applied sciences

MDPI

Article

SEPIM: Secure and Efficient Private Image Matching †

Zaid Ameen Abduljabbar [1,2], Hai Jin [1,*], Ayad Ibrahim [2], Zaid Alaa Hussien [1,3], Mohammed Abdulridha Hussain [1,2], Salah H. Abbdal [1] and Deqing Zou [1]

1 Cluster and Grid Computing Lab, Services Computing Technology and System Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China; alsulamizaid@gmail.com (Z.A.A.); zaidpc2005@gmail.com (Z.A.H.); mohsubber@gmail.com (M.A.H.); salahrfesh@gmail.com (S.H.A.); deqingzou@hust.edu.cn (D.Z.)

2 Department of Computer Science, University of Basrah, Basrah 61001, Iraq; mraiadibraheem@gmail.com

3 Department of Management, Southern Technical University, Basrah 61001, Iraq

* Correspondence: hjin@hust.edu.cn; Tel.: +86-27-8754-3529 (ext. 8033); Fax: +86-27-8755-7354

† This paper is an extended version of paper published in the 11th International Conference on Green, Pervasive and Cloud Computing (GPC'16), Xi'an, China, 6–8 May 2016.

Academic Editor: Antonio Fernández-Caballero

Received: 24 May 2016; Accepted: 22 July 2016; Published: 29 July 2016

Abstract: Matching a particular image within extensive datasets has become increasingly pressing in many practical fields. Hence, a number of matching methods have been developed when confidential images are used in image matching between a pair of security agencies, but they are limited by either search cost or search precision. In this paper, we propose a privacy-preserving private image matching scheme between two parties where images are confidential, namely secure and efficient private image matching (SEPIM). The descriptor set of the queried party needs to be generated and encrypted properly with the use of a secret key at the queried party side before being transferred to the other party. We present the development and validation of a secure scheme to measure the cosine similarity between two descriptor sets. To hasten the search process, we construct a tree-based index structure by utilizing the *k*-means clustering algorithm. The method can work without using any image encryption, sharing, and trusted third party. SEPIM is relatively efficient when set against other methods of searching images over plaintexts, and shows a higher search cost of just 14% and reduction in search precision of just 2%. We conducted several empirical analyses on real image collections to demonstrate the performance of our work.

Keywords: secure private image matching; feature protection; *k*-means clustering; secure multiparty computing (SMC); speeded up robust features (SURF) descriptors; homomorphic encryption

1. Introduction

The recent explosion of the World Wide Web and increasing interest from various multimedia fields has seen a concurrent significant elevation of the importance of digital images. The increased requirements placed on efficient private image matching (PIM) techniques in various applications interacting with reality have coincided with this. These applications may include social media [1,2] business community [3], e-health [4], and criminal suspect identification, etc. In the context of private image retrieval, similar images are usually brought together such that similar images can be retrieved efficiently once a query image is sent. In general, the PIM method refers to a process whereby a pair of parties determines their common matching values or similarities, whilst maintaining privacy for their own data. Hence, PIM only requires the magnitude of similarity, rather, content similarity.

According to [5], private matching (PM) can be classified into three scenarios. In the first scenario, the parties involved, namely *Alice* and *Bob*, must both learn the final results of PM as a result of the so-called symmetric PM. The second scenario involves a non-symmetric PM where only one party

learns if a commonality of values exists. The third scenario seeks to determine the number of common elements rather than whether values match exactly. All of these requirements have been met and addressed using different PM protocols.

We employ the second scenario in a secure manner to meet the requirements of actual security applications. Simply stated, in some cases, protecting the privacy of images during the matching process is necessary. Consider the following example to determine the importance of a security issue. Suppose a security agency is searching for data related to a potential terrorist suspect. The agency may wish to check whether images related to the suspect can be found in local police databases. However, for security purposes, neither the agency nor the local police want to reveal their images unless a need to share exists. One way to identify such a need is to detect similarities between the agency's query (in the form of images) and the local police's image collections. Once the need for sharing information is verified, the agency and local police can exchange only shared information. During the process of identifying similar images, the best choice for both parties is to not disclose the query image and the database, and have the former learn only of the existence of any commonality of image matching values (second scenario).

Specifically, in image matching cases used for searching an image related to terrorist suspects, the effectiveness of a security agency may be reliant on the speed and accuracy with which a process of finding a matching image can be undertaken. The timeliness and accuracy with which such a security agency may acquire this information can have real-world effects. For this reason, it is important that a robust and fast system be constructed to secure and quickly scan sizable amounts of data, which is as effective and complex as images' plaintext searches. Such a process is referred to as secure and efficient private image matching (SEPIM).

Most image matching (IM) approaches define an image representation and a distance metric that reduce the amount of data stored per image and the time cost of database search. Feature vectors (descriptors) of each image in the database are extracted and stored. During the matching, the descriptors of the query image are compared against their counterparts in the database to determine the most relevant image. However, keeping descriptors in their clear text may reveal information on some objects in the image. Thus, such descriptors should be encrypted in such a way that their distances are preserved without decryption.

In this paper, we address the question of how to search for similar images between two parties in a privacy-preserving manner without losing image confidentiality. Given image I, *Alice* would like to determine whether there are images in *Bob's* collection D that are similar to I (e.g., duplicate, near duplicate, somewhat close, etc.) without disclosing either I or D. We focus primarily on security, where protecting the descriptors of images is necessary. Specifically, our proposed scheme supports speeded up robust features (SURF) local-feature [6,7] with cosine similarity [8], as well-known metric to score matching images, and employs homomorphic encryption [9] to protect the confidentiality of descriptors. To increase the effectiveness of the image search, we build a tree-based index structure by employing the k-means clustering algorithm to classify large-scale images database into various clusters. The method allows only the inquiring side to see the matching value. Hence, only *Alice* is interested in determining whether she has any image in common with *Bob*, without worrying about the leakage of unnecessary information.

The contributions of this paper are as follows. First, a trivial solution to achieve secure and private image matching is to utilize a trusted third party (TTP). *Alice* sends I to the TTP and *Bob* sends D to the TTP, and then TTP can investigate and inform *Alice* whether images similar to I can be found in *Bob's* collection. However, in real life situations, finding a completely trustworthy third party is a difficult task. Our work does not require such a third party. Second, the applications of PIM often suffer from significant overhead for the image encryption operation. Our scheme can work without image encryption and still maintain the privacy of the parties involved. Third, in SEPIM, the duration of an enquiry and its accuracy are comparable with enquiries conducted using plaintext search methods.

Finally, for communication cost, our scheme only requires one round of communication between a query side and a data owner side, while others need multiple rounds of communication.

The rest of this paper is organized as follows. Related works are reviewed and discussed in Section 2. Section 3 introduces the preliminary techniques. Section 4 introduces the security requirements and the problem definition. Section 5 provides the detail construction of SEPIM. The evaluation of SEPIM is provided in Section 6, and conclusions and future works are drawn in Section 7.

2. Related Works

Ever since Freedman et al. [10] brought up the first solution using a private matching mechanism to prevent the leakage of unnecessary information between two parties, a number of authors have subsequently proposed different private matching mechanisms. These mechanisms typically conform to the different requirements of such parties in PM or are the results of fine-tuning to achieve low overhead in terms of computational cost and to enhance the search precision. However, most of these schemes suffer from drawbacks. Keeping this in view, we will present related works pertaining to PM and its drawbacks. Works within the context of image private matching will also be highlighted.

The important factors in the field of PM are the protocol of private set union (PSU), [4,11,12], and private set intersection (PSI) [10,13–15], respectively. Cristofaro et al. [16] revealed that these techniques do not provide adequate privacy on the server end and, thus, a server could compromise privacy. In [16], a scenario is proposed where users are allowed to learn only the magnitude of the shared values instead of the exact values. Such a scenario uses the Private Set Union Cardinality (PSI-CA) and a third-party server. Ferreira et al. [17] proposed a system to search encrypted images databases stored on a server maintained by a third-party service provider. Under both [16,17], the server should not know its stored data. Our work obviates the use of any third party for security purposes.

Shashank et al. [18] applied private information retrieval (PIR) techniques to protect the privacy of the query image when searching over a public database. However, such a method assumes that the database is public when such database is supposed to be private. The proposed methods in [16–18] are also not suitable for evaluating similarity. These approaches can achieve an exact match, thereby limiting the ability to develop efficient solutions.

In [19], Agrawal et al. proposed a method for private matching using double encryption under the assumption that $x \in X, E(E'(x)) = E'(E(x))$, where E is the encryption function. To determine the common elements between two parties, the authors proposed using the crypto-hash function. Initially, such a function should be decided between the parties involved. Thus, this approach encourages a curious party to utilize a brute force attempt using the same hash function to determine uncommon elements over a finite domain of elements. In our work, we avoid the use of any hash function to prevent a curious user from obtaining additional information.

The schemes put forward by Lu et al. [20] emphasized secure image search over encrypted datasets and the maintenance of data security, through Min-Hash and order preserving encryption. Their scheme is useful in implementation for an image search process based on bag-of-features (BOF), but has inferior search precision up to 20% compared with those using the Fisher vector as a foundation [21,22]. Despite the alternative protected search being proposed by Lu et al. [23], offering extended protection and embedding in secure BOF, this proposal greatly reduces the effectiveness of search efficiency when put alongside [20] (see the analysis in [23]).

Specifically, to enhance the search precision, Perronnin et al. [21] implemented the Fisher vector within the search system, and demonstrated that the scheme based on a Fisher vector can achieve better performance compared to schemes based on BOF in terms of search precision. Following [21], developed Fisher vector based image search methods are suggested to further improve search precision in various instances [22].

Under both [21,22], the Fisher vector was based on scale-invariant feature transform (SIFT), which is used to extract local feature vectors. Our work uses the SURF method that produces fewer

local features and also can be bettered in terms of feature extraction processing speed [6,7] compared with the SIFT method used by [21,22]. The comparison between SURF and SIFT local feature is discussed further in Section 3.

In recent years, researchers have proposed feature descriptors based on intensity such as binary robust independent elementary features (BRIEF) [24] and SYnthetic BAsis (SYBA) [25]. The BRIEF and SYBA descriptors are binary descriptors that consist of a binary string including the results of intensity comparison at random pre-determined pixel locations. These two descriptor methods employ faster feature detectors and provide lowering descriptor size than SIFT and SURF [26,27]. For object recognition, BRIEF and SYBA outperformed both SIFT and SURF for high performances value. However, BRIEF and SYBA did not perform well when there is a large viewpoint change, invariance to rotation, and illumination changes [26,27]. In other words, as the descriptor is mostly responsible for improving the feature detector by extracting rotation and illumination invariant descriptors, descriptors such as BRIEF and SYBA that are truly disassociated with any detector would be unable to enhance the capabilities of the detectors, thereby SURF outperformed both BRIEF and SYBA algorithms for high recall values used with orientation and illumination changes. For this reason, our work uses the SURF algorithm.

In our work, to compare the similarity of two images Img_1 and Img_2, their corresponding SURF descriptor vectors V_1 and V_2 will be normalized. Then, by utilizing the principal component analysis (PCA) transform [22], lowering the dimensionality of vectors V_1 and V_2 can be minimized with a loss that has no effect on discriminative power. We will analyse the relationship between lowering dimension and search precision in Section 6. Finally, the similarity of Img_1 and Img_2 is evaluated by the secure cosine similarity.

3. Preliminaries

Before providing our proposed scheme, we briefly explain the method used to extract the feature vectors for the image collection and the method used to measure the search precision.

3.1. Feature Extraction

Most feature vectors are either global vectors, such as a global color histogram, or local vectors such as SIFT descriptors [28,29] and SURF descriptors [6,7]. The first model generates an extreme compressed feature vector for each image. Such a model can effectively identify global similarities, e.g., how many colors two images share. The second model searches the image to identify the interest key points invariant to scale and orientation. A feature descriptor is generated for each key point. Compared with global-feature based image retrieval, local-feature based image retrieval characteristically acquires more accurate results than the globally based equivalent but requires comparatively complex metrics relating to distance. In this paper, we will focus on SURF local features model, which has the advantage of identifying local similarities, e.g., scenes and objects. PCA transform is employed to achieve distance metric efficiency [22].

Specifically, the SURF algorithm [6,7] is a novel scale and rotation-invariant detector and descriptor. SURF approximates or even outperforms a previously proposed SIFT algorithm [28,29], which is patented, with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster. Thus, the method posited here improves feature extraction speed. SURF extracts the feature vectors of the provided image as follows. First, SURF selects several interest points at distinctive locations in the image, such as corners, blobs, and T-junctions. Such points are selected in such a way that enables the detector to find the same physical interest points under different viewing conditions. Next, the neighborhood of every interest point is represented by a feature vector. This descriptor has to be distinctive and robust to noise, detection displacements, and geometric and photometric deformations. The descriptor vectors are matched between different images. Matching is based on a distance between the vectors, e.g., the Euclidean distance or cosine similarity. Figure 1 illustrates the interest points of Lena image and their counterparts in the same image after rotation.

Formally, given image *Img*, we use the SURF algorithm to generate its feature vectors $F = \{v_1, v_2, ..., v_k\}$, where k is the number of interest points in the provided image. Note that different images may differ in the number of descriptors k. Then, we utilize the PCA transform [22] to facilitate a lowering of the dimensionality of SURF vectors so that the performance of the matching time can improve with ineffectual of discriminative power. Recall that the relationship between lowering dimension and the precision of image searches is analysed further in Section 6.

Figure 1. Speeded up robust features (SURF) interest point of two images.

3.2. Mean Average Precision

Mean Average Precision (MAP) [30] is a method to evaluate the search precision of image search, which is commonly employed by existing image search algorithms [21,22]. Through MAP, a mean value of precision is calculated over a series of searches. Suppose *Alice* runs a search enquiry into *Bob's* dataset, with a return of a limited number of results set $\{Img_1, Img_2, ..., Img_{10}\}$. In the result set, if two results Img_1 and Img_{10}, which correspond to real similarity of the requested image, we obtain the application of an AP (Average Precision) value as $AP = (1/1 + 2/10)/2$ (if k similar images in the dataset be overlooked in the results set, $AP = (1/1 + 2/10 + 0 \times k)/(2 + k)$). The formula for calculating MAP in n searches is $= \sum_{i=1}^{n} AP_i/n$. Further information about MAP calculations can be found in [30].

4. Security Definition and Problem Statement

4.1. Security Definition

Our security definition follows the secure multiparty computing definition of Goldreich et al. [31] and private matching [5]. We assume that the parties involved are semi-honest. A semi-honest party follows the steps of the protocol using the party's correct input, but attempts to utilize what it sees during the execution of the protocol to compromise security. This model guarantees that parties who follow the protocol correctly cannot gain any knowledge on the other party's input data except for the output. No additional information is disclosed and information that can be inferred from its own input is avoided.

4.2. Problem Statement

The common notations listed in Table 1 are used throughout this paper. Our proposed scheme includes two parties, namely, *Alice* and *Bob*, each of whom has a collection of images. We assume that the images of both parties are private. Given an image I of *Alice*, we are interested in determining whether *Bob's* collection contains an image similar to I without disclosing *Bob's* database to *Alice* and vice versa. We evaluate the similarity of two images under the SURF local feature vector model, in which each image is represented as a set of vectors. Let $D = \{Img_1, ..., Img_m\}$ denote the set of m images in *Bob's* collection. Without disclosing I to *Bob* and D to *Alice*, our objective is to find a set

of images in D similar to I without disclosing the matching results to *Bob*. We term such protocol as SEPIM. Formally, SEPIM is defined as

$$SEPIM(I, D) = \alpha_1, \alpha_2, ..., \alpha_M. \quad (1)$$

Table 1. Common symbols used.

Symbol	Meaning
D	Large-scale image database
I	Query image
m	Number of images in *Bob's* collection
k	Number of descriptors of *Alice's* image
p	Number of descriptors of each one of *Bob's* images
n	Size of descriptors

SEPIM returns the M similarity scores $\alpha_1, \alpha_2, ..., \alpha_M$ to *Alice* instead of returning the actual images. At another time, *Alice* can retrieve the similar image from *Bob*. To evaluate the similarity between two images, each party initially extracts the feature vectors for each image in its own collection. Several metrics are used to evaluate the similarity between the sets of the two feature vectors such as Euclidean distance and cosine similarity [8]. The cosine similarity (CSIM) between vectors v_1 and v_2 of size n can be defined as follows:

$$CSIM(v_1, v_2) = \frac{\sum_{i=1}^{n} v_1[i].v_2[i]}{\|v_1\|.\|v_2\|}, \quad (2)$$

where $\|v\|$ is the Euclidian length of vector v, and is defined as the following:

$$\|v\| = \sqrt{\sum_{i=1}^{n} v[i]^2}. \quad (3)$$

Given normalized vectors $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$, cosine similarity can be written as:

$$CSIM(\overrightarrow{V}_1, \overrightarrow{V}_2) = \sum_{i=1}^{n} \overrightarrow{V}_1[i].\overrightarrow{V}_2[i]. \quad (4)$$

Here,

$$\overrightarrow{V}[i] = \frac{v[i]}{\|v\|}. \quad (5)$$

Given two images, Img_1 and Img_2, of the two feature vector sets $F_1 = \{v_1, v_2, ..., v_k\}$ and $F_2 = \{s_1, s_2, ..., s_p\}$, respectively. Algorithm 1 illustrates how the distance between two feature vector sets can be measured through the cosine similarity without preserving privacy.

Algorithm 1 Insecure Image Distance Calculation

Input: two feature vectors $F_1 = \{v_1, v_2, ..., v_k\}$ and $F_2 = \{s_1, s_2, ..., s_p\}$ of two images.
All vectors v_i and s_i are of the same size n.
Output: $Dist$: distance between F_1 and F_2.
$Dist = 0$;
For $i = 1$ to k **do**
  Compute $\overrightarrow{v}_i$ as in Equation (5)
  For $j = 1$ to p **do**
   Compute $\overrightarrow{s}_i$ as in Equation (5)
   $D_j = 1 - CSIM(\overrightarrow{v}_i, \overrightarrow{s}_j)$
  End for$//j$
  $Dist = Dist + min(D_j), \forall j = 1, ..., p$
End for$//k$
$Dist = Dist/k$

Table 2 shows a trivial example for *Alice* image, which is represented by a set of three vectors of size 5. The first three columns are the feature vectors, while the last three columns are their corresponding normalized versions. Similarly, Table 3 illustrates the collection of *Bob*, which consists of two images. In addition, this table is interpreted in the same way as Table 2.

Table 2. *Alice's* Image.

***Alice* Image**					
Feature vector F_1			**Normalized vector $\overrightarrow{F}_1$**		
Feature vector sets			**Normalized vector sets**		
v_1	v_2	v_3	$\overrightarrow{v}_1$	$\overrightarrow{v}_2$	$\overrightarrow{v}_3$
1	3	3	0.1348	0.5145	0.75
5	2	1	0.6742	0.343	0.25
2	4	2	0.2697	0.686	0.5
3	1	1	0.4045	0.1715	0.25
4	2	1	0.5394	0.343	0.25

Table 3. *Bob's* Collection.

***Bob* Collection of Two Images**											
Feature vectors						**Normalized vectors**					
F_2			F_3			$\overrightarrow{F}_2$			$\overrightarrow{F}_3$		
Feature vectors sets						**Normalized vectors sets**					
s_1	s_2	s_3	x_1	x_2	x_3	$\overrightarrow{s}_1$	$\overrightarrow{s}_2$	$\overrightarrow{s}_3$	$\overrightarrow{x}_1$	$\overrightarrow{x}_2$	$\overrightarrow{x}_3$
2	1	3	2	3	3	0.3592	0.1796	0.5388	0.417	0.7746	0.6882
1	2	2	1	2	2	0.1796	0.3592	0.3592	0.2085	0.5164	0.4588
3	4	1	0	1	1	0.5388	0.7184	0.1796	0	0.2582	0.2294
1	3	1	3	0	1	0.1796	0.5388	0.1796	0.6255	0	0.2294
4	1	4	3	1	2	0.7184	0.1796	0.7184	0.6255	0.2582	0.4588

To compute the distance between *Alice's* image and the first image in *Bob's* collection, we have to compute distance between the feature vector sets F_1 and F_2. Thus, the distance between F_1 and F_2 can be calculated as follows:

$Dist_1 = (min((1 - CSIM(\overrightarrow{v}_1, \overrightarrow{s}_1)), (1 - CSIM(\overrightarrow{v}_1, \overrightarrow{s}_2)), (1 - CSIM(\overrightarrow{v}_1, \overrightarrow{s}_3))) + ... + min((1 - CSIM(\overrightarrow{v}_3, \overrightarrow{s}_1), (1 - CSIM(\overrightarrow{v}_3, \overrightarrow{s}_2), (1 - CSIM(\overrightarrow{v}_3, \overrightarrow{s}_3))))/3$
$=(min(0.225, 0.225, 0.1766) + min(0.1067, 0.1375, 0.1991) + min(0.1981, 0.2367, 0.1918))/3$
$= (0.1766 + 0.1067 + 0.1918)/3 = 0.1584.$

We note that these calculations are based on minimum cosine values of corresponding vectors sets. We also note that we compute distance between vectors using the dot product, which is equivalent to cosine distance since we assume feature vectors are normalized. Similarly, the distance between F_1 and F_3 is $Dist_2 = 0.1375$. Thus, we can conclude that the second image in *Bob's* collection is more similar to *Alice's* image than the first one because it has a shorter distance.

As shown in the above example, the main step in evaluating similarity between two images is the dot product between their corresponding normalized vectors. Therefore, once we know how to calculate the dot product in a privacy-preserving manner, we can calculate the distance between any two images without sharing their contents.

In the following subsection, we will demonstrate a homomorphic encryption-based protocol [32] for computing the dot product operation in a privacy-preserving mode. We then show how to utilize such a protocol as a tool in designing our proposed SEPIM.

4.3. Secure Dot Product Based on Homomorphic Encryption

Homomorphic encryption is a probabilistic public key encryption [9,32]. Let $HE_{pk}(x)$ and $HD_{pr}(y)$ be the encryption and decryption functions in this system with public key pk and private key pr. Without private key pr, no adversary can guess the plaintext x in polynomial time. Furthermore, $HE_{pk}(x)$ has a semantic security [33] property, which means no adversary can compute any function of the plaintext from the ciphertext set. Interestingly, the full homomorphic encryption has two amazing properties, namely: additive and multiplicative. Additive property allows adding two encrypted numbers, i.e., $HE_{pk}(x1) \times HE_{pk}(x2) = HE_{pk}(x1 + x2)$. Given a constant c and a ciphertext $HE_{pk}(x)$, the multiplicative property works as follows: $HE_{pk}(x)^c = HE_{pk}(c \times x)$. In this paper, we adopt Paillier's system [34] for the practical implementation because of its efficiency.

Let u and v be secure vectors of *Alice* and *Bob*, respectively. *Both* vectors are of the same size n. Below, we show how homomorphic encryption can be used to compute the secure dot product between u and v. At the beginning, *Alice* encrypts her private vector component-wise, i.e., $z_i \leftarrow HE_{pk}(u_i)$, and sends the encrypted vector z to *Bob*. Upon receiving z, *Bob* computes the encrypted component-wise product between z and v based on the multiplicative property, (i.e., $y_i = z_i^{vi}, for\ all\ i = 1, ..., n$). He then sums up these products based on the additive homomorphic property to compute the encrypted dot product *EDot* such as: $EDot = y_1 + y_2 + ... + y_n$. After receiving *EDot* from *Bob*, *Alice* uses her private key pr to decrypt it and to obtain the plaintext value of $u \times v$, i.e., $HD_{pr}(EDot) = u \times v$. Note that *Alice's* private vector u is not revealed to *Bob* because only encrypted values of u are sent to *Bob*. Therefore, without prior knowledge of *Alice's* private key, neither u vector nor matching plaintext can be recovered by semi-trusted *Bob* or any adversary. Thus, this method meets the requirement of second scenario as explained in Section 1 with respect to privacy-preserving.

5. Construction of Secure and Efficient Private Image Matching (SEPIM)

As shown in Figure 2, our SEPIM protocol distributes scores calculation between the two participant parties and is composed of two phases, initialization and matching. In the first phase, each party computes the feature vector set for each image in its own collection and then normalizes each vector to enable the assessment of the cosine similarity. We demonstrate the proposed scheme using SURF descriptors, although this scheme is applicable to other feature vectors. Then, through the use of Algorithm 2, the data owner (*Bob*) builds an effective search index tree *TreeIndex* used in image search.

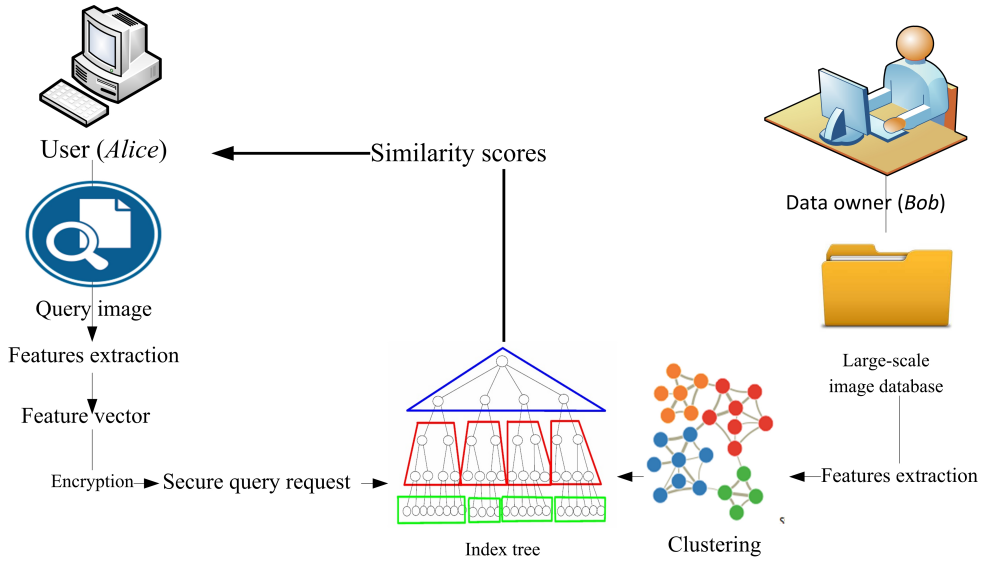

Figure 2. Architecture of Secure and Efficient Private Image Matching (SEPIM) scheme.

Algorithm 2 Index Tree Building

Input: $\{\overrightarrow{V_i}\}, 1 \leq i \leq m, R$: k-means cluster number
Output: *TreeIndex*.
$N_r = m$;//Number of vectors in cluster r : CL_r
Child_Gen($\{\overrightarrow{V_i}\}_{1 \leq i \leq m}, R$);
Function Child_Gen($vectors\{V_i\}, R$) begin
k-means$(\{V_i\}, R) \rightarrow R$ cluster CL_r;
For $r = 1$ to R **do**
If $N_r > R$ Then
Child_Gen($vectors\{V_i\}_{i \in CL_r}, R$);
End for//r

In the second phase, *Alice* generates a secure image search request. Upon receiving of *Alice's* request, *Bob* executes the secure and efficient private image matching algorithm (Algorithm 3) to retrieve values indicating similarity scores to *Alice* instead of returning the actual images. At another time, *Alice* can retrieve the similar image from *Bob*. We now introduce the detail of each algorithm in our SEPIM construction.

Algorithm 3 Secure and Efficient Private Image Matching

Input: I: *Alice's* image.
$D = \{Img_1, ..., Img_m\}$: *Bob*$'s$ collection.
Output: $\alpha_1, \alpha_2, ..., \alpha_M$: the similarity scores.
Initialization:
Alice:
-Generate the homomorphic encryption public key pair (pr, pk).
-Define the similarity threshold *Sth*.
-Send *pk* and *Sth* to *Bob*.
-Use SURF algorithm to extract the feature vector set $F = \{v_1, v_2, ..., v_k\}$ for the image I, all vectors v_i are of the same size n.
-Compute $\overrightarrow{v}_i$ as in Equation (5), for $i = 1, ..., k$, and replace it with v_i in F.
Bob:
For each image $Img_j \in D, \forall_j = 1, ..., m$
- Use SURF algorithm to extract the feature vector set $F_j = \{s_1, s_2, ..., s_p\}$.
- Compute $\overrightarrow{s}_i$ as in Equation (5), for $i = 1, ..., p$, and replace it with s_i in F_j.
Endfor $//j$
-The owner randomly selects R centroids
-Use index tree building algorithm (Algorithm 2) to cluster the feature vector set $\overrightarrow{\{F_j\}}, 1 \leq j \leq m$ into clusters $CL_r, 1 \leq r \leq R$ and builds index tree structure.
Matching:
Alice:(*first round*)
For $i = 1$ to k **do**
Encrypt the elements of vector $\overrightarrow{v}_i$ as:
$z_{ij} \leftarrow HEnc_{pk}(\overrightarrow{v}_{ij})$ for all $j =, 1, ..., n$
Endfor $//i$
- Send z to *Bob*
Bob:(Step1)
-Starting the search enquiry from the top level to leaf level of *TreeIndex*, and at every level:
- Acquire the set of node vectors $\{VF_{hi}\}, 1 \leq i \leq R$;
- Calculate the cosine similarity based on secure dot product between the *Alice's* search vectors set and the node vectors set $\{VF_{hi}\}, 1 \leq i \leq R$;
- Finds the entry points of the next level by finding the nodes with vectors that have the minimal distances to the received search image vector z according to the threshold *Sth*;

Algorithm 3 *Cont*.

-The process (step 1) repeated until a leaf is reached

-Get all vectors $\{\overrightarrow{F_j}\}, 1 \leq j \leq M$ associated with the indexes of index tree (with leaf level of *TreeIndex*)

Bob:(Step2)

For $m = 1$ to M **do**

- Get the feature vector set F_m of image m.
- Compute the secure dot product set between the *Alice's* vector set and the vector set of image m as:

$Dot\{m\} = Secure\_dot\_product(Z, F_m)$;

Endfor$//m$

- Send $Dot\{m\}$ to *Alice*.

Alice:(*second round*)

- Receive *Dot* from *Bob*, where each element in *Dot* is a matrix of $[k, p]$ dimensions.
- **For** $m = 1$ to M **do**

Set X to be matrix m of Dot.

$Sum = 0$;

For $i = 1$ to k **do**

For $j = 1$ to p **do**

$sub_j = 1 - HDec_{pr}(X_{ij})$ // this is because: distance=1−similarity

Endfor$//j$

$min = minimum(sub)$

$Sum = Sum + min$;

Endfor$//i$

Compute the distance with image m as:

$\alpha_m = sum/k$

Endfor$//m$

5.1. Index Tree Building

To speed up the image search process, the data owner *Bob* needs to construct an index tree *TreeIndex* for large-scale images dataset as an example shown in Figure 2. We follow the method similar to the TreeBASIS descriptor in the context of the search process using k-means clustering [35]. Particularly, we implement the k-means clustering algorithm in a process of classifying image dataset into various clusters. Specifically, the data owner runs Algorithm 2 recursively separating the images dataset into R clusters, stopping when no cluster has greater than R images. The feedback from Algorithm 2 provides the data owner with a base from which to connect each image's descriptor vector $\overrightarrow{V_i}$ to a leaf node of *TreeIndex*, as well as allowing those nodes within a same cluster to be able to connect to the identical non-leaf node as their father. A k-dimensional mean descriptor vector VF_{hi} can also be assigned to identical non-leaf father nodes by the data owner, with h representing the height of the father node within *TreeIndex*, and I representing the index at height-h.

An element within VF_{hi} has its measure quantified as the mean value for the elements of its connected children. Such mean values will be employed during the search process of the tree up to leaf level, since the formula presented relies on the relevance of pair of images can be measured by the cosine similarity of their descriptor vectors. A formula representing VF_{hi} is:

$$\overrightarrow{VF}_{hij} = \frac{\sum_{\overrightarrow{v_i} \in Children} \overrightarrow{v_{ij}}}{|Children|}. \quad (6)$$

5.2. Secure and Efficient Private Image Matching

The implementation of SEPIM utilizes homomorphic encryption to evaluate similarity. The main steps are highlighted in Algorithm 3. To match her private image, *Alice* goes into two

rounds. In the first round, she encrypts her feature vector set and sends them to *Bob*. *Bob* receives *Alice's* encrypted vectors. As an example shown in Figure 3, *Bob* begins with the top level of the *TreeIndex* when undertaking a search request, subsequently moving to leaf level. At each level, *Bob* discovers the entry points of the subsequent level as he moves through the levels, through finding the node which has a vector VF_{hi} of lowest cosine distance, calculated by a secure dot product to the search image vector.

When he reaches the leaf level, (i.e., level *h-n* (see Figure 3), *Bob* will acquire the indexes of images which have the best relevances with the image used in the search enquiry. In other words, *Bob* will get all vectors $\{\overrightarrow{F_l}\}_{1\leq l\leq M}$ associated with the leaf of *TreeIndex*. Finally, he employs the secure_dot_product subroutine (as explained in Algorithm 4) to return the dot product matrix of the input vector set and the feature vector set $\{\overrightarrow{F_l}\}_{1\leq l\leq M}$ of each image that have top lower distances with the search request. The details of the above subroutine are explained in Section 4.3. Without loss of generality and to make the presentation clearer, we assume that all of *Bob's* images have the same number p of descriptors. In the second round, *Alice* uses her private key to decrypt the dot product terms and obtains the actual values, which will be employed in assessing the similarity scores as explained in Algorithm 1.

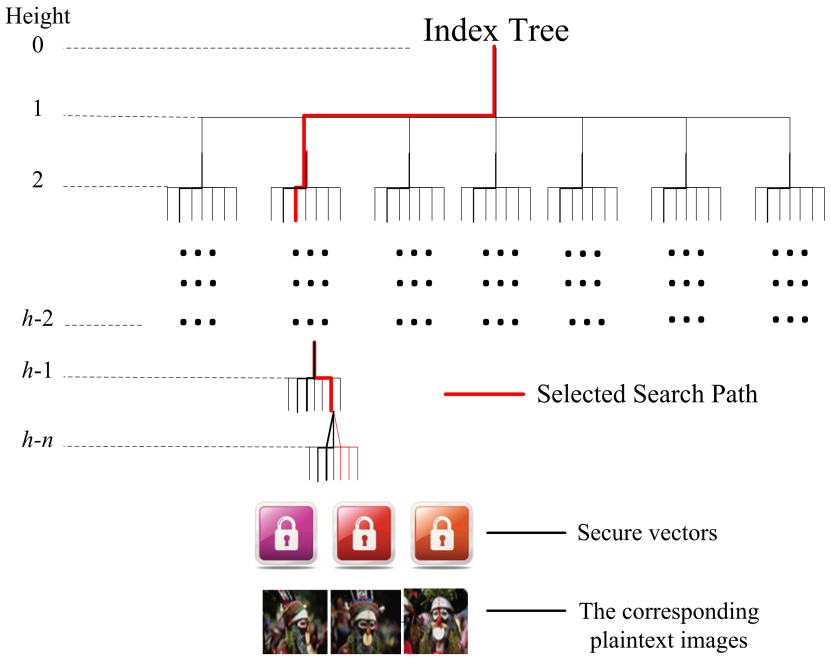

Figure 3. Image Search Example.

Algorithm 4 Secure_dot_product(Z, F)

Input: two feature vector sets Z and F of sizes k and p, respectively.
Output: $Dot[k, p]$: the encrypted sup-product terms between Z and F vectors.
For $i = 1$ to k **do**
  For $j = 1$ to p **do**
    $Dot_{ij} = HDec_{pk}(0)$; // initial value
    For $t = 1$ to n **do**
      $Dot_{ij} = (Z_{it}^{v_{jt}}) \times Dot_{ij}$
    Endfor//t
  EndFor//j
Endfor//i

6. System Evaluation

6.1. Security Analysis

Here, it is posited that the encrypted vectors, secure dot product based on homomorphic encryption, and encrypted query will not disclose information to the *Bob*.

6.1.1. Security of Encryption

In our SEPIM model, in order that a secure search request u is constructed consisting of feature vectors, we use a homomorphic encryption cryptographic scheme $z_i \leftarrow HE_{pk}(u_i)$, to ensure the security of features. For the search process, all vectors V_i, VF_{hi} associated with the index tree are encrypted through search process by secure dot product based on homomorphic encryption. As proved in the existing homomorphic encryption scheme [33], as long as the secret private key pr remain confidential to the *Bob*, it is impossible to obtain the plaintext format of V_i and VF_{hi}, in the known ciphertext homomorphic encryption model. Moreover, every item of V_i has an assigned weight, which differs from image to image contents. Therefore, the feature vectors pertaining to *Alice's* search request and all vectors assigned with the index tree are kept secure by means of homomorphic encryption and the secure dot product based on homomorphic encryption, respectively.

Theorem 1. *The Secure Request and Secure Vectors Will Not Reveal Extra Information Between Two Parties*

Proof of Theorem 1. In order to quantify the amount of information that is leaked to the *Bob* from the secure vectors V_i, VF_{hi} or secure search request z_i, we measure the dependency between these secure vectors and secure search request, and their corresponding images dataset D. The rationale informing this method is that low dependency translates into low information leakage. We use the mutual information (MI) entropy [36] to measure the dependency between two entities. MI is defined as follows:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y), \tag{7}$$

where *H(X)* and *H(Y)* are the entropy of the one-way hash function SHA-1 of the original image X and the secure vector or secure search request Y , respectively. The entropy of the random variable Z is defined as

$$H(Z) = \sum_{i=1}^{n} -P(z_i) log P(z_i), \tag{8}$$

and $H(X,Y)$ is the join entropy between the two variables X and Y , defined as

$$H(X,Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} -P(x_i,y_i) log P(x_i,y_i), \tag{9}$$

where $P(x)$ and $P(x,y)$ are the probabilities of occurrence of the outcome x, and the pair(x,y). Throughout our experiments, the secure search request and secure vectors yielded a low MI value equal to 0.0013. □

6.1.2. Search Unlinkability

In the SEPIM design, each search query has its different encryption key. Consequently, two search queries have different encrypted search vectors z even if they are from the same query image, thereby guaranteeing the unlinkability of different search queries.

6.2. Complexity Analysis

In this section, we measure the complexity of our proposed scheme in terms of computing time and communication cost. For computing time complexity, at the first round of *Alice's* side, encryption represents the most expansive operation, which is bounded by $O(k)$ or 70 ms for each image, where k is the number of descriptors in the input image. At *Bob's* side, the secure dot product subroutine is run $log_R(m)$ times, and each time it takes the complexity of $O(k.p)$. Thus, the overall computing time complexity of this step is $O((k.p)log_R(m))$, which is equivalent to 162.3 ms. Decryption represents the most expansive operation in the second round of *Alice's* side, and it takes around 65 ms for each similarity score. Thus, the total computing time of this step is bounded by $O(M.k.p)$ operations, where M is the number of similarity scores. With respect to the communication cost, we can summarize it as follows: in the first round, *Alice* sends $k.n$ values to *Bob* and *Bob* sends back $M.p.k$ values to *Alice*. Suppose that each value has b-bit long, then the total complexity is bounded as $O(b(k.n + M.p.k))$ bits.

6.3. Experimental Results

To evaluate the performance of our SEPIM construction in terms of search efficiency, search precision, effectiveness, and invulnerability to adversaries, we report the experimental results of the proposed scheme on a real image database containing 10,000 color images from the Corel dataset [37]. Our experiments are conducted on a 2.2 GHz Intel i7- 4702MQ processor, with a Windows 7 operating system of 64-bits, and 8 GB RAM (Lenovo PC HK limited, Hong Kong, China). We use MATLAB R2008a to implement our experiments. We used Java class to implement the Paillier cryptosystem.

The R of the index tree is set as 50 in our implementation. For the SURF descriptors, the size of each descriptor is 4096 elements, i.e., n = 4096. We also apply PCA transform on SURF vectors as in [22] to achieve dimension reduction. Specifically, we use PCA-2048, PCA-1024, PCA-512, PCA-256, and PCA-128 to denote PCA transforms that reduce the dimension of a SURF vector from n = 4096 to n = 2048, n = 1024, n = 512, n = 256, and n = 128, respectively. The normalized vectors are scaled by a user specific factor to convert the normalization (between 0 and 1) into integer numbers because the encryption function is applied only on integer values.

6.4. Search Efficiency

6.4.1. System Setup

To setup the system, the data owner first employs a SURF algorithm in order to extract a descriptor vector for search, each of which, in turn, requires the computation of a $O(m)(p)$-dimensional. Then, the data owner encrypts all descriptor vectors used for the search process, and each requires the computation of $O(m)(p)$-dimensional secure dot products. Following this step, the data owner proceeds to construct the index tree *TreeIndex* with $\sum_{r=0}^{logR(m)-1} R^r$ k-means$(\frac{m}{R^r}, R)$ operations. We evaluate the building cost of *TreeIndex* within the framework of lowering the dimension using the PCA method and the dataset size. Figure 4 shows that lowering dimension can lower the building cost of *TreeIndex*, since a majority of functions in the k-means algorithm are L_2 distance computation, and, therefore, with linear cost to the vector's dimension. Regarding the size of the data set, it is clear that the expansion of dataset size will necessitate a higher computational cost for building a *TreeIndex*, (see Figure 5) because the owner requires undertaking more k-means clustering on an increased amount of vectors.

Note that the initial setup for the system is a one-time function, which does not affect the effectiveness of the search process in real-time. In addition, enhancements to the implementation may be made through customizing it towards the procedure of tree construction through the employment of the parallel k-means clustering library [38], in which one billion 128-dimensional vectors might be processed within a 50 min period.

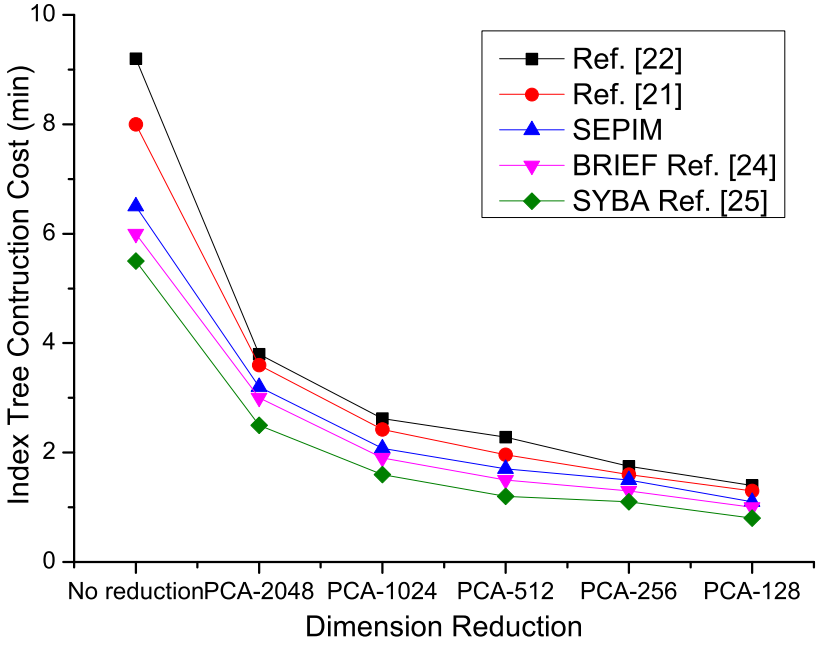

Figure 4. Index tree building cost for 10,000 images of variable dimension reductions. PCA: principal component analysis; BRIEF: binary robust independent elementary features; SYBA: SYnthetic BAsis.

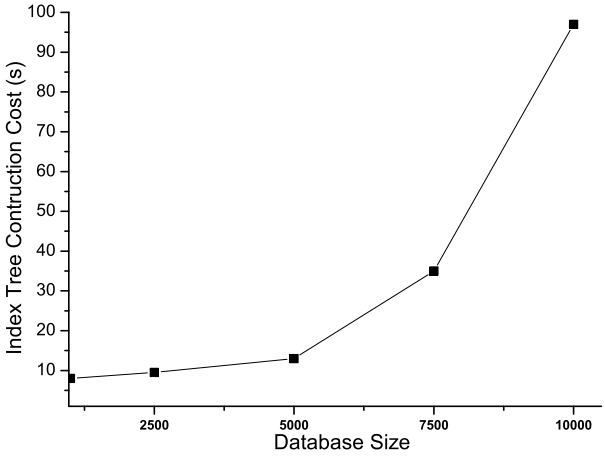

Figure 5. Index tree building cost of various image datasets using PCA-256 dimension reduction.

6.4.2. Search Time

In this experiment, we investigate the performance of our proposed scheme in terms of matching time. Our scheme requires the searching cost of index tree $O((k.p)log_R(m))$ to compute the search request. As shown in Figure 6, *Bob* is able to search a database of 10,000 within a period of 162.3 ms using SEPIM. SEPIM induces a higher search cost of just 14% over schemes which search images using plaintext [21,22,24,25]. The additional time cost of our work can be considered as a reasonable cost for achieving a secure matching. Figure 6 presents ways in which lessening dimension might

extend the enhancement of the search efficiency of SEPIM due to gains in cost among secure dot product computation, together with a reduced input/output (I/O) cost through search the index tree. In particular, PCA-256 can compress the file size from 35 MB to fewer than 3 MB of 10,000 images' vectors allocated to the index tree (see Figure 7), and these can also be cached simply within memory to speed up the search process.

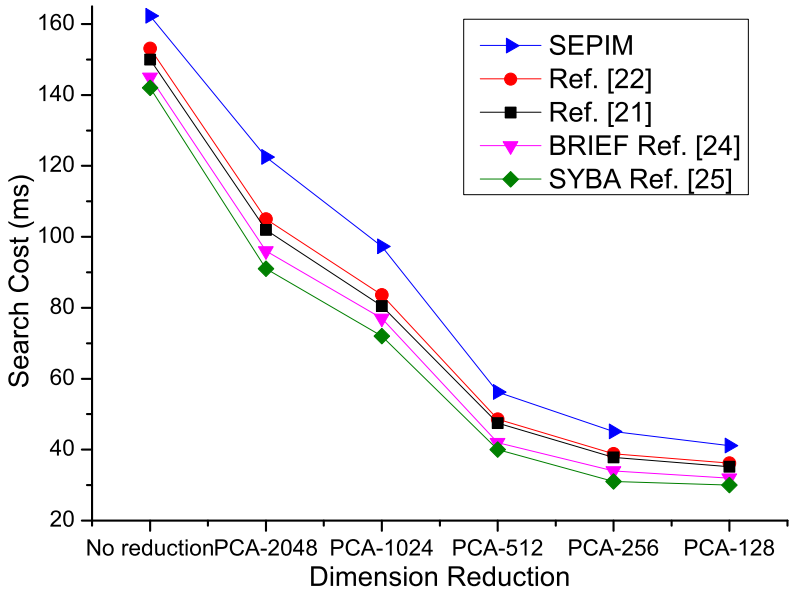

Figure 6. Search cost required for over 10,000 images with various dimension reductions.

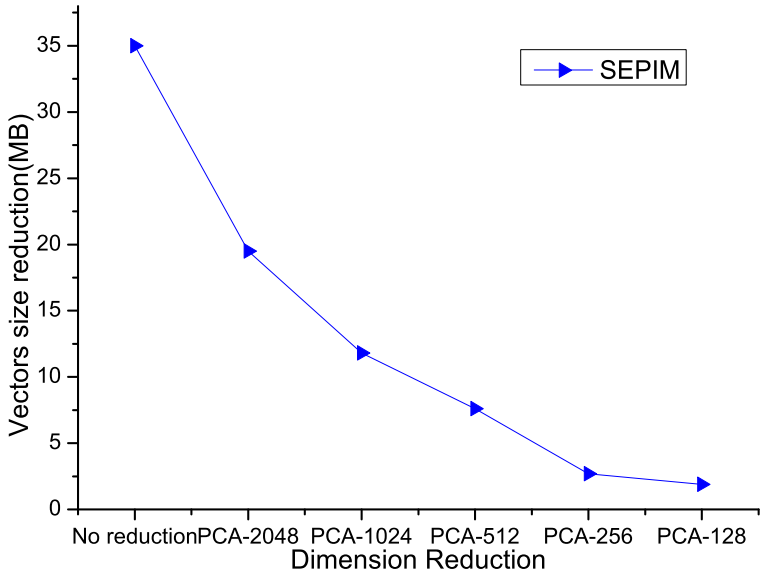

Figure 7. Vectors' size reduction over 10,000 images with various dimension reductions.

Clearly, PCA transform results in some data loss in relation to specific image features, possibly impacting the search precision in a limited way and somewhat offsetting the self-evident relationship between improved system effectiveness and lowering the descriptor vector using PCA transform. In the next part, we will analyse the search precision of SEPIM, and evaluate how to establish correct PCA strength might help to balance the effectiveness of a SEPIM-influenced system with regard to its efficiency and precision.

6.5. Search Precision

Figure 8 evaluates the level of search precision to which SEPIM conforms under variable dimension reductions. This formula employs MAP, which was introduced in Section 3.2, and demonstrates SEPIM acquiring a similar precision in search as plaintext image search schemes [21,22,24,25], with just 2% lower precision. SURF shows better performances as compared with SYBA and BRIEF approaches. Such approaches did not perform well when there is a large viewpoint change, and their lack of invariance to rotation is prominent. Because BRIEF and SYBA, unlike SURF, does not correct for orientation.

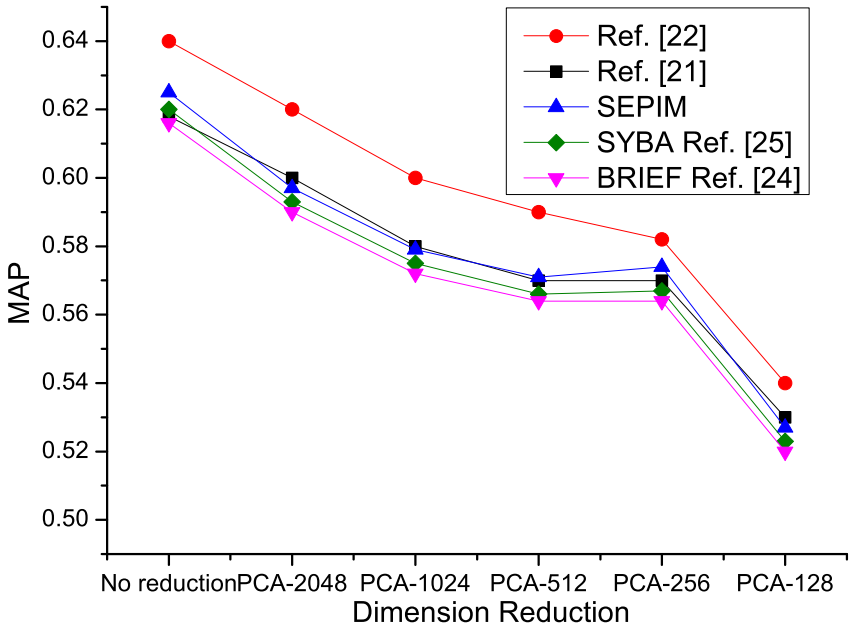

Figure 8. Variable dimension reductions affecting search precision for over 10,000 Images. MAP: mean average precision.

The figure renders clear the negative relation between the MAP of SEPIM and a dimension reduction over PCA-256, the stability of which runs counter to when it is lesser than PCA-256. From these results, it appears that PCA-256 is the ideal setting required to balance both the precision of the search and its efficiency.

6.6. Effectiveness

We test the ability of our proposed scheme to retrieve the images most similar to the provided query. Figure 9 shows samples of our results. The first column represents the provided image queries. The other columns show the returned images arranged according to their similarity to the

corresponding query. The columns show that our scheme can usually retrieve images in the same category as that of the query image.

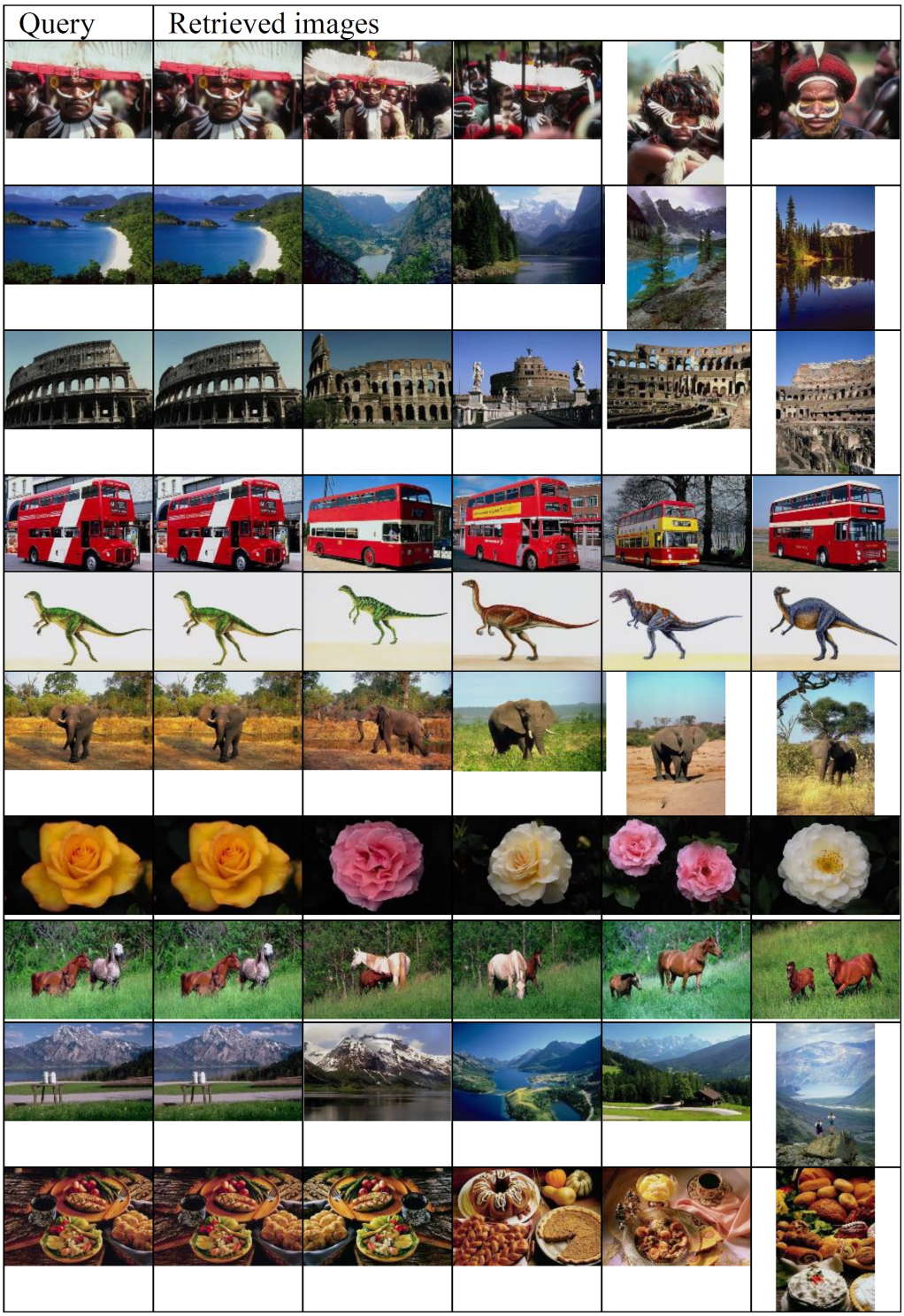

Figure 9. Selected result of retrieved images from 10,000 Images.

6.7. Protection against an Adversary

As our scheme uses a private key *pr* to encrypt the feature vectors of *Alice*, hence, no adversary, including *Bob*, can obtain the correct matching scores if they have no knowledge of the key. In this

experiment, we attempt to determine how difficult it would be for *Bob* to attempt to learn the matching scores using a set of invalid private keys. The first row in Figure 10 shows the retrieved image under the valid private key. The remaining rows show the retrieved images under invalid keys. The first column represents the provided image queries.

Figure 10. Effect of private key on security image matching.

7. Conclusions

Conducting image matching while preserving confidentiality is a challenging task. This paper presents a secure scheme that evaluates similarities between image collections of two parties without compromising their privacy. We utilized the homomorphic properties to design a secure protocol to achieve cosine similarity between two feature vector sets. Specifically, we used SURF descriptor to extract feature vectors. Our proposed framework for secure private image matching is not limited to a specific feature vector and instead can work under different features. We improve the performance of the matching time to scale for massive databases. Particularly, we apply the well-known *k*-means clustering technique to select representative descriptors for each image because clustering selects fewer descriptors. Thus, distance calculation could be largely reduced and consequently decrease matching costs. The practical value of our work is demonstrated through several experimental results. Our formula implementation over 10,000 images shows that SEPIM generates only minimal losses in search time and precision when placed against existing image search methodologies for plaintexts. Following this line of research, our future work will intend to apply search access control, as it is a great characteristic for the secure image search method and the data owner has greater control over those who can employ the search process.

Acknowledgments: This work is supported by National 973 Fundamental Basic Research Program of China under grant No. 2014CB340600.

Author Contributions: All authors extensively discussed the contents of this paper and contributed to its preparation. Zaid Ameen Abduljabbar, Hai Jin, and Ayad Ibrahim proposed and developed the model, drafted the manuscript, and performed experiments. Zaid Alaa Hussien and Mohammed Abdulridha Hussain performed the literature overview and analyzed results. The revisions of this manuscript were done by Salah H. Abbdal and Deqing Zou.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, M.; Cao, N.; Yu, S.; Lou, W. FindU: Privacy-preserving personal profile matching in mobile social networks. In Proceedings of the 2011 IEEE International Conference on Computer Communications, Shanghai, China, 10–15 April 2011; pp. 2435–2443.
2. Zhang, R.; Zhang, Y.; Sun, J.; Yan, G. Fine-grained private matching for proximity-based mobile social networking. In Proceedings of the 2012 IEEE International Conference on Computer Communications, Orlando, FL, USA, 25–30 March 2012; pp. 1969–1977.
3. Dai, Q.; Kauffman, R. Business models for Internet-based E-procurement systems and B2B electronic markets: An exploratory assessment. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 6 January 2001; pp. 1–10.
4. Frikken, K. Privacy-preserving set union. In *Applied Cryptography and Network Security*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4521, pp. 237–252.
5. Li, Y.; Tygar, J.; Hellerstein, J. Private matching. In *Computer Security in the 21st Century*; Springer: New York, NY, USA, 2005; pp. 25–50.
6. Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L.V. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* **2008**, *110*, 346–359.
7. Truong, D.D.; Ngoc, C.S.; Nguyen, V.T.; Tran, M.T.; Duong, A.D. Local descriptors without orientation normalization to enhance landmark regconition. In *Knowledge and Systems Engineering*; Springer International Publishing: Madrid, Spain, 2014; Volume 244, pp. 401–413.
8. Nguyen, H.; Bai, L. Cosine similarity metric learning for face verification. In *Computer Vision—ACCV 2010*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6493, pp. 709–720.
9. Choi, S.; Ghinita, G.; Bertino, E. Secure mutual proximity zone enclosure evaluation. In Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas, TX, USA, 4–7 November 2014; pp. 133–142.
10. Freedman, M.; Nissim, K.; Pinkas, B. Efficient private matching and set intersection. In *Advances in Cryptology—EUROCRYPT 2004*; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3027, pp. 1–19.
11. Hazay, C.; Nissim, K. Efficient set operations in the presence of malicious adversaries. *J. Cryptol.* **2012**, *25*, 383–433.
12. Hong, J.; Kim, J.W.; Kim, J.; Park, K.; Cheon, J.H. Constant-Round Privacy Preserving Multiset Union. Cryptology ePrint Archive. Available online: http://eprint.iacr.org/ (accessed on 13 Augest 2011).
13. Kissner, L.; Song, D. Privacy-preserving set operations. In *Advances in Cryptology—CRYPTO 2005*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3621, pp. 241–257.
14. Jarecki, S.; Liu, X. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *Theory of Cryptography*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5444, pp. 577–594.
15. Hazay, C.; Lindell, Y. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *Theory of Cryptography*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4948, pp. 155–175.
16. De Cristofaro, E.; Gasti, P.; Tsudik, G. Fast and private computation of cardinality of set intersection and union. In *Cryptology and Network Security*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7712, pp. 218–231.
17. Ferreira, B.; Rodrigues, J.; Leitao, J.; Domingos, H. Privacy-preserving content-based image retrieval in the cloud. In Proceedings of the 34th Symposium on Reliable Distributed Systems (SRDS), Montreal, QC, Canada, 28 September–1 October 2015; pp. 11–20.

18. Shashank, J.; Kowshik, P.; Srinathan, K.; Jawahar, C. Private content based image retrieval. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008.
19. Agrawal, R.; Evfimievski, A.; Srikant, R. Information sharing across private databases. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, CA, USA, 10–12 June 2003; pp. 86–97.
20. Lu, W.; Swaminathan, A.; Varna, A.L.; Wu, M. Enabling search over encrypted multimedia databases. *SPIE Proc.* **2009**, *7254*, 18–29.
21. Perronnin, F.; Liu, Y.; Sanchez, J.; Poirier, H. Large-scale image retrieval with compressed Fisher vectors. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; pp. 3384–3391.
22. Douze, M.; Ramisa, A.; Schmid, C. Combining attributes and Fisher vectors for efficient image retrieval. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 20–25 June 2011; pp. 745–752.
23. Lu, W.; Varna, A.L.; Wu, M. Confidentiality-preserving image search: A Comparative study between homomorphic encryption and distance-preserving randomization. *IEEE Access* **2014**, *2*, 125–141.
24. Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P. *Computer Vision—ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5–11 2010, Proceedings, Part IV*; Springer: Berlin, Germany, 2010; pp. 778–792.
25. Desai, A.; Lee, D.J.; Ventura, D. An efficient feature descriptor based on synthetic basis functions and uniqueness matching strategy. *Comput. Vis. Image Underst.* **2016**, *142*, 37–49.
26. Mukherjee, D.; Wu, Q.M.J.; Wang, G. A comparative experimental study of image feature detectors and descriptors. *Mach. Vis. Appl.* **2015**, *26*, 443–466.
27. Desai, A.; Lee, D.J.; Ventura, D. Matching Affine Features with the SYBA Feature Descriptor. In *Advances in Visual Computing: 10th International Symposium, ISVC 2014, Las Vegas, NV, USA, December 8–10, 2014, Proceedings, Part II*; Springer International Publishing: Cham, Switzerland, 2014; pp. 448–457.
28. Lowe, D. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110.
29. Chao, J.; Huitl, R.; Steinbach, E.; Schroeder, D. A novel rate control framework for SIFT/SURF feature preservation in H.264/AVC video compression. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 958–972.
30. Yue, Y.; Finley, T.; Radlinski, F.; Joachims, T. A Support vector method for optimizing average precision. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherland, 23–27 July 2007; pp. 271–278.
31. Goldreich, O.; Micali, S.; Wigderson, A. How to play ANY mental game. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 25–27 May 1987; pp. 218–229.
32. Goethals, B.; Laur, S.; Lipmaa, H.; Mielikainen, T. On private scalar product computation for privacy-preserving data mining. In *Information Security and Cryptology—ICISC 2004*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3506, pp. 104–120.
33. Goldwasser, S.; Micali, S.; Rackoff, C. The knowledge complexity of interactive proof-systems. In Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, Providence, RI, USA, 6–8 May 1985; pp. 291–304.
34. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99*; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1592, pp. 223–238.
35. Spencer, F.; Alok, D.; Lee, D.J.; Ventura, D.; Wilde, D. Efficient Tree-Based Feature Descriptor and Matching Algorithm. *J. Aerosp. Inf. Syst.* **2014**, *11*, 596–606.
36. Venkatasubramanian, S. Measures of anonymity. *Privacy-Preserving Data Mining: Models and Algorithms*; Springer US: New York, NY, USA, 2008; Volume 34, pp. 81–103.
37. Corel Test Set. Available online: http://wang.ist.psu.edu/_jwang/test1.tar (accessed on 10 February 2008).
38. Liao, W. Parallel *k*-Means Data Clustering. Available online: http://www.ece.northwestern.edu/wkliao/Kmeans/index.html (accessed on 5 December 2013).