Towards Secure 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
zaidalsulami@yahoo.com, hjin@hust.edu.cn
[2] University of Basrah, Basrah, Iraq
[3] Southern Technical University, Basrah, Iraq

Abstract. Currently, image matching is being used in many daily life applications such as *content-based image retrieval* (CBIR), computer vision, and near duplicate images. Hence, a number of matching methods have been developed. However, most proposed methods do not address the challenges involved when confidential images are used in image matching between two security agencies. Thus, interest to develop a secure method, particularly one that can be used in privacy-preserving image matching, is growing. This paper addresses the challenge of privacy-preserving image matching between two parties where images are confidential. 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. The method can work without using any image encryption, sharing, and trusted third party. We conduct several empirical analyses on real image collections to demonstrate the performance of our work.

Keywords: Secure private image matching · Feature protection · Secure multiparty computing · Surf descriptors · Homomorphic encryption

1 Introduction

Digital images have become a significant part of our lives because of the development of the Internet and the growing demand from various multimedia fields. This demand raises the need for efficient and robust *private image matching* (PIM) methods in many real-world applications, including social media [1,2] business community [3], and e-health [4]. 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, PIM method is a set of operations through which two parties determine their common matching values without disclosing extra information. Hence, PIM only requires the magnitude of similarity rather contents similarity.

X. Huang et al. (Eds.): GPC 2016, LNCS 9663, pp. 301–315, 2016.
DOI: 10.1007/978-3-319-39077-2_20

According to [5], *private matching* (PM) can be classified into three scenarios. In the first scenario, the parties involved, namely *Alice* and *Bob*, both must 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 exist. The third scenario seeks to determine the number of common elements rather than whether values match exactly. All 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 not to disclose the query image and the database, and has the former learn only of the existence of any commonality of image matching values (second scenario). Such a process is referred to as *secure private image matching* (SPIM).

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 scheme uses cosine similarity [6], a well-known metric to score matching images, and employs homomorphic encryption [7] to protect the confidentiality of descriptors. 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.

Most feature vectors are either global vectors, such as global color histogram or local vectors such as *scale-invariant feature transform* (SIFT) descriptors

[8,9] and *speeded up robust features* (SURF) descriptors [10,11]. The first model generates an extreme compressed feature vector for each image. Such 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. In this paper, we will focus on local features model, which has the advantage of identifying local similarities, e.g., scenes and objects.

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 SPIM 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.

The rest of this paper is organized as follows. Related works are reviewed and discussed in Sect. 2. Section 3 introduces the security requirements and problem definition. Section 4 provides the proposed scheme. Experimental results are provided in Sect. 5, and conclusions and future works are drawn in Sect. 6.

2 Related Works

Ever since Freedman et al. [12] brought up the first solution using 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 term of computational cost. 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 private image matching will also be highlighted.

The important factors in the field of PM are the protocol of *private set union* (PSU) [4,13,14] and *private set intersection* (PSI) [12,15–17], respectively. Cristofaro et al. [18] reveal that these two approaches do not provide adequate privacy on the server end and thus, a server could compromise privacy. In [18], a scenario is proposed where users are allowed to learn only the magnitude of the shared values instead of the exact values. Such scenario uses the *Private Set Union Cardinality* (PSI-CA) and a third-party server. Similarly, Lu et al. [19] proposed a system to search encrypted multimedia databases stored on a server maintained by a third-party service provider. Under both [18,19], the server should not know its stored data and using third-party services where communication has a way through it, it does probably not keep maintaining the privacy aspect of the matching values. Our work obviates the use of any third party for security purposes.

Shashank et al. [20] applied *private information retrieval* (PIR) techniques to protect the privacy of the query image when searching over a public database. However, such method assumes that the database is public when such database is supposed to be private. The proposed methods in [18–20] are also not suitable for evaluating similarity. Both approaches can achieve an exact match, thereby limiting the ability to develop efficient solutions.

In [21], 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 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.

3 Security Definition and Problem Statement

3.1 Security Definition

Our security definition follows the *secure multiparty computing* (SMC) definition of Goldreich et al. [22] and private matching (second scenario) [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 to only queried party. No additional information is disclosed and information that can be inferred from its own input is avoided.

Table 1. Common symbols used

Symbol	Meaning
N	Size of descriptors
M	Number of images in *Bob's* collection
K	Number of descriptors of *Alice's* image
P	Number of descriptor of each one of *Bob's* images

3.2 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 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 SPIM. Formally, SPIM is defined as

$$SPIM(I, D) = \alpha_1, \alpha_2, ..., \alpha_m \tag{1}$$

SPIM 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 [6]. The cosine similarity between vectors v_1 and v_2 of size n can be defined as follow:

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

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

$$\|v\| = \sqrt{\sum_{i=1}^{n} v[i]^2} \tag{3}$$

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

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

Here

$$\overrightarrow{V}[i] = \frac{v[i]}{\|v\|} \tag{5}$$

Given two images, Im_1 and Im_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 while preserving privacy.

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

Algorithm 1. Insecure Image Distance Calculation

Input: two feature vectors $F_1 = \{v_1, v_2, \ldots, v_k\}$ and $F_2 = \{s_1, s_2, \ldots, s_p\}$ 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. *Alice's* image

Alice image					
F1			$\overrightarrow{F}_2$		
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

the last three columns are their corresponding normalized versions. Similarly, Table 3 illustrates the collection of *Bob*, which consists of two images. Also this table is interpreted in the same way as Table 2.

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. The distance between F_1 and F_2 can be calculated as follows:

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

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

Table 3. *Bob's* collection

Bob collection of two images											
F_2			F_3			$\overrightarrow{F}_2$			$\overrightarrow{F}_3$		
s_2	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

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 [23] 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 SPIM.

3.3 Secure Dot Product Based on Homomorphic Encryption

Homomorphic encryption is a probabilistic public key encryption [7,23]. 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 [24] 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 [25] 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., $y_i = z_i^{vi}, foralli = 1, ..., n$). 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 Sect. 1 with respect to privacy-preserving.

4 Our Proposed Scheme

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

4.1 Feature Extraction

In this paper, we utilize the SURF algorithm [10,11], which is a novel scale and rotation-invariant detector and descriptor. SURF approximates or even outperforms previously proposed SIFT algorithm [8,9], which is patented, with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster. Generally speaking, 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 time 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.

Formally, given the image Im, 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. Figure 1 illustrates the interest points of Lena image and their counterparts in the same image after rotation.

4.2 Secure Private Image Matching (SPIM)

The implementation of SPIM utilizes the homomorphic encryption to evaluate similarity. The main steps are highlighted in Algorithm 2. Our proposed protocol distributes scores calculation between the two participant parties and is composed of two phases, initialization and matching phases. 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 assessment of the cosine similarity.

We demonstrate the proposed scheme using SURF descriptors in this paper, although this scheme is applicable to other feature vectors. 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*. Once *Bob* receives *Alice's* encrypted vectors, he employs the *secure_dot_product* subroutine (as explained in Algorithm 3) to

Fig. 1. SURF interest point of two images

return the dot product matrix of the input vector set and the feature vector set of each image in his collection. The details of the above listed subroutine are explained in Subsect. 3.3. Without loss of generality and to make the presentation clearer, we assume that all *Bob's* images have the same number p of descriptors. At the second round, *Alice* uses her private key to decrypt the dot product terms and obtain the actual values, which will be employed in assessing the similarity scores as explained in Algorithm 1. Hence, without the knowledge of *Alice's* private key no adversary is able to get the right matching scores, even the *Bob*.

4.3 Time and Communication 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, the encryption represents the most expensive operation, which is bounded by *O(k)*, where k is the number of descriptors in the input image. At *Bob's* side, the secure dot product subroutine runs m times, and each time it takes the complexity of *O(k.p)*. Thus the overall computing time complexity of this step is *O(m.k.p)*. Decryption represents the most expensive operation in the second round of *Alice's* side, which bounded by *O(m.k.p)* operations. With respect to the communication cost, we can summarize it as follows: in the first round, *Alice* sends $k.n$ values to *Bob* and *Bobs* sends back $m.p.k$ values to *Alice*. Suppose that each value has b-bit long, then the total communication cost is bounded as: $O(b(k.n + m.p.k))$ bits.

5 Experimental Results

In this section, we report the experimental results of the proposed scheme on a real image database containing 1000 color images from the Corel dataset [26]. The images are grouped by content into 10 categories. Each category contains 100 images. These categories include *African, Beach, Architecture, Buses,*

Algorithm 2. Secure 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).
-Send pk 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 Eq. (5), for $i = 1, ..., k$, and replace it with v_i in F.
Bob:
For each image $Im_j \in D, \forall_j = 1, ..., m$
- Use SURF algorithm to extract the feature vector set $F_j = \{s_1, s_2, ..., s_p\}$.
- Compute $\overrightarrow{s}_i$ as in Eq. (5), for $i = 1, ..., p$, and replace it with s_i in F_j.
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:
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$//m$

Algorithm 3. 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 **do**
      $Dot_{ij} = (Z_{it}^{v_{jt}}) \times Dot_{ij}$
    Endfor//t
  EndFor//p
Endfor//k

Dinosaurs, Elephants, Flowers, Horses, Mountain, and *Food*. Image sizes are either 256×384 or 384×256. 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. We use MATLAB R2008a to implement our experiments. We use Java class to implement Paillier cryptosystem. For the SURF descriptors, the size of each descriptor is 64 elements, i.e., $n = 64$. The normalized vectors are scaled by a user specific factor to convert the normalization (between 0 and 1) into an integer numbers because the encryption function is applied only on integer values. The SURF vectors are already normalized to vector units and thus not require normalization.

5.1 Effectiveness

In this experiment, we test the ability of our proposed scheme to retrieve the most similar images to the provided query. Figure 2 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 their corresponding query. The columns show that our scheme can usually retrieve images in the same category as that of the query image.

5.2 Efficiency

In this experiment, we investigate the performance of our proposed scheme in term of matching time. Our scheme requires n exponentiations and n homomorphic additions to compute the distance between each two vectors. Such expensive operations cause our scheme to become slower than the non-secure scheme. Figure 3 illustrates the average time cost of our scheme against the non-secure scheme. In both cases, results are drawn as the number of image queries increased. Every query image is matched against 1000 images. The average time cost of our scheme to match a single query is about 3.4 s, while the other scheme requires 0.48 s. The additional time cost of our work can be considered as a reasonable cost for achieving a secure matching. Our ongoing research focuses on reducing the feature vector set of each image to improve the efficiency.

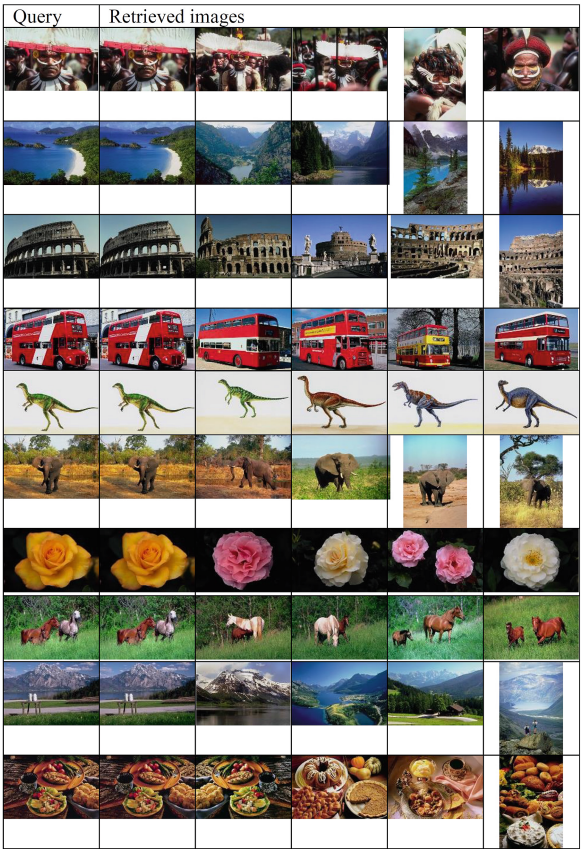

Fig. 2. Selected result of retrieved images

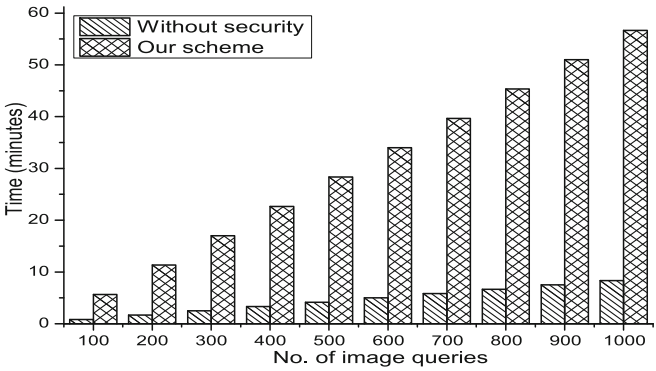

Fig. 3. Matching time

5.3 Protection Against an Adversary

As our scheme uses a private key 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 Fig. 4 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.

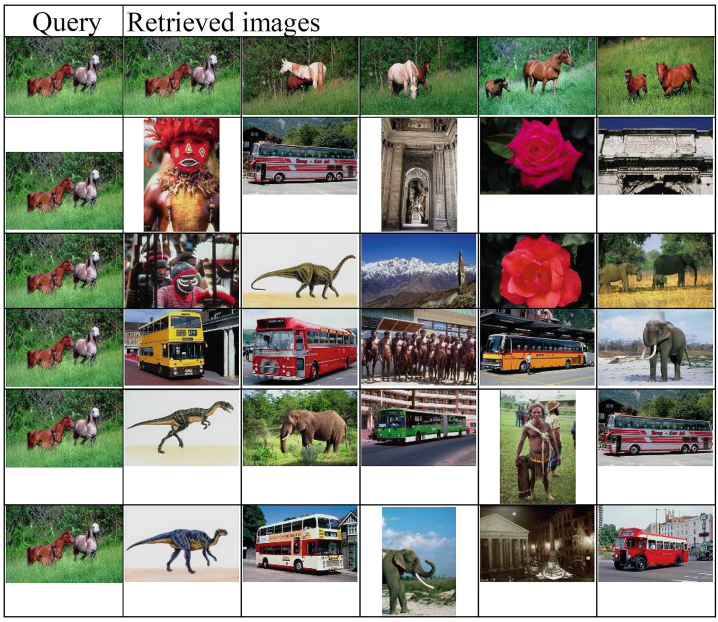

Fig. 4. Effect of private key on security image matching

6 Conclusion

Conducting image matching while preserving confidentiality is a challenging task. This paper presents a secure scheme that evaluate similarity between image collections of two parties without compromising their privacy. We utilize the homomorphic properties to design a secure protocol to achieve cosine similarity between two feature vector sets. Specifically, we use SURF descriptor to extract feature vectors. Interestingly, our proposed framework for secure private image matching is not limited to a specific feature vectors and instead can work under different features. The practical value of our work is demonstrated through several experimental results. Following this line of research, our future work will

attempt to improve the performance of the matching time to scale for massive databases. We intend to apply clustering techniques to select representative descriptors for each image because clustering selects fewer descriptors. Thus, distance calculation could be largely reduced and consequently decreasing matching costs.

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

References

1. Li, M., Cao, N., Yu, S., Lou, W.: FindU: privacy-preserving personal profile matching in mobile social networks. In: Proceedings of 2011 INFOCOM, Shanghai, China, pp. 2435–2443. IEEE Press, April 2011
2. Zhang, R., Zhang, Y., Sun, J., Yan, G.: Fine-grained private matching for proximity-based mobile social networking. In: Proceedings of 2012 INFOCOM, Orlando, FL, USA, pp. 1969–1977, March 2012
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, pp. 1–10, January 2001
4. Frikken, K.: Privacy-preserving set union. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 237–252. Springer, Heidelberg (2007)
5. Li, Y., Tygar, J., Hellerstein, J.: Private matching. In: Lee, D.T., Shieh, S.P., Tygar, J.D. (eds.) Computer Security in the 21st Century, pp. 25–50. Springer, USA (2005)
6. Nguyen, H.V., Bai, L.: Cosine similarity metric learning for face verification. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010, Part II. LNCS, vol. 6493, pp. 709–720. Springer, Heidelberg (2011)
7. 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, SIGSPATIAL 2014, pp. 133–142. ACM, New York (2014)
8. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
9. 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. Circ. Syst. Video Technol. **25**(6), 958–972 (2015)
10. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
11. Truong, D.-D., Ngoc, C.-S.N., Nguyen, V.-T., Tran, M.-T., Duong, A.-D.: Local descriptors without orientation normalization to enhance landmark regconition. In: Huynh, V.N., Denoeux, T., Tran, D.H., Le, A.C., Pham, B.S. (eds.) KSE 2013, Part I. AISC, vol. 244, pp. 401–413. Springer, Heidelberg (2014)
12. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
13. Hazay, C., Nissim, K.: Efficient set operations in the presence of malicious adversari. J. Cryptology **25**(3), 383–433 (2012)

14. Hong, J., Kim, J.W., Kim, J., Park, K., Cheon, J.H.: Constant-round privacy preserving multiset union. Cryptology ePrint Archive, Report 2011/138 (2011). http://eprint.iacr.org/
15. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
16. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
17. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
18. De Cristofaro, E., Gasti, P., Tsudik, G.: Fast and private computation of cardinality of set intersection and union. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 218–231. Springer, Heidelberg (2012)
19. Lu, W., Swaminathan, A., Varna, A.L., Wu, M.: Enabling search over encrypted multimedia databases. In: Proceedings of Media Forensics and Security, SPIE, San Jose, CA, USA, vol. 7254, pp. 18–29 (2009)
20. Shashank, J., Kowshik, P., Srinathan, K., Jawahar, C.: Private content based image retrieval. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, pp. 1–8. IEEE Press, June 2008
21. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD 2003, pp. 86–97. ACM, New York (2003)
22. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987, pp. 218–229. ACM, New York (1987)
23. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
24. 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, STOC 1985, pp. 291–304. ACM, New York (1985)
25. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
26. Corel test set. http://wang.ist.psu.edu/_jwang/test1.tar