to learn regional information as well as helped segmentation and classification process simultaneously. Different from SVM which works by finding a hyperplane to separate vector spaces, Bayesian network accomplishes the annotation tasks by modeling the conditional probabilities from training samples. In ref [36, 37], Bayesian networks were built by clustering global image features to calculate the conditional probabilities. Another widely used technique is artificial neural network (ANN). Take ref [38] as an instance, based on the assumption that after image segmentation, the largest part of an image significantly characterizes the entire image, Park et al. annotated images using a 3-layer ANN. With the flourish of deeper ANN structures, such as convolutional neural network (CNN), in various vision tasks [39–41], these deeper frameworks have also been applied to image annotation tasks. In ref [42], Yunchao et al. proposed to solve image annotation problem by training CNN with rankings. Jian et al. [43] combined CNN with recurrent neural network (RNN) to address the problem of the keyword dependency during annotation. However, all of these image annotation works raise privacy issues when delegated to the cloud since unencrypted images need to be outsourced. Therefore, to address such privacy concerns, this paper proposes CPAR, which utilizes the power of cloud computing to perform automatic image annotation for users, while only providing encrypted image information to the cloud.

8. Conclusion

In this paper, we propose CPAR that enables privacy-preserving image annotation using public cloud servers. CPAR uniquely integrates randomized k-d forest with a privacy-preserving design, and thus boosting the annotation efficiency using cloud. Specifically, CPAR proposes the lightweight privacy-preserving L_1 distance $PL1C-RF$ and KL-Divergence $PKLC-RF$ comparison schemes, and then utilizes them together with order-preserving encryption to support all required operations in image annotation and randomized k-d forest search. Our $PL1C-RF$, $PKLC-RF$ and privacy-preserving randomized k-d forest can also be utilized as independent tools for other related fields, especially for efficient similarity measurement on encrypted data. Thorough security analysis is provided to show that CPAR is secure in the defined threat model. Extensive numerical analysis as well as prototype implementation over the well-known IAPR TC-12 dataset demonstrate the practical performance of CPAR in terms of efficiency and accuracy.

References

[1] Oneslager, R. (2018), How Many Photos Were Taken Last Year?, `https://blog.forever.com/forever-blog/2018/1/22/how-many-photos-were-taken-last-year`.

[2] Privacyrights.org (2017), The Privacy Implications of Cloud Computing, `https://www.privacyrights.org/blog/privacy-implications-cloud-computing`.

[3] Boxcryptor, Encrypt your files in your dropbox, https://www.boxcryptor.com/en/dropbox.

[4] Amazon Simple Storage Service, Protecting data using encryption, http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html.

[5] Song, D.X., Wagner, D. and Perrig, A. (2000) Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00 (Washington, DC, USA: IEEE Computer Society): 44–55.

[6] Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G. (2004) Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004* (Springer Berlin Heidelberg), 506–522. doi:10.1007/978-3-540-24676-3_30.

[7] Wang, C., Cao, N., Li, J., Ren, K. and Lou, W. (2010) Secure ranked keyword search over encrypted cloud data. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, ICDCS '10 (Washington, DC, USA: IEEE Computer Society): 253–262. doi:10.1109/ICDCS.2010.34.

[8] Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T. and Li, H. (2013) Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13 (New York, NY, USA: ACM): 71–82. doi:10.1145/2484313.2484322.

[9] Wang, B., Yu, S., Lou, W. and Hou, Y.T. (2014) Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *INFOCOM, 2014 Proceedings IEEE*.

[10] Google (2016), Cloud Vision API, `https://cloud.google.com/vision/`.

[11] Scale, Scale Image Annotation API, `https://www.scaleapi.com/image-annotation`.

[12] Tian, Y., Hou, Y. and Yuan, J. (2017) Capia: Cloud assisted privacy-preserving image annotation. In *Communications and Network Security (CNS), 2017 IEEE Conference on* (IEEE): 1–9.

[13] Muja, M. and Lowe, D.G. (2014) Scalable nearest neighbor algorithms for high dimensional data. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence **36**(11): 2227–2240.

[14] Escalante, H.J., Hernández, C.A., Gonzalez, J.A., López-López, A., Montes, M., Morales, E.F., Enrique Sucar, L. *et al.* (2010) The segmented and annotated iapr tc-12 benchmark. *Comput. Vis. Image Underst.* **114**(4): 419–428. doi: 10.1016/j.cviu.2009.03.008, URL http://dx.doi.org/10.1016/j.cviu.2009.03.008.

[15] Makadia, A., Pavlovic, V. and Kumar, S. (2010) Baselines for image annotation. *Int. J. Comput. Vision* **90**(1): 88–105. doi:10.1007/s11263-010-0338-6, URL http://dx.doi.org/10.1007/s11263-010-0338-6.

[16] Zhou, H. and Wornell, G. (2014) Efficient homomorphic encryption on integer vectors and its applications. In *Information Theory and Applications Workshop (ITA), 2014* (IEEE): 1–9.

[17] Boldyreva, A., Chenette, N., Lee, Y. and O'Neill, A. (2009) Order-preserving symmetric encryption. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09 (Berlin, Heidelberg: Springer-Verlag): 224–241. doi:10.1007/978-3-642-01001-9_13, URL http://dx.doi.org/10.1007/978-3-642-01001-9_13.

[18] Roche, D.S., Apon, D., Choi, S.G. and Yerukhimovich, A. (2016) Pope: Partial order preserving encoding. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16 (New York, NY, USA: ACM): 1131–1142. doi:10.1145/2976749.2978345, URL http://doi.acm.org/10.1145/2976749.2978345.

[19] Bentley, J.L. (1975) Multidimensional binary search trees used for associative searching. *Communications of the ACM* **18**(9): 509–517.

[20] Rane, S., Sun, W. and Vetro, A. (2010) Privacy-preserving approximation of l1 distance for multimedia applications. In *2010 IEEE International Conference on Multimedia and Expo*: 492–497. doi:10.1109/ICME.2010.5583030.

[21] William, B.J. and Joram, L. (1984) Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics* **26**: 189–206.

[22] Science, M.C. and Laboratory, A.I. (2017), Labelme dataset, http://labelme.csail.mit.edu/Release3.0/browserTools/php/dataset.php.

[23] Bishop, C.M. (2006) *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Secaucus, NJ, USA: Springer-Verlag New York, Inc.).

[24] Brakerski, Z., Gentry, C. and Halevi, S. (2013) Packed ciphertexts in lwe-based homomorphic encryption. In *16th International Conference on Practice and Theory in Public-Key Cryptography (PKC)*: 1–13.

[25] Katz, J. and Lindell, Y. (2007) *Chapter 11, Introduction to Modern Cryptography* (Chapman & Hall/CRC).

[26] Developers, N. (2013) Numpy. *NumPy Numpy. Scipy Developers* .

[27] Bradski, G. *et al.* (2000) The opencv library. *Doctor Dobbs Journal* **25**(11): 120–126.

[28] Wasilewski, F. (2006), PyWavelets - Wavelet Transforms in Python, https://pywavelets.readthedocs.io/en/latest/.

[29] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M. *et al.* (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**: 2825–2830.

[30] Wang, X.J., Zhang, L., Jing, F. and Ma, W.Y. (2006) Annosearch: Image auto-annotation by search. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (IEEE), **2**: 1483–1490.

[31] Russell, B.C., Torralba, A., Murphy, K.P. and Freeman, W.T. (2008) Labelme: a database and web-based tool for image annotation. *International journal of computer vision* **77**(1-3): 157–173.

[32] Verma, Y. and Jawahar, C.V. (2012) Image annotation using metric learning in semantic neighbourhoods. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12 (Berlin, Heidelberg: Springer-Verlag): 836–849. doi:10.1007/978-3-642-33712-3_60, URL http://dx.doi.org/10.1007/978-3-642-33712-3_60.

[33] Chapelle, O., Haffner, P. and Vapnik, V.N. (1999) Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks* **10**(5): 1055–1064.

[34] Cusano, C., Ciocca, G. and Schettini, R. (2003) Image annotation using svm. In *Electronic Imaging 2004* (International Society for Optics and Photonics): 330–338.

[35] Shi, R., Feng, H., Chua, T.S. and Lee, C.H. (2004) An adaptive image content representation and segmentation approach to automatic image annotation. In *International conference on image and video retrieval* (Springer): 545–554.

[36] Vailaya, A., Figueiredo, M.A., Jain, A.K. and Zhang, H.J. (2001) Image classification for content-based indexing. *IEEE transactions on image processing* **10**(1): 117–130.

[37] Rui, S., Jin, W. and Chua, T.S. (2005) A novel approach to auto image annotation based on pairwise constrained clustering and semi-naive bayesian model. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International* (IEEE): 322–327.

[38] Park, S.B., Lee, J.W. and Kim, S.K. (2004) Content-based image classification using a neural network. *Pattern Recognition Letters* **25**(3): 287–300.

[39] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*: 1097–1105.

[40] Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*: 580–587.

[41] Schroff, F., Kalenichenko, D. and Philbin, J. (2015) Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*: 815–823.

[42] Gong, Y., Jia, Y., Leung, T., Toshev, A. and Ioffe, S. (2013) Deep convolutional ranking for multilabel image annotation. *arXiv preprint arXiv:1312.4894* .

[43] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C. and Xu, W. (2016) Cnn-rnn: A unified framework for multi-label image classification. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on* (IEEE): 2285–2294.