A Privacy-Preserving Bipartite Graph Matching Framework for Multimedia Analysis and Retrieval

Wei-Ta Chu
National Chung Cheng University
Chiayi, Taiwan
wtchu@ccu.edu.tw

Feng-Chi Chang
National Chung Cheng University
Chiayi, Taiwan
winderif@gmail.com

ABSTRACT
The emergence of cloud computing provides an unlimited computation/storage for users, and yields new opportunities for multimedia analysis and retrieval research. However, privacy of users, e.g., search intention, may be leaked to the server and maliciously utilized by companies or individuals with animus. This paper presents a privacy-preserving multimedia analysis framework based on a widely-adopted structure, i.e., bipartite graph, so that multimedia analysis and retrieval in the encrypted domain is enabled. This work aims to keep the server unaware of what the user wants to retrieve, and at the same time take advantage of the server's computation power. Homomorphic encryption schemes and communication protocols in the encrypted domain are integrated to facilitate bipartite graph construction and implement the Hungarian algorithm to find the best matching. Two applications, video tag suggestion and video copy detection, are developed on top of the privacy-preserving framework, and the evaluation results demonstrate that performance obtained in the encrypted domain is comparable with that obtained in the plain text domain.

Categories and Subject Descriptors
H.2.0 [**Database Management**]: General – Security, integrity, and protection. H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *retrieval models, search process, selection process.*

General Terms
Algorithms, Experimentation, Security.

Keywords
Bipartite graph matching, privacy preserving, Paillier cryptosystem, DGK cryptosystem, garbled circuit, Hungarian algorithm.

1. INTRODUCTION
With the advancement of network throughput, distributed storage, and flourishing of social media platforms, the web has become the largest database that conveys almost unlimited amount of information. Technologies of parallel processing, efficient indexing, and multiple cores, further facilitate the development of a cloud computing environment, so that the web can be viewed as a powerful storage and computation platform. This trend has especially urged multimedia retrieval research due to its demand for large storage and immense computation. It can be envisioned that more multimedia data would be uploaded to the cloud environment, not only for storage or sharing, but also for retrieval or analysis purposes.

Despite popularity and powerful computation of a cloud environment, storing personal data or uploading image queries in an open environment give rise to severe privacy issues. Recently, although researchers have studied privacy-preserving methodologies in many fields, such as privacy-preserving data mining [8] and text document search in the encrypted domain [9], relatively fewer studies have been conducted for privacy-preserving multimedia retrieval. In a client-server content-based image retrieval architecture, users upload their query images to the server, and request the server's database and computation power to conduct large-scale image retrieval. These query images may contain users' personal information, or from multiple queries the server may collect statistics that may leak users' private matters. Privacy-preserving multimedia retrieval, therefore, gives rise to an exacting technical challenge: After receiving the data sent from users, the server should not be able to infer what have been uploaded, but it should be able to find from its database the ones similar to the uploaded data. For the server side, the server may just answer YES or NO to users about whether it holds data similar to queries, without revealing what exactly it has in the database. Depending on application fields, sometimes data available on the server are collected with huge efforts and are very valuable, such as medical images that are highly privacy sensitive and big multimedia that embed implicit correlation between heterogeneous media. Therefore, in this work we focus on a combination of multimedia analysis and privacy-preserving protocols to build a two-way privacy system, i.e., protecting both the client's and the server's privacy.

Figure 1(a) shows a general multimedia retrieval framework. Alice uploads a query in the representation of feature vectors to Bob, who then calculates similarity between the query and data from his database. Bob sends Alice the data with largest similarity to the query or just an answer of YES or NO. For the server side, Bob needs to calculate similarity and performs sorting to find best results to be sent back to the client. Figure 1(b) shows a general privacy-preserving multimedia retrieval framework, where operations in the shaded area are in the encrypted domain. Alice sends encrypted feature vectors to Bob. Bob then operates similarity calculation and sorting in the encrypted domain, and sends encrypted retrieval results to Alice, who later decrypts the returned results to get the plain text results. In this paper, we propose a bipartite graph framework in the encrypted domain and adopt private value comparison protocols to implement the shaded area.

ICMR'15, June 23–26, 2015, Shanghai, China.
Copyright 2015 ACM 978-1-4503-3274-3/15/06 $15.00
http://dx.doi.org/10.1145/2671188.2749286

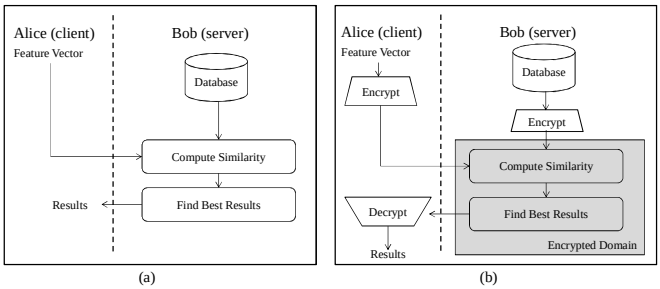

Figure 1. General multimedia retrieval frameworks in the (a) plain text domain and in the (b) encrypted domain.

Contributions of this paper are twofold. First, with encryption techniques we design a privacy-preserving multimedia analysis framework on top of bipartite graph matching. Cryptosystems with homomorphic encryption operations and communication protocols for comparing private numbers are introduced to enable privacy protection. Second, based on the general framework, two applications of video tag suggestion and video copy detection are developed to validate the proposed framework and embody privacy-preserving multimedia analysis that has not widely studied before.

The rest of this paper is organized as follows. Section 2 presents related works. Section 3 first introduces the basics of bipartite graph matching, and then describes details of graph construction. Section 4 provides the details of privacy-preserving bipartite graph matching. With the proposed framework, two applications are described in Section 5. Section 6 presents evaluation results to validate the proposed framework, followed by concluding remarks in Section 7.

2. RELATED WORKS

2.1 Privacy-Preserving Multimedia Retrieval

Shashank et al. [24] proposed one of the first privacy-preserving image retrieval systems, which was originally designed for private information retrieval with index structures like hierarchical structure and hash. With cryptographic techniques, including order preserving encryption and randomized hash functions, Lu et al. [11] developed secure and efficient image retrieval system where query images, data images, and the corresponding index structures, are processed in the encrypted domain. For secure image feature extraction, Hsu et al. [12] proposed privacy-preserving SIFT (Scale-Invariant Feature Transform) extraction. Not specifically for multimedia retrieval, privacy-preserving face recognition is a highly related topic and was first proposed by Erkin et al. [4]. They introduced cryptographic tools to project face images into eigenfaces and measure face similarity. Sadeghi et al. [7] modified the encryption schemes mentioned in [4] with garbled circuits, and made privacy-preserving face recognition more efficient. Generally, although cryptographic techniques may be ready for data analysis and manipulation in the encrypted domain, potential and embodiment of privacy-preserving multimedia retrieval systems are still not well investigated.

2.2 Bipartite Graph Matching for MM Retrieval

The processing pipeline of a multimedia retrieval system mainly includes feature extraction, indexing, similarity measurement, and result ranking. Existing works [11][12][24] primarily concentrate on either secure index structure or feature extraction. In this paper, we investigate similarity measurement between images in the encrypted domain. Although selection of similarity measurement is application-dependent, we notice that bipartite graph matching has been widely adopted in many video analysis researches. In such works, videos are first divided into shots, and shots respectively from two videos are viewed as two disjoint sets in a bipartite graph. Similarity between two videos is then measured by finding the minimum cost (or maximum weight) bipartite graph matching. Conceptually, the minimum cost (maximum weight) matching denotes the minimum distance (maximum similarity) we can obtain among all possible matchings between two sets of video shots.

More specifically, the work in [13] constructed a bipartite graph between a query video clip and a targeted video, and by finding the maximum cardinality matching, in the targeted video the clip similar to the query clip is located. The same research group also adopted this idea to detect gradual transition in videos [14], where parts of pixels were sampled from two frames and were viewed as disjoint node sets to construct a bipartite graph, followed by similarity measured by finding the maximum cardinality matching. Xu et al. [15] constructed a bipartite graph to describe similarity between keyframes of two videos. They slightly modified the standard matching algorithm to achieve fast near-duplicate video detection. Kim et al. [16] determined the maximum cardinality matching based on a bipartite graph describing keyframe similarity, and demonstrated robust performance on video copy detection. Based on a similar idea, Bai et al. [17] conducted rush video summarization after measuring similarity between video shots. Images can be described by local patches, which can be treated as nodes in the bipartite graph, and thus image similarity can be measured by the matching situation [18]. 3D models can be represented by sets of 2D views. A bipartite graph can be constructed based on sets of views, and the same idea is used to measure similarity between 3D models [19]. Chu et al. [20] described video keyframes and text-based tags as bags of visual words, which were then treated as nodes to construct a bipartite graph linking two modalities. The maximum weight matching was then determined to facilitate tag suggestion and localization.

We clearly see the importance of bipartite graph matching on multimedia analysis and retrieval. Our work investigates using encryption techniques to construct bipartite graphs and enabling the well-known Hungarian algorithm [1] in the encrypted domain to find the best matching. The reasons to develop the framework based on bipartite graphs are twofold. First, bipartite graphs are widely adopted in various multimedia analysis researches, and generality and effectiveness have been extensively demonstrated. Second, processes for graph construction and graph matching are mostly linear operations, which can be effectively implemented in the encrypted domain by cryptosystems and communication protocols.

Specifically, the Paillier cryptosystem [2] is utilized to construct homomorphic operations for evaluating distances between nodes, and weighted bipartite graphs can be constructed. Homomorphic operations associated with a comparison protocol, which compares encrypted values through communication between user and server, are then employed to implement the Hungarian algorithm. Computation intensive jobs (in the encrypted domain) are taken by the server, such as graph construction and graph matching. The meaning of encrypted matching results can be realized only when they are decrypted by the user. This is why user's privacy is protected, and computation power of the server is well adopted in the meantime.

3. PRIVACY-PRESERVING BIPARTITE GRAPH CONSTRUCTION

3.1 Basics of Bipartite Graph Matching

Formulation of Bipartite Graph Matching. Consider two sets $\mathcal{A}$ and $\mathcal{B}$ that both consist of n elements (nodes). Let $c_{i,j}$ denote the cost of the assignment $i \to j$, $i \in \mathcal{A}$, $j \in \mathcal{B}$. Let the binary variable $x_{i,j}$ denote an assignment:

$$x_{i,j} = \begin{cases} 1 & \text{if the element } j \in \mathcal{B} \text{ is assigned to the element } i \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases}$$

The problem of minimum cost bipartite graph matching is defined as

$$minimize \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j}$$

$$\text{subject to } \sum_{i=1}^{n} x_{i,j} = 1, \forall j = 1, 2, ..., n \text{ and}$$
$$\sum_{j=1}^{n} x_{i,j} = 1, \forall i = 1, 2, ..., n.$$

To simplify the discussion, we assume that the cardinalities of $\mathcal{A}$ and $\mathcal{B}$ are the same. In the real case, we can always make this assumption true. If $|\mathcal{A}| = m$, $|\mathcal{B}| = n$, and $m < n$, we set $c_{k,j} = 0$ for $k = m+1, ..., n$. It can be easily seen that, because $\sum_{k=m+1}^{n} \sum_{j=1}^{n} c_{k,j} x_{k,j} = 0$, the minimum cost found based on the newly derived $n \times n$ matrix is the same as that based on the original $m \times n$ matrix.

In real applications, each node of $\mathcal{A}$ and $\mathcal{B}$ is represented by a feature vector, and the cost $c_{i,j}$ of an assignment is usually measured by the Euclidean distance between two vectors. Generally, a high cost $c_{i,j}$ means that features of the ith node of $\mathcal{A}$ and the jth node of $\mathcal{B}$ are very dissimilar.

The Hungarian Algorithm. The minimum cost bipartite graph matching problem can be solved by the Hungarian algorithm originally proposed by Kuhn and Munkres [1]. A variation of this algorithm facilitating fast implementation is described as follows.

Let C be the cost matrix whose elements are $c_{i,j}$, $1 \le i \le n$, $1 \le j \le n$. The Hungarian algorithm proceeds as:

- Step 1: For each row in C, find the smallest entry of the row, and subtract it from all entries of the row. This yields that at least one entry in each row is zero. An entry $c_{i,j}$ equal to zero after subtraction means that assigning the ith node of $\mathcal{A}$ to the jth node of $\mathcal{B}$ yields the least cost.
- Step 2: For each column in C, find the smallest entry of the column, and subtract it from all entries of the column.
- Step 3: Chose the row or column with the minimum positive number of zeros (if there are more than one with this minimum number of zeros, choose arbitrarily). If a row was selected, trace a vertical line through one of the zeros of the selected row. If a column was selected, trace a horizontal line through one of the zeros of the selected column. Trace lines until each zero has at least one line through it. Denote the number of lines k.
  - If $k < n$, find the minimum uncovered number (no line through it), denoted as ν. Subtract ν from every uncovered number. Add ν to every number covered by two lines. Go back to the start of Step 3.
  - If $k = n$, go to Step 4.
- Step 4: Start with the top row, and work downwards to make assignments. An assignment is uniquely made when there is exactly one zero in the row. Once an assignment is made, delete the row and the column containing this zero.
  - If fewer than n assignments can be made, and all the remaining rows contain more than one zero, switch to columns. Start with the leftmost column, and work rightwards to make assignments.
  - Iterate between row assignments and column assignments until as many unique assignments as possible. If n assignments are still not achieved, i.e., no unique assignment can be made either with rows or columns, make one arbitrarily by selecting an entry with a zero.

Privacy-preserving graph construction and matching. To ensure privacy protection, operations of graph construction and the Hungarian algorithm should be conducted in the encrypted domain.

- Graph construction: The most critical operation at this stage is calculating the assignment cost between nodes, which is often measured by the Euclidean distance. We need encrypted subtraction and multiplication operators.
- Steps 1 and 2 of the Hungarian algorithm: The smallest entry should be found first, and thus an encrypted comparison scheme is urgently needed. An encrypted subtraction is then needed.
- Step 3 of the Hungarian algorithm: The encrypted comparison scheme is again applied, and encrypted subtraction and addition are needed.
- Step 4 of the Hungarian algorithm: This step consists of only condition matching. Because feature vectors are encrypted before graph construction, the server just conducts condition matching, and knows nothing about the meaning of an assignment. The number n is known to the server, but this would not reveal the result of graph matching.

In the following, we briefly describe the application scenario of privacy-preserving multimedia retrieval based on the bipartite graph matching framework. Assume that an element of $\mathcal{A}$ is represented by a vector $\boldsymbol{a}_i = (a_1, a_2, ..., a_m)$, $i = 1, ..., n$, and an element of $\mathcal{B}$ is represented by a vector $\boldsymbol{b}_j = (b_1, b_2, ..., b_m)$, $j = 1, ..., n$, and two parties, Alice and Bob, jointly participate in this process. The set $\mathcal{A}$ is uploaded by Alice, who would like to know whether there is a set $\mathcal{B}$ in Bob's database that is identical or similar to the uploaded set. Alice wants to take advantage of Bob's computation power to calculate data similarity, but meanwhile she is not willing to leak anything about the uploaded data to Bob.

Figure 2 shows the framework for privacy-preserving bipartite graph matching. Alice sends encrypted feature vectors to Bob. Bob encrypts his database, constructs the bipartite graph, and finds the minimum cost bipartite graph matching by the Hungarian algorithm in the encrypted domain. The matching results are sent back to Alice, who decrypts and postprocesses them to interpret matching results, e.g., the degree of similarity between nodes or node assignment situations.

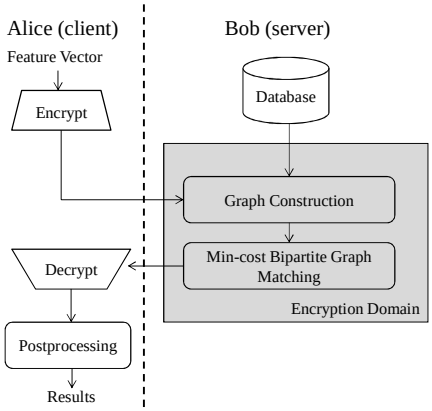

Figure 2. The privacy-preserving bipartite graph matching framework.

3.2 Graph Construction

The central step to construct a bipartite graph is calculating the cost linking any two nodes in two different sets. In the plain text domain, the cost between $\boldsymbol{a}_i = (a_1, a_2, ..., a_m)$ and $\boldsymbol{b}_j = (b_1, b_2, ..., b_m)$ is usually measured by the squared Euclidean distance:

$$\begin{aligned} c_{i,j} &= (b_1 - a_1)^2 + (b_2 - a_2)^2 + \cdots + (b_m - a_m)^2 \\ &= \underbrace{\sum_{k=1}^{m} b_k^2}_{T_1} + \underbrace{\sum_{k=1}^{m} (-2a_k b_k)}_{T_2} + \underbrace{\sum_{k=1}^{m} a_k^2}_{T_3}, \end{aligned} \quad (1)$$

In the following, we use the notation $[s]$ to denote the ciphertext generated by the Paillier cryptosystem [2] based on s. With the

homomorphic property, to compute $[c_{ij}]$, it suffices to compute encrypted T_1, T_2, and T_3, and then

$$[c_{i,j}] = [T_1] \cdot [T_2] \cdot [T_3]. \quad (2)$$

In real applications, Alice sends $[a_1], [a_2], ..., [a_m]$ to Bob, and Bob knows the plain text $b_1, b_2, ..., b_m$ from his database. Therefore, for the first term T_1, Bob can directly compute $\sum_{k=1}^{m} b_k^2$ and encrypt it to obtain $[T_1]$. For the second term T_2, the value $[-2a_k b_k]$ can be calculated by $[a_k]^{(-2b_k)}$, where $(-2b_k)$ is directly computed by Bob, and $[a_k]$ is received from Alice. Bob finally computes $[T_2] = \prod_{k=1}^{m} [-2a_k b_k]$. The third term T_3 is constituted by the sum of squared a_k's. However, Bob only knows $[a_k]$, and $[a_k]^2 \neq [a_k^2]$. Therefore, Bob blindly adds an uniformly random element r_k to a_k to obtain $[x_k] = [a_k + r_k] = [a_k] \cdot [r_k]$. Note that for different a_k's, the random elements r_k's are different. Bob sends $[x_k]$ to Alice for decryption. Alice thus obtains x_k and is able to compute x_k^2 as well as $T_3' = \sum_{k=1}^{m} x_k^2$. She encrypts T_3' and sends $[T_3']$ to Bob, who then computes

$$[T_3] = [T_3'] \cdot \prod_{k=1}^{m} ([a_k]^{(-2r_k)} \cdot [-r_k^2]), \quad (3)$$

because

$$[x_k^2] \cdot [a_k]^{(-2r_k)} \cdot [-r_k^2] = [(a_k + r_k)^2 - 2r_k a_k - r_k^2] = [a_k^2]. \quad (4)$$

After all three terms are obtained, the encrypted squared Euclidean distance between two vectors can be computed by Bob. After evaluating all distances between elements of $\mathcal{A}$ and $\mathcal{B}$, a weighted bipartite graph is constructed.

In multimedia applications where high-dimensional feature vectors may be extracted, the computation and communication costs of the protocol would be high. An improved protocol called *packing* was thus proposed in [7] to reduce the complexity.

Packing. In order to fully take advantage of capacity of one ciphertext, assume that any a_k can be compactly represented by ℓ bits, and a ciphertext is represented by K bits according to the setting of the encryption algorithm. The random value r_k added to form x_k is selected to be represented by $\ell + \sigma$ bits, and thus x_k is represented by $\ell + \sigma$ bits. Therefore, the number of encrypted x_k that can be packed into a single ciphertext is $K' = \left\lfloor \frac{K}{\ell+\sigma} \right\rfloor$. For $k = 1, ..., K'$, encrypted a_k's with random perturbation r_k's in a pack is

$$[x] = \left[\sum_{k=1}^{K'} 2^{(\ell+\sigma)(k-1)}(a_k + r_k)\right] = \left[\sum_{k=1}^{K'} 2^{(\ell+\sigma)(k-1)} r_k\right] \prod_{k=1}^{K'} \left[a_k\right]^{2^{(\ell+\sigma)(k-1)}}. \quad (5)$$

Bob sends $[x]$ to Alice, who decrypts it to obtain $x = (x_{K'}...x_1)$ with $x \in \{0,1\}^{\ell+\sigma}$. By accumulating all necessary x's, she can compute $T_3' = \sum_{k=1}^{m} x_k^2$ and sends the encrypted $[T_3']$ to Bob.

The packing scheme reduces the number of communication from m to $\lceil \frac{m}{K'} \rceil$. In our implementation, a ciphertext obtained by the Paillier cryptosystem is represented by 1024 bits ($K = 1024$), the a_k's are represented by at most 15 bits ($\ell = 15$), and the random value r_k is selected to be represented by 95 bits $(\ell + \sigma) = (15 + 80) = 95$. Therefore, at most $K' = \left\lfloor \frac{K}{\ell+\sigma} \right\rfloor = \left\lfloor \frac{1024}{95} \right\rfloor = 10$ ciphertexts are packed together.

Pre-computation. Alternatively, an assumption can be reasonably made to alleviate the communication cost without leaking client's privacy. Assume that Alice knows that Bob needs the encrypted squared norm of $\boldsymbol{a}_i = (a_1, a_2, ..., a_m)$ in advance, and thus she can calculate $\sum_{k=1}^{m} a_k^2$ and encrypt it as $[T_3] = \left[\sum_{k=1}^{m} a_k^2\right]$ before sending to Bob.

4. PRIVACY-PRESERVING BIPARTITE GRAPH MATCHING

After graph construction, the Hungarian algorithm operated in the encrypted domain is designed to find the minimum cost matching. Assume that the cost matrix in ciphertext is $[C] = [c_{i,j}]$, $1 \leq i, j \leq n$. At Step 1 (Step 2) of the Hungarian algorithm, the smallest entry in each row (column) should be found first. Consider the first row $[c_{1,j}]$, $1 \leq j \leq n$, for example, we compare encrypted costs $[c_{1,2j+1}]$ and $[c_{1,2j+2}]$, for $1 \leq j \leq \lfloor \frac{n}{2} \rfloor - 1$, by using a comparison protocol that will be described later. After processing all entries, the $\lfloor \frac{n}{2} \rfloor$ smaller entries are then compared by the same procedure again, and so forth. After $\lceil \log_2 n \rceil$ iterations, the minimum entry $[c_{1,j^*}]$ can be found. The minimum value is subtracted from each entry in the row, i.e., $[c_{1,j} - c_{1,j^*}] = [c_{1,j}] \cdot [c_{1,j^*}]^{(-1)}$.

At Step 3 of the Hungarian algorithm, the matrix entries with values equal to encrypted zeros are first found, and then we use as few lines as possible to pass through them. To determine whether a ciphertext is equal to an encrypted zero, a zero checking process is adopted. If the number of lines k is less than the dimension of the cost matrix, we find the minimum uncovered number by the comparison protocol, and then subtract the value from every uncovered number in the encrypted domain, and add the value to every number covered by two lines in the encrypted domain also. If the number of lines k is equal to the dimension of the cost matrix, the Step 4 of the Hungarian algorithm proceeds, where no special encryption operations are required. After Step 4, the minimum cost matching is found, and the matching results are sent to Alice. Alice then decrypts to interpret the meaning of matching, and conducts postprocesses if needed.

In the following, we design the core components of the privacy-preserving Hungarian algorithm. Readers may need to refer details of the cryptosystems in reference.

4.1 Comparison Protocol

Assume that two ciphertext values $[a]$ and $[b]$ known to Bob are ℓ-bit. Bob first computes

$$[z] = [2^\ell + a - b] = [2^\ell] \cdot [a] \cdot [b]^{-1}, \quad (6)$$

where z is a positive $(\ell + 1)$-bit value because $0 \leq a, b < 2^\ell$. If $a < b$, the most significant bit of z, denoted by z_ℓ, is 0. Therefore, by checking z_ℓ, whether $[a]$ is larger than $[b]$ can be immediately determined. The value of z_ℓ can be computed by

$$z_\ell = 2^{-\ell} \cdot (z - (z \mod 2^\ell)). \quad (7)$$

This equation only consists of linear combination (one multiplication and one subtraction) in the encrypted domain, and can be easily conducted by Bob. As Bob knows $[a]$ and $[b]$ and can easily compute $[z]$, now the problem is the computation of $[z \mod 2^\ell]$.

Finding $[\mathbf{z} \mod \mathbf{2}^\ell]$. Erkin et al. [4] design a comparison protocol where Bob seeks help from Alice. First, Bob generates a uniformly random $(\kappa + \ell + 1)$-bit value r. The value κ is a security parameter, and $\kappa + \ell + 1 \ll \log_2 N$, where N is the multiplication result of two prime numbers. Similar to calculation of the third term of eqn. (1), Bob blindly adds the number r to z by doing encrypted domain multiplication:

$$[d] = [z + r] = [z] \cdot [r]. \quad (8)$$

The value $[d]$ is sent to Alice, who decrypts it and reduces d modulo 2^ℓ. The obtained value is then encrypted and returned to Bob.

From the design of eqn. (6), we have $d = z + r \mod 2^\ell$ and

$$(z \mod 2^\ell) = ((d \mod 2^\ell) - (r \mod 2^\ell)) \mod 2^\ell. \quad (9)$$

Alice has sent $[d \mod 2^\ell]$ and Bob knows r. Bob thus can compute

$$[\tilde{z}] = [(d \mod 2^\ell) - (r \mod 2^\ell)] = [(d \mod 2^\ell)] \cdot [(r \mod 2^\ell)]^{-1}. \quad (10)$$

If $d \mod 2^\ell \geq r \mod 2^\ell$, it is safe to take modulo 2^ℓ for $[\tilde{z}]$, and $[\tilde{z}]$ is the right result equal to $[z \mod 2^\ell]$. If $r \mod 2^\ell \geq d \mod 2^\ell$, an underflow will occur when taking modulo 2^ℓ, and thus adding 2^ℓ to $[\tilde{z}]$ would give the right result. So far, Bob could correctly obtain $[z \mod 2^\ell]$ if he could compare two private inputs: $\hat{d} = d \mod 2^\ell$ held by Alice and $\hat{r} = r \mod 2^\ell$ held by Bob.

Comparing private values. To compare private values, the DGK homomorphic encryption scheme is utilized. As mentioned in [5] and [6], comparing with the Paillier system, the DGK scheme provides more efficient encryption as it works in very small plaintext space. This characteristic is especially good for us to compare $\hat{d}$ and $\hat{r}$, which are much smaller than the message needed to be encrypted in the Paillier system. We use $\langle \cdot \rangle$ to denote DGK encryption.

Assume that Alice has run the DGK key generation algorithm and has sent the public key to Bob. Alice takes the bit representation of $\hat{d} = \hat{d}_{\ell-1}\hat{d}_{\ell-2}...\hat{d}_0$, encrypts each bit to obtain $\langle \hat{d}_{\ell-1} \rangle, ..., \langle \hat{d}_0 \rangle$, and sends them to Bob. Bob then chooses $t \in \{1, -1\}$ and computes

$$\langle o_i \rangle = \langle \hat{d}_i - \hat{r}_i + t + 3 \sum_{j=i+1}^{\ell-1} w_j \rangle = \langle \hat{d}_i \rangle \cdot \langle -\hat{r}_i \rangle \cdot \langle t \rangle \cdot \left(\prod_{j=i+1}^{\ell-1} \langle w_j \rangle \right)^3, \quad (11)$$

where $\langle w_j \rangle = \langle \hat{d}_j \oplus \hat{r}_j \rangle$, which can be used because Bob knows $\hat{r}_j$. Specifically, this value can be computed by

$$\langle w_j \rangle = \langle \hat{d}_j \oplus \hat{r}_j \rangle = \langle \hat{d}_j + \hat{r}_j - 2\hat{d}_j\hat{r}_j \rangle = \langle \hat{d}_j \rangle \cdot \langle \hat{r}_j \rangle \cdot \langle \hat{d}_j \rangle^{-2\hat{r}_j}. \quad (12)$$

To avoid $\hat{d} = \hat{r}$, differing bits are appended to both $\hat{d}$ and $\hat{r}$ so that actually $2\hat{d} + 1$ and $2\hat{r}$ are compared.

In the case of $t = 1$, if $\hat{d}$ is larger than $\hat{r}$, all o_i are nonzero. If $\hat{r}$ is larger than $\hat{d}$, then exactly one o_i is equal to zero, which is at the position of the most significant differing bit. When t is selected as -1, the same situation occurs, except that the zero occurs if $\hat{d}$ is larger.

To avoid leaking any statistics of private values in Bob's side, Bob now masks $\langle o_i \rangle$ with a uniformly random number $r_i \in Z_u^*$, where u is the small prime defined in the DGK system, by

$$\langle e_i \rangle = \langle o_i \cdot r_i \rangle = \langle o_i \rangle^{r_i}, \quad (13)$$

and sends them to Alice. Alice decrypts all e_i and checks whether one of them is zero. She encrypts a bit λ, which indicates whether the condition is satisfied, by the Paillier encryption, i.e., $[\lambda]$, and sends it to Bob. There are totally four cases about the selection of t (two choices) and the result $[\lambda]$ (two possibilities). As Bob knows the selection of t, he can derive the correspondence between $[\lambda]$ and the condition after a few communications.

Finally, Bob knows whether $\hat{d}$ is larger than $\hat{r}$, with which he can compute $[z \mod 2^\ell]$, and thus can determine the larger one of two ciphertext values $[a]$ and $[b]$. By embedding this comparison protocol into the Hungarian algorithm, the minimum-cost bipartite graph matching can be determined in the encrypted domain.

Note that the comparison protocol mainly works based on the DGK scheme, and graph construction mentioned in Sec. 3.2 mainly works based on the Paillier scheme. The reason for this difference is that only small plaintext space can be adopted in the DGK cryptosystem, and the dynamic ranges of feature vectors for constructing a graph are too large to be affordable. On the other hand, the encrypted numbers to be compared at the graph matching stage are much smaller, and can be processed by the more efficient DGK cryptosystem.

4.2 Checking Zero

At Step 3 of the Hungarian algorithm, we need to determine which entry is zero in the encrypted domain, which can be done by the comparison protocol. We know $z_\ell = 0 \Leftrightarrow p < q$ in the comparison protocol, where z_ℓ is the most significant bit of z and $z = 2^\ell + p - q$. Assume that we let $p = 0$ and $q = x$, and then we know $z_\ell = 0 \Leftrightarrow 0 < x$. In other words, the value x is not equal to zero. On the other hand, $z_\ell = 1 \Leftrightarrow 0 \geq x$. But x is certainly not negative number, and thus we know $z_\ell = 1 \Leftrightarrow 0 = x$. With this protocol, we are able to compare a private number $[x]$ with $[0]$.

4.3 Garbled Circuit for Minimum Search

Comparing two encrypted numbers in the encrypted domain mentioned above relies on homomorphic operations and the comparison protocol. The computation of exponents and frequent interaction between server and client give rise to high costs. In order to reduce computation and communication, the garbled circuit can be used to improve the efficiency of minimum search, in which operations are based on Boolean circuits [22][23].

The garbled circuit for minimum search is described in the following. Bob wants to find the minimum of encrypted numbers. Note that the garbled circuit works only for plain text, and thus Bob cannot directly employ it to find the minimum. He needs to send encrypted numbers and the setting of the garbled circuit to Alice, and then Alice can evaluate the minimum value using the garbled circuit, encrypt it, and send it back to Bob.

Figure 3 shows the protocol for minimum search with a garbled circuit. Assume that Bob wants to find the minimum of two encrypted numbers $[p]$ and $[q]$, he chooses two random values r_p and r_q, adds these to $[p]$ and $[q]$, and obtains the encrypted values $[y_p]$ and $[y_q]$. Next, Bob sends $[y_p]$ and $[y_q]$ to Alice, and then she can decrypt them to obtain y_p and y_q. With y_i and r_i, the minimum value of encrypted numbers can be determined by using the garbled circuit. Figure 4 shows the garbled circuit for minimum search, including two SUBTRACTION gates, one MINIMUM gate, and one ADDITION gate. In the garbled circuit, p and q are first recovered from y_i and r_i, i.e., $p = y_p - r_p$, and then Alice can obtain the minimum value x_{min}. To protect Bob's information, the garbled circuit runs the blinding protocol using a random number r_{min} provided by Bob. After evaluation of the garbled circuit, Alice obtains the output y_{min} without learning anything about x_{min}. Alice encrypts y_{min} and sends $[y_{min}]$ to Bob. Finally, Bob can use the chosen random number r_{min} to get the encrypted minimum value $[x_{min}] = [y_{min} - r_{min}]$, which is the minimum of $[p]$ and $[q]$.

5. APPLICATIONS

The proposed framework is employed to build two applications: video tag suggestion and localization, and video copy detection.

5.1 Privacy-Preserving Video Tag Suggestion

We slightly modify the framework proposed in [20] to enable privacy-preserving video tag suggestion. This framework simultaneously accomplishes tag suggestion and location, as illustrated in Figure 5. A user uploads his personal video and roughly provides some tags, e.g., the tag t_0 in Figure 5, to describe this video. The goal of this work is to localize user-provided tags into appropriate video shots, and suggest new tags, which are retrieved from the web, to each video shot. Relationships between video keyframes and tags retrieved from the web are described by a bipartite graph, and the best matching is determined to suggest tags best fit the keyframes.

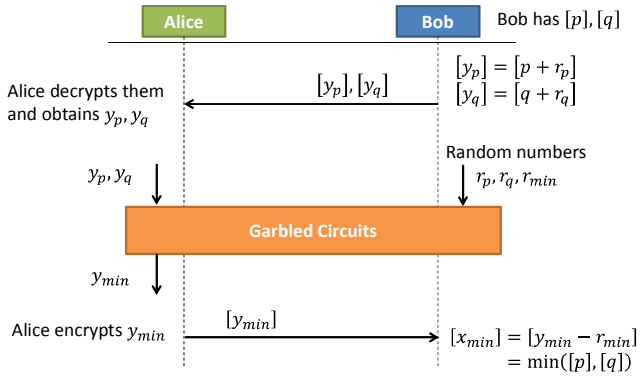

Figure 3. The protocol for minimum search with garbled circuit.

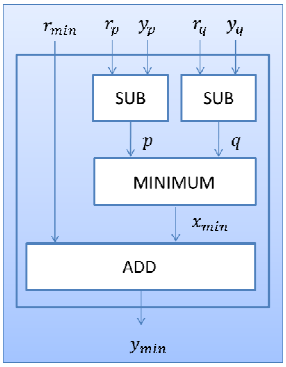

Figure 4. The garbled circuit for minimum search.

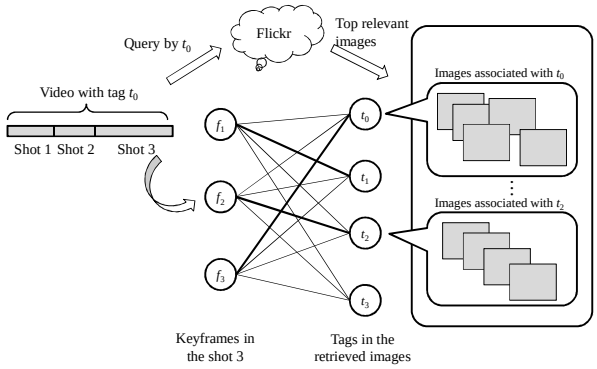

Figure 5. Video tag suggestion and localization based on bipartite graph matching.

At the client side, shot boundaries in the test video are first detected, and keyframes for each video shot are extracted by the global K means algorithm [3]. Keyframes of the video, denoted by $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_M\}$, are then represented by visual word histograms, which are built on top of the TOP-SURF toolkit [21]. At the server side, the server retrieves the top N images related to each commonly used tag from Flickr by tag search. Each retrieved image may be associated with multiple tags, and these tags provide extensive knowledge to facilitate tag expansion and localization. Images associated with the same tag are then clustered together. That is, if there are T different tags in the retrieved images, T image clusters would be formed. Let $\mathcal{B}_i = \{\boldsymbol{b}_{i,1}, \boldsymbol{b}_{i,2}, ..., \boldsymbol{b}_{i,L}\}$ denote the retrieved images associated with the tag t_i. The tag t_i is represented by the average visual word histogram of $\mathcal{B}_i$. With this design, tags are represented as the same way as keyframes. Note that these processes are separately conducted in the client side and the server side, in the plain text domain.

The client encrypts the visual word histograms $\{\boldsymbol{a}_i\}$ and sends to the server, and in the server a bipartite graph is constructed in the encrypted domain. Two disjoint sets of nodes in this graph respectively denote keyframes and tags, and each edge is given a weight calculated based on the *encrypted squared Euclidean distances* $[D(\boldsymbol{a}_i, t_j)]$ defined in eqn. (1) between keyframes and tags. That is,

$$[D(\boldsymbol{a}_i, t_j)] = \left[\sum_{k=1}^{K} (h_i^a(k) - h_j^t(k))^2\right] = [H_1] \cdot [H_2] \cdot [H_3], \quad (14)$$

where $h_i^a(k)$ and $h_j^t(k)$ respectively denote the kth histogram values of the keyframe $\boldsymbol{a}_i$ and the tag t_j, $[H_1]$ is the encrypted value of $\sum (h_j^t(k))^2$, $[H_2]$ is the encrypted value of $\sum(-2h_i^a(k)h_j^t(k))$, and $[H_3]$ is the encrypted value of $\sum (h_i^a(k))^2$ calculated based on the communication protocol mentioned previously. After graph construction, the minimum cost bipartite matching is then found by the Hungarian algorithm implemented in the encrypted domain. After decrypting matching results, the keyframe-tag correspondence is found. Tags associated with keyframes in the same shot are collected to annotate each video shot. Taking Figure 5 as an example, the third video shot has three keyframes, which are best matched (shown in bold edges) with t_0, t_1, and t_2, respectively. Therefore, the original tag t_0 is localized into the third video shot, and new tags t_1 and t_2 are suggested to tag this shot as well.

5.2 Privacy-Preserving Video Copy Detection

The Video Linkage system [16] is respectively implemented in the plain text and encrypted domains, and performance obtained from two domains will be compared. First, keyframes are extracted from video shots and are represented by HSV color histograms, YCbCr color layouts, and motion vector histograms. The distance between two keyframes f_i and f_j is calculated by linearly combining the squared Euclidean distances $d_{ij}^{(k)}$, $k = 1, 2, 3$, derived from three features, as illustrated in Figure 6. That is,

$$D(f_i, f_j) = \sum_{k=1}^{3} w_k d_{ij}^{(k)}. \quad (15)$$

Three weights are $w_1 = w_2 = w_3 = 1/3$.

A video can be viewed as a collection of keyframes. Taking keyframes of two videos as two disjoint sets of nodes, denoted by $g_1 = \{f_{11}, f_{12}, ..., f_{1m_1}\}$ and $g_2 = \{f_{21}, f_{22}, ..., f_{2m_2}\}$, a bipartite graph is constructed by evaluating the distance $D(f_i, f_j)$ between any two frames f_i and f_j. Based on this graph, the minimum cost bipartite graph matching M is found by the Hungarian algorithm. The matching result is sent from the server to the client, and the client determines whether two videos are copies by calculating the similarity measure:

$$VL(g_1, g_2) = \frac{\sum_{f_{1i}, f_{2j} \in M} (1 - D(f_i, f_j))}{m_1 + m_2 - |M|}, \quad (16)$$

where $|M|$ denotes the number of edges in the match M. If $VL(g_1, g_2)$ is larger than an empirical threshold, these two videos are said to be video copies. The equation (15) is written in the plain text domain, but in the privacy-preserving framework it is implemented by encryption-based operations. That is,

$$[D(f_i, f_j)] = \left[\sum_{k=1}^{3} w_k d_{ij}^{(k)}\right] = \prod_{k=1}^{3} [d_{ij}^{(k)}]^{w_k}, \quad (17)$$

where $[d_{ij}^{(k)}]$ is calculated by eqn. (1) and three weights are $w_1 = w_2 = w_3 = 1/3$.

Note that in [16] other measurement variants based on graph matching are proposed. Performance based on these measurements is similar, but some variants are more efficient to be calculated. In this work we only implement one of the basic measurements for copy

detection to demonstrate that the proposed encryption framework is adaptable for video copy detection.

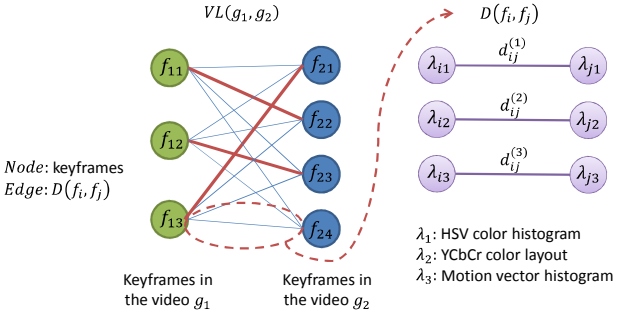

Figure 6. Video copy detection using bipartite graph matching.

6. EVALUATION

Two applications were implemented in Java and executed in a PC with Intel Core 2 Quad 2.40GHz, 4GB RAM. Client and server were implemented as different programs passing messages to each other. The secure parameter for both Paillier and DGK cryptosystems is set as 1024 bits, and the secure parameter κ for uniform random numbers is set as 80 bits. Settings of the packing method described in Sec. 3.2 are $\ell = 50$, $K' = 10$, and $\sigma = 80$. The input length of the comparison protocol described in Sec. 4.1 is set as 30 bits.

6.1 Privacy-Preserving Tag Suggestion

The video tag suggestion application is evaluated based on the Youtube videos provided in [20], in which top three rated videos from 15 categories in Youtube were downloaded, consisting of 1368 shots and 3176 keyframes. There are averagely 11.67 tags for each video after filtering out stop words. Through Flickr API, images with associated tags relevant to a query were retrieved.

We use the visual dictionary consisting of 10,000 visual words, which is provided by the TOP-SURF toolkit [21]. With the bag of visual words representation, similarity between keyframes and tags can be measured to construct a bipartite graph.

We respectively implement this process in the plain text domain and in the encrypted domain, and show the average tag suggestion accuracy for each category of Youtube in Table 1. The tag suggestion accuracy of a video is calculated as the ratio of the number of correct tags (manually judged) to the number of all suggested tags. From Table 1, the overall performance obtained in the encrypted domain is comparable with that obtained in the plain text domain, although tagging accuracy is content dependent as we expected. Performance difference between two domains comes that in the encrypted domain all values are quantized into integers and the rounding errors yield slight difference in similarity calculation as well as the results of graph construction.

Table 2 shows the average time for calculating the squared norm of the query vector, i.e., $[T_3]$ defined in eqn. (3). As we expect, the basic operation involves frequent communication between Alice and Bob, and thus requires more execution time. By the packing method reducing the number of communication, and the pre-computation scheme further eliminating two-way communication, execution time can be significantly reduced.

Table 3 shows execution time of minimum search for different number of private values. If the number of private values is larger, the ratio of time needed by homomorphic operations to that needed by the garbled circuit would be larger. The results reveal that the garbled circuit is efficient in graph matching if the server has a large number of collected tags. Note that we emphasize relative execution time rather than absolute execution time in Table 2 and Table 3, because we only implemented this system on a single PC rather than a real cloud environment having extremely powerful computation.

Figure 7 shows sample results of tag suggestion. The suggested tags for each video shot are displayed as subtitles. In the first three examples, the suggested tags appropriately describe the corresponding visual content; in the last example, the "image" tag is too general and the "century" tag is inappropriate to describe the baby laughing shot.

Table 1. Tag suggestion accuracy in the plain text domain and in the encrypted domain.

Categories	Plain text domain	Encrypted domain
Autos & Vehicles	0.63	0.62
Comedy	0.48	0.47
Education	0.36	0.39
Entertainment	0.63	0.62
Film & Animation	0.35	0.35
Gaming	0.31	0.27
Howto & Style	0.55	0.52
Music	0.66	0.65
News_Politics	0.78	0.77
Nonprofits & Activism	0.63	0.61
People & Blogs	0.46	0.47
Pets & Animals	0.61	0.61
Science & Technology	0.76	0.78
Sports	0.71	0.67
Travel & Events	0.65	0.65
Average	0.57	0.56

Table 2. Average time to calculate $[T_3]$ defined in eqn. (3).

	Basic	Packing	Pre-computation
Average Time (sec.)	16.83	2.19	0.08

Table 3. The execution time (sec.) of minimum search for different number of elements (N).

N	Garbled circuit	Homomorphic operations	Time ratio
100	36.5	101.2	2.77
200	61.9	203.3	3.29
500	137.8	509.1	3.69
1000	265.5	1020.6	3.84
2000	519.2	2051.7	3.95

6.2 Privacy-Preserving Video Copy Detection

Following the experimental setting of [16], the MUSCLE-VCD-2007 dataset [10] is used to evaluate the privacy-preserving video copy detection. The source dataset includes 101 videos with a total length of 80 hours, consisting of 49,751 shots and 204,510 keyframes. There are two sets of query videos, and we only use the first query set consisting of 15 query videos, with total 2,983 shots and 10,702 keyframes. A query video may be generated from a clip from a source video with various transformations, such as color adjustment and blur.

With the combination of HSV color histogram, YCbCr color layout, and motion vector histogram, squared Euclidean distances between query video keyframes and source video keyframes can be measured to construct a bipartite graph, and the best matching between them is determined by the Hungarian algorithm. We respectively implement this process in the plain text domain and in the encrypted domain, and found that, in both domains, correct video copies can be detected for 13 of 15 queries. This again verifies the encrypted-domain process achieves the same performance as in the plain text domain.

Figure 7. Sample results of video tag suggestion and localization. The first three samples shows correct suggestion results, while the fourth sample shows incorrect suggestion results (tags shown in red).

7. CONCLUSION

We have proposed adopting homomorphic cryptosystems and secure comparison protocols to build a privacy-preserving bipartite matching framework that can be utilized in a wide range of multimedia analysis researches. With the proposed framework, the user can take advantage of server's computation power to accomplish multimedia analysis, while the server keeps unaware of the user' data so that user's privacy is protected.

Two stages are involved in the bipartite graph matching framework: graph construction and graph matching. At the graph construction stage, the Paillier cryptosystem is used to implement homomorphic encryption, and a communication protocol is utilized to facilitate calculating edge weights. At the matching stage, the encrypted-domain Hungarian algorithm is developed based on the DGK homomorphic encryption scheme and a private number comparison protocol. The garbled circuit is further adopted to reduce computation cost. Two applications of video tag suggestion and video copy detection are built to verify correctness of the proposed framework. The privacy preserving framework achieves performance comparable to the plaintext version, even though the bipartite graph is constructed and analyzed in the encrypted domain.

Acknowledgements

The work was partially supported by the Ministry of Science and Technology in Taiwan under the grant MOST103-2221-E-194-027-MY3.

8. REFERENCES

[1] R. Diestel. Graph Theory. Heidelberg, Springer, 2005.

[2] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. Proceedings of the International Conference on Theory and Application of Cryptographic Techniques, pp. 223-238, 1999.

[3] A. Likas, N. Vlassis, N., and J.J. Verbeek, The global k-means clustering algorithm. Pattern Recognition, vol. 36, pp. 451-461, 2003.

[4] Z. Erkin, M. Franz, and J. Guajardo. Privacy-preserving face recognition, Proceedings of International Symposium on Privacy Enhancing Technologies, pp. 235-253, 2009.

[5] I. Damgard, M. Geisler, M. Kroigaard. Efficient and secure comparison for online auctions. Proceedings of the Australasian Conference on Information Security and Privacy, pp. 416-430, 2007.

[6] I. Damgard, M. Geisler, M. Kroigaard. A correction to efficient and secure comparison for online auctions. International Journal of Applied Cryptography, vol. 1, no. 4, 2009.

[7] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. Proceedings of International Conference on Information Security and Cryptology, pp. 229-244, 2009.

[8] R. Agrawal and R. Srikant. Privacy-preserving data mining. Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 439-450, 2000.

[9] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A. Varna, S. He, M. Wu, and D. Oard. Confidentiality-preserving rank-ordered search. Proceedings of ACM Workshop on Storage Security and Survivability, pp. 7-12, 2007.

[10] J. Law-To, A. Joly, and N. Boujemaa. Muscle-VCD-2007: a live benchmark for video copy detection, 2007. http://www-rocq.inria.fr/imedia/civr-bench/.

[11] W. Lu, A. Swaminathan, A.L. Varna, and M. Wu. Enabling search over encrypted multimedia databases. Proceedings of SPIE Conference on Media Forensics and Security, 2009.

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

[13] Z.K.G. do Patrocinio, S.J.F. Guimaraes, and H.B. de Paula. Bipartite graph matching for video clip localization. Proceedings of Brazilian Symposium on Computer Graphics and Image Processing, pp. 129-138, 2007.

[14] S.J.F. Guimaraes, Z.K.G. do Patrocinio, K.J.F. Souza, and H.B. de Paula. Gradual transition detection based on bipartite graph matching. Proceedings of IEEE International Workshop on Multimedia Signal Processing, 2009.

[15] H. Xu, L. Liu, L.-F. Sun, and S.-Q. Yang. Fast and robust detection of near-duplicates in web video database. Proceedings of IEEE International Conference on Multimedia & Expo, pp. 293-296, 2008.

[16] H.-S. Kim, J. Lee, H. Liu, and D. Lee. Video linkage: group based copied video detection. Proceedings of International Conference on Content-Based Image and Video Retrieval, pp. 397-406, 2008.

[17] L. Bai, Y. Hu, S. Lao, A.F. Smeaton, and N.E. O'Connor. Automatic summarization of rushes video using bipartite graphs. Multimedia Tools and Applications, vol. 49, no. 1, pp. 63-80, 2010.

[18] Z. Zhang, Z.-N. Li, and M.S. Drew. Learning image similarities via probabilistic feature matching. Proceedings of IEEE International Conference on Image Processing, pp. 1857-1860, 2010.

[19] Y. Gao, Q. Dai, M. Wang, and N. Zhang. 3D model retrieval using weighted bipartite graph matching. Signal Processing: Image Communication, vol. 26, pp. 39-47, 2011.

[20] W.-T. Chu, C.-J. Li, and Y.-K. Chou, Tag suggestion and localization for web videos by bipartite graph matching. Proceeding of International ACM Workshop on Social Media, pp. 35-40, 2011.

[21] B. Thomee, E.M. Bakker, and M.S. Lew. TOP-SURF: a visual words toolkit. Proceedings of ACM Multimedia Conference, pp. 1473-1476, 2010.

[22] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. Proceedings of International Conference on Cryptology and Network Security. vol. 5888, pp. 1-20, 2009.

[23] R. Lagendijk, Z. Erkin, and M. Barni, Encrypted signal processing for privacy protection, IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 82-105, 2013.

[24] J. Shashank, P. Kowshik, K. Srinathan, and C. Jawahar, Private content based image retrieval, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2008.