

# Enabling Privacy-preserving Image-centric Social Discovery

Xingliang Yuan<sup>†</sup>, Xinyu Wang<sup>†</sup>, Cong Wang<sup>†</sup>, Anna Squicciarini<sup>‡</sup>, and Kui Ren<sup>§</sup>

<sup>†</sup>City University of Hong Kong, Hong Kong

<sup>‡</sup>Pennsylvania State University, United States

<sup>§</sup>State University of New York at Buffalo, United States

<sup>†</sup>xingliang.yuan@student.cityu.edu.hk, {xinyuwang, congwang}@cityu.edu.hk; <sup>‡</sup>asquicciarini@ist.psu.edu; <sup>§</sup>kuiren@buffalo.edu

**Abstract**—The increasing popularity of images at social media sites is posing new opportunities for social discovery applications, i.e., suggesting new friends and discovering new social groups with similar interests via exploring images. To effectively handle the explosive growth of images involved in social discovery, one common trend for many emerging social media sites is to leverage the commercial public cloud as their robust backend datacenter. While extremely convenient, directly exposing content-rich images and the related social discovery results to the public cloud also raises new acute privacy concerns. In light of the observation, in this paper we propose a privacy-preserving social discovery service architecture based on encrypted images. As the core of such social discovery is to compare and quantify similar images, we first adopt the effective Bag-of-Words model to extract the “visual similarity content” of users’ images into image profile vectors, and then model the problem as similarity retrieval of encrypted high-dimensional image profiles. To support fast and scalable similarity search over hundreds of thousands of encrypted images, we propose a secure and efficient indexing structure. The resulting design enables social media sites to obtain secure, practical, and accurate social discovery from the public cloud, without disclosing the encrypted image content. We formally prove the security and discuss further extensions on user image update and the compatibility with existing image sharing social functionalities. Extensive experiments on a large Flickr image dataset demonstrate the practical performance of the proposed design. Our qualitative social discovery results show consistency with human perception.

## I. INTRODUCTION

Images represent a popular form of self-expression in online social media sites [1]. Users may upload images to share fragments of their personal life, show representations of their interest and events while connecting with known or even unknown peers. In particular, photos increasingly provide opportunities for social discovery applications. Due to the semantic richness of image content, many social applications are now directly exploring images to suggest new friends and discover new social groups with similar interests. For example, people on Flickr or Pinterest can upload their images to find their recommended social groups, within which a group of people shall share the same interests or similar experiences [2]. The idea is therefore to enable *social discovery*, using images as the connection medium.

At the core, image based social discovery consists of comparing and quantifying similar images at large scale, e.g.,

over 3.5 million images uploaded daily on Flickr [1], which inevitably demands huge storage and computation resources. To effectively handle this large-scale computational and storage problem, one common technology trend for many emerging social media sites is to leverage the commercial public cloud as their backend datacenter, for the robust and elastic power with reduced cost. Among others, the popular photo sharing service, Instagram [3], is such an example; users’ uploaded images are stored directly in the Amazon cloud, rather than passing through their on-premise local servers.

While extremely convenient, directly exposing content-rich images and the related social discovery results to the public cloud also raises new acute privacy concerns. Firstly, semantically rich images may easily reveal content-sensitive information, if not protected well. For example, a student’s photo of 2013 graduation might be used to discover the whole 2013 graduates, and even expose the student’s family members and other friends. Secondly, image-based social discovery in outsourced scenario may cause even higher privacy risks, as it further exposes personal profiles, revealing information such as education background, living places, etc., to the public cloud, which is known to have very broad attack surface. In fact, some cloud-related new security threats are still not yet fully understood [4]. As many images are now directly stored to the public cloud in unencrypted form, how to satisfactorily address these acute privacy concerns can be a pivotal issue.

In light of above observations, in this paper, we propose our study on a new privacy-preserving image-centric social discovery system that can be deployed at untrusted public cloud. Different from most existing works on privacy in social discovery that focus solely on image context, such as manually annotated image tags, captions, attributes, etc., [5], [6], our proposed secure social discovery design builds on top of images’ visual features [7]. The rationale is that, consistent with the well-known homophily theory [8], similar users are likely to connect with one another. Hence, in the context of images, we expect two users associated with images with similar content to have higher chances sharing similar interest, and therefore wanting to connect with one another. To enable content-based social discovery, following a paradigm commonly used in image processing and retrieval [9], we first adopt the effective Bag-of-Words model to extract the

“visual similarity content” of users’ images into individual image profiles in the form of high dimensional vectors. Then, we model the problem of privately comparing and quantifying image content for social discovery as secure similarity retrieval over encrypted high-dimensional image profiles.

Similar to the existing social media sites, our proposed system also treats public cloud as the robust backend data center. The only difference is that the public cloud hosts all the images in encrypted form, but should still be able to facilitate fast and scalable similarity discovery over millions of encrypted images. For that purpose, we propose a secure and very efficient indexing structure from building blocks of off-the-shelf locality-sensitive hashing [10], cuckoo hashing [11], as well as recent advancements of searchable symmetric encryption [12]–[14], etc. We will show how these techniques can be utilized in a non-trivial way to achieve practical and accurate social discovery in a provably secure fashion. The resulting design enables social media sites to obtain encrypted social discovery results from the backend cloud, while protecting both the image content as well as the related “visual similarity” image profiles. Our system is expected to naturally complement existing context-based approaches and serve for more advanced privacy-aware social applications. The contributions are summarized as follows:

- 1) Our system is the first to deliver the privacy-preserving image-centric social discovery services in an outsourced architecture, which allows fast and scalable discovery over millions of encrypted users’ images.
- 2) Our proposed secure indexing structure guarantees the privacy protection on image content and user image profiles, while maintaining reasonably good social discovery accuracy, indexing space efficiency, and constant-sized bandwidth cost.
- 3) We formally prove the security of our system design and discuss their practical extensions, including supporting user profile update. Compatibility with sharing services of existing social media sites is also discussed.
- 4) Experiments over 1 million data records created from Flickr image dataset (MIRFlickr-1M [15]) demonstrate the practical performance of the proposed design on discovery latency, space saving, and high accuracy. The social discovery results are also shown to be consistent with human perception.

The paper organization: Section II states the system architecture, threat model, and preliminaries. Section III elaborates our detailed system design of privacy-preserving image-centric social discovery. Section IV presents the formal security analysis of our system design. Section V reports the performance evaluations, followed by Section VI that overviews the related works. Finally, Section VII concludes the whole paper.

## II. PROBLEM STATEMENT

### A. System Architecture and Threat Model

The intuition for image-centric social discovery is to use image content to interpret the user interests. If two users post images on similar content or topics, such as the same places visited, similar types of food, pets, or plants they like,

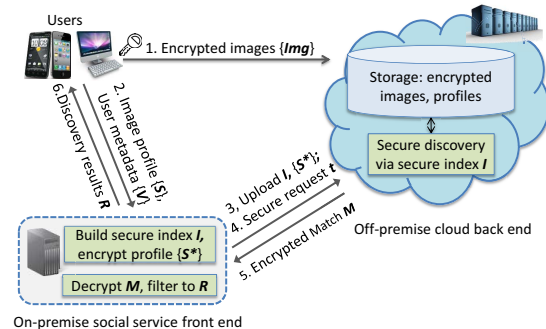


Fig. 1: Our proposed system architecture

then they are likely to have similar interest, and therefore they may connect to possible friends, peers or even love partners [7], [16]. Our approach is to recommend friends by *quantifying* similarity of visual content between users. Clearly, the higher the similarity, the higher the chances that the two users would share similar interests. If two users’ preferred images contain similar “visual words”, then their corresponding image profiles will be closer to each other. For privacy preservation, the problem is thus how to find users with nearest image profiles with regard to any targeted user(s), among a large number of encrypted image profiles. Hereafter, we formulate it as secure similarity search over encrypted users’ high-dimensional image profiles.

**System service flow:** The high-level system architecture is illustrated in Fig. 1. It involves three cooperating entities: the users (*U<sub>sr</sub>*), the trusted social media site as the on-premise service front-end (*SF*), and the untrusted cloud server (*CS*) as the off-premise backend. Our service flow involves three phases: 1) **User data upload** shown in **Steps 1 & 2**: For privacy protection, each *U<sub>sr</sub>* first encrypts all her images  $\{Img\}$ , then uploads them directly to *CS*. Meantime, each *U<sub>sr</sub>* undertakes the preferred images to quantify her own image-centric interests via a small-size image profile  $S$  and related metadata  $V$ , both of which will be sent to *SF*. 2) **Service frontend initialization** shown in **Step 3**: Upon receiving all users’  $\{S, V\}$ , *SF* will fast group the similar image profiles via an advanced secure index  $I$  to enable the efficient and secure social discovery later at *CS*. 3) **Privacy-preserving social discovery** shown in **Step 4, 5 & 6**: For any targeted user(s), *CS* is able to continuously process the secure social discovery request, in the form of secure trapdoor  $t$ , from *SF*, and returns encrypted matching result  $M$ , without seeing the encrypted images  $\{Img\}$  or the users’ image profiles  $\{S\}$ . The encrypted match  $M$  will be decrypted and filtered by *SF* to derive actual social discovery result  $R$  for targeted users.

For practical consideration, the proposed architecture should support any subsequent user profile update on the index  $I$ , without affecting correctness, efficiency, and security. It should also support standard social image sharing functionalities, where users can use different encryption techniques [17], [18] to precisely enforce fine-grained access to their encrypted images even in outsourced scenarios.

**Threat assumptions:** We consider the security threats primarily from the “honest-but-curious” cloud, who faithfully follows the designated operations yet intends to infer users’ encrypted image contents, image profiles, and related information. We focus on protecting confidentiality of those images and image profiles outsourced in a public cloud. We do not consider other attacks, such as those on data integrity, availability, etc., in this paper, which can be handled by various orthogonal mechanisms. For our design, we assume that the service front end of the media site and all users are trustworthy. We assume they can protect their secrets, such as keys and parameters, and all communications are secure and authenticated.

### B. Definitions of System Functions

Our system will instantiate seven functions, where GenProf and ComputeLSH run at each *U<sub>sr</sub>*, Gen, ConSecIdx, GenT<sub>pd</sub>r and GetRec run at *SF*, and SecRec runs at *CS*:

- $S \leftarrow \text{GenProf}(\{Img\}, \Delta)$ : given images  $\{Img\}$  and function parameter  $\Delta$ , output the user image profile  $S$ .
- $V \leftarrow \text{ComputeLSH}(S, \mathbf{h})$ : given profile  $S$  and LSH function parameter  $\mathbf{h}$ , output the user metadata  $V$ .
- $K \leftarrow \text{Gen}(1^\lambda)$ : given input security parameter  $\lambda$ , output the secret key  $K$ .
- $(I, \mathbf{S}^*) \leftarrow \text{ConSecIdx}(K, \mathbf{S}, \mathbf{V})$ : given the secret key  $K$ , image profile set  $\mathbf{S}$  and metadata set  $\mathbf{V}$ , output a secure index  $I$  and ciphertexts  $\mathbf{S}^*$ .
- $\mathbf{t} \leftarrow \text{GenTpd}r(K, V)$ : given the secret key  $K$  and metadata  $V$ , output a secure discovery trapdoor  $\mathbf{t}$ .
- $M \leftarrow \text{SecRec}(\mathbf{t}, I)$ : given secure trapdoor  $\mathbf{t}$  and index  $I$ , output secure match  $M$ , a set of encrypted image profiles.
- $R \leftarrow \text{GetRec}(K, M)$ : given the secret key  $K$  and secure match  $M$ , output the actual recommendation result  $R$ .

We will also use some basic cryptographic primitives:

- $f(k, \cdot), g(k, \cdot), G(\cdot)$ : pseudo-random functions (PRF).
- $\text{Enc}(k, \cdot), \text{Dec}(k, \cdot)$ : symmetric key based semantic secure encryption/decryption function.

### C. Preliminaries

Our design involves locality-sensitive hashing and cuckoo hashing as building blocks. We provide a brief review of these notations next.

**Locality-sensitive hashing:** LSH is an algorithm to solve the approximate nearest neighbor search in high-dimensional spaces [10]. The idea is to hash the high-dimensional input data points via specially-designed LSH functions, where similar data points are hashed into the same bucket with higher probability than distant ones. Let  $O$  be the domain of data points and  $D$  be the distance measure between points.

**Definition 1** (Locality-sensitive Hashing). *Given distance  $r_1, r_2, r_1 < r_2$ , and probability value  $p_1, p_2, p_1 > p_2$ , a function family  $\mathcal{H} = \{h : O \rightarrow U\}$  is  $(r_1, r_2, p_1, p_2)$ -locality-sensitive if for any  $o_i, o_j \in O$ : if  $D(o_i, o_j) \leq r_1$  then  $P[h(o_i) = h(o_j)] \geq p_1$ ; if  $D(o_i, o_j) > r_2$  then  $P[h(o_i) = h(o_j)] \leq p_2$ .*

For search with good accuracy, multiple LSH hash tables are needed [10], [19]. One needs to hash the query point  $q$

into buckets in hash tables to find all candidate points, and rank them based on their distance to  $q$ .

**Cuckoo hashing:** cuckoo hashing [11] is designed to build high performance hash tables. It can disperse objects with hash collisions into different choice of hash tables for better load balance. The standard cuckoo hashing is defined below:

**Definition 2** (Cuckoo Hashing). *Given two hash tables  $T_1$  and  $T_2$ , both of them contain  $w$  space units. Two independent and random hash functions  $h_1, h_2 : O \rightarrow \{0, w-1\}$  are associated to  $T_1$  and  $T_2$ , where  $O$  is the object domain. One object  $o \in O$  can be inserted either in the bucket  $h_1(o)$  of  $T_1$  or the bucket  $h_2(o)$  of  $T_2$ .*

If the hash collision happens, the object in the occupied bucket will be kicked away to another hash table. When more objects are inserted, the number of kick-away operations will increase. In practice, the standard cuckoo hashing can be extended with more than 2 hash tables.

## III. OUR PROPOSED DESIGN

### A. User Data Upload

The core of image-centric social discovery is to quantify visual similarity from users’ images as mentioned before. Since images are naturally stored at users’ local devices with increasing computing power, our system adopts the “crowd-sourcing” strategy to decentralize the tedious and potentially burdensome image preprocessing work to individual users. Each *U<sub>sr</sub>* can preprocess her own preferred images offline to assist the Service frontend initialization at *SF* later on. Specifically, two tasks are assigned to each *U<sub>sr</sub>*’s client:

**User Image Profile Generation:** We represent each user’s image-centric interests as an image profile based on the Bag-of-Words model (BoW) [9], which is shown effective in contexts of image similarity comparison and retrieval [7], [16]. In function GenProf, the parameter  $\Delta$  denotes the visual word vocabulary  $\Delta$ , which can be pre-trained and shared by *SF*. Then for each preferred *Img*, a BoW vector is obtained by extracting image features and clustering them with regard to  $\Delta$ , where each entry in BoW vector denotes the occurrences of the corresponding visual word. We then aggregate all BoW vectors for a certain user and normalize it to create the user image profile vector  $S = \{s_1, \dots, s_m\}$ , where  $s_i$  reflects the strength of user’s preference on the  $i$ -th visual word in  $\Delta$ , and  $m$  is the size of  $\Delta$ . We note that such user image profile generation is not unique. In order to discover users with similar image-centric interests, we need a metric to measure the closeness of the high-dimensional user image profile vectors. As suggested in prior work [10], [20], various metrics, e.g., Euclidean, cosine, Jarcad distances, etc., work well for the purposes. Without loss of generality, we adopt Euclidean distance in the system design and implementation of this paper. We leave the effectiveness comparison against other metrics in our future work.

**User Metadata Generation:** For high quality of similarity evaluation, “visual words” vocabulary  $\Delta$  may contain more than hundreds of words, which makes the user image profile

$S$  high-dimensional. To assist  $SF$  to build the secure similarity index for social discovery shown in Section III-B, our system further demands each  $U_{sr}$  to generate some metadata  $V$  for her image profile  $S$ , in the form of locality-sensitive hash values. In function ComputeLSH, the parameter  $\mathbf{h} = \{h_1, h_2, \dots, h_l\}$  denotes the set parameters of  $l$  LSH functions, introduced in Section II-C. With image profile  $S$ , each  $U_{sr}$  can compute her own metadata  $V = \{h_1(S), \dots, h_l(S)\}$ . Similar to visual word vocabulary  $\Delta$ , parameter  $\mathbf{h}$  can be pre-shared by  $SF$ .

Each  $U_{sr}$  will also encrypt her images and upload them to  $CS$  for privacy protection. We will show these encrypted images are compatible with social data sharing services by exploring advanced encryption techniques in Section III-E.

### B. Service Frontend Initialization

To facilitate cloud to remotely perform privacy-preserving image-centric social discovery, the major task for  $SF$  is to build a secure similarity search index over encrypted high-dimensional image profiles. The question is how to make the design practical and be able to support millions of user profiles with high accuracy and strong security guarantees?

One straightforward approach is to first cluster similar user image profiles into the same bucket in LSH based hash tables. Then, a secure index can be built after encrypting all buckets and inserting random paddings [21]. However, the unbalanced load of LSH [19] and secure index requirement [12] make this approach suffer from unscalable index size  $O(n^2)$  for  $n$  data items and large bandwidth cost  $O(n)$  for each search request interaction. It is far from practical when meeting millions of user profiles at social media sites.

**Combining LSH and Cuckoo Hashing:** Thus, we have to explore more advanced secure index designs that aim to take into account the practical performance parameters such as index size, index load factor<sup>1</sup>, and security overhead, from the very beginning of the design. For this purpose, we resort to a recent index design [22] in plaintext domain which utilizes the combination of LSH and cuckoo hashing for fast and efficient similarity search over big data. As introduced, cuckoo hashing is known for building practical, high-performance hash tables. It uses multiple hash tables and allows multiple hash locations for each data item. Thus, data items can be “moved” among those positions during data insertions in index building. Such a design provides a lot of flexibility on LSH load balance and helps increase hash table load factors [11]. The resulting index in [22] is shown to support high load-balance, low search latency, and reasonable search accuracy. However, it supports only plaintext data without security.

**Towards Secure and Efficient Similarity Index:** How to leverage the aforementioned observation and design a provably secure index with all the salient features remains a challenging problem. Recall that in [22], the basic idea is to insert each data item into one of the  $l$  initialized hash tables associated with  $l$  LSH functions. If any collision happens, we will use cuckoo-driven approach to kick away the existing data item and insert

<sup>1</sup>The ratio of the number of data items to the number of entries in the index

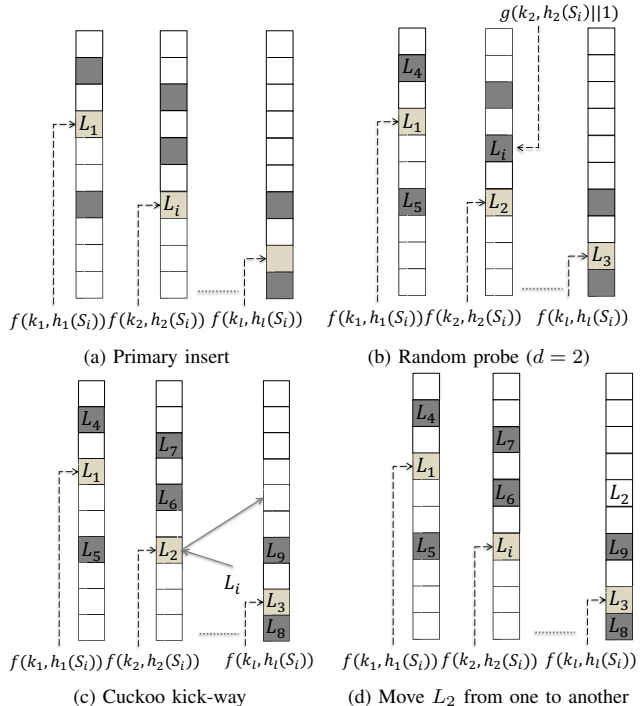


Fig. 2: Our proposed index construction Insertion phase

the very data item into another hash table from the remaining  $l - 1$  choices. To further resolve the hash collision, probing technique is also utilized [19]. But for security guarantees, we need to do all the operations in a privacy-preserving manner. We achieve this by exploring a suite of carefully constructed security designs. Below we illustrate our design blueprint.

Firstly, our design uses one-way functions, like keyed PRF, to secure the sensitive LSH values while preserving the locality of close image profiles. Note that LSH does not preserve the one-way property as pseudo-random functions. Thus, it is possible to reveal the input image profiles by observing LSH values. Secondly, a specially designed random probing technique is explored in index building to protect the similar image profiles with hash collisions. Specifically, the way we search for empty buckets to resolve collisions is random. Finally, we encrypt data items stored in the hash table buckets via carefully designed random masks. Such masks can be extended for index update as shown later. Random paddings are utilized to make all buckets indistinguishable. We illustrate our design in Fig. 2 and Algorithms 1, 2 & 3.

Let image profile set be  $\mathbf{S} = \{S_1, \dots, S_n\}$  for  $n$  users. Let  $L_i$  be the identifier for  $S_i$  which enables  $CS$  to locate  $S_i$ 's physical location. Let the user metadata set be  $\mathbf{V} = \{V_1, \dots, V_n\}$ , where  $V_i = \{h_1(S_i), \dots, h_l(S_i)\}$  includes  $l$  LSH values of  $S_i$ . Let  $K = (k_1, \dots, k_l, k_s)$  be secret keys via  $\text{Gen}(1^\lambda)$  and  $f : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \{0, 1\}^u$ ,  $g : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \{0, 1\}^u$  be PRF. Function  $\text{ConSecIdx}(K, \mathbf{V}, \mathbf{S})$  constructs  $I$  through the following phases.

- **Setup phase:**

- 1) **Index initialization:** set the index load factor  $\tau \in (0, 1]$

---

**Algorithm 1: ConSecIdx( $K, S, V$ )**

---

**Data:**  $k = \{k_1, \dots, k_l, k_s\}$ : secret key set;  
 $S = \{S_1, \dots, S_n\}$ : user image profile set;  
 $V = \{V_1, \dots, V_n\}$ : user profile metadata set;  
**Result:**  $I$ : secure index.

```
begin
1   $w \leftarrow \lceil N/l \rceil$ ;
2   $I \leftarrow \text{new } T[l]$ ;
3   $\text{Loop} \leftarrow 0$ ;
4  for  $i \leftarrow 1$  to  $n$  do
5      Insert :
6      if  $\text{PrimaryInsert}(L_i, V_i) = \text{true}$  then
7          goto Next;
8      if  $\text{RandomProbe}(L_i, V_i, d) = \text{true}$  then
9          goto Next;
10     while  $\text{Loop} \leq \text{MaxLoop}$  do
11         // Cuckoo Kick-away.
12         Randomly pick  $j \in [1, l]$ ;
13          $\text{pos} \leftarrow f(k_j, V_i[j])$ ;
14          $L_k \leftarrow T_j[\text{pos}]$ ;
15          $T_j[\text{pos}] \leftarrow L_i$ ;
16          $\text{Loop} ++$ ;
17         goto Insert; // insert  $L_k$ .
18     rehash();
19     Next :
20     for  $i \leftarrow 1$  to  $n$  do
21          $B_i \leftarrow \text{BucketEnc}(L_i)$ ; // Encrypt buckets.
22          $S_i^* \leftarrow \text{Enc}(k_s, S_i)$ ; // Encrypt image profiles.
```

---

defined as  $\tau = n/N$ , where  $N$  is the size of index  $I$ . Then create  $l$  hash tables  $T$ . Each table  $T$  contains  $w = \lceil N/l \rceil$  buckets, where each bucket is  $u$ -bit length. Set random probe range as  $d$  within each table.

• **Insertion phase**, for  $1 \leq i \leq n$ :

2) *Primary insertion*: insert  $L_i$  to one of  $l$  hash tables. For  $j$  from 1 to  $l$ , if  $T_j[f(k_j, h_j(S_i))]$  is empty, then insert  $L_i$  to  $T_j[f(k_j, h_j(S_i))]$ .

3) *Random probe*: If the above  $l$  buckets are occupied, extra  $d$  random buckets for each table are probed. For  $j$  from 1 to  $l$  and  $\delta$  from 1 to  $d$ , if  $T_j[f(k_j, h_j(S_i) \parallel \delta)]$  is empty, then insert  $L_i$  to  $T_j[f(k_j, h_j(S_i) \parallel \delta)]$ .

4) *Cuckoo kick-away*: If no bucket is empty, randomly select  $j$  from 1 to  $l$  and kick way  $L_k$  located in  $T_j[f(k_j, h_j(S_i))]$ . Then insert  $L_i$  to  $T_j[f(k_j, h_j(S_i))]$  and re-insert  $L_k$  back via phases 1, 2&3 in an iterative fashion.

• **Encryption phase**:

5) *Bucket encryption*: for  $1 \leq i \leq n$ , generate random mask  $r_i = g(k_j, j \parallel \text{pos})$  for each  $L_i$ , which is located at  $T_j[\text{pos}]$ ,  $0 \leq \text{pos} < w$ . Then encrypt bucket  $B_i = r_i \oplus L_i$ .

6) *Random padding*: fill up the remaining  $w * l - n$  empty buckets with random values with equal length of  $B_i$ .

7) *Image profile encryption*: for  $1 \leq i \leq n$ , call  $\text{Enc}(k_s, S_i)$  to output  $S_i^*$ .

**Remark:** The space complexity of our proposed secure index is  $O(n)$ , because no duplicated copies of image profiles is stored after *cuckoo kick-away*. The security strength of our index design lies in the fact that all buckets of hash tables are filled in a secured fashion, determined by keyed pseudo-random functions, random paddings, and random masking. In

---

**Algorithm 2: PrimaryInsert( $L_i, V_i$ )**

---

**Data:**  $L_i$ : image profile identifier to be inserted;  
 $V_i$ : user metadata vector;  
**Result:**  $b$ : boolean *true* or *false*.

```
begin
1  for  $j \leftarrow 1$  to  $l$  do
2       $\text{pos} \leftarrow f(k_j, V_i[j])$ ;
3      if  $T_j[\text{pos}] = \text{NULL}$  then
4           $T_j[\text{pos}] \leftarrow L_i$ ;
5          return true;
6  return false;
```

---

---

**Algorithm 3: RandomProbe( $L_i, V_i, d$ )**

---

**Data:**  $L_i$ : image profile identifier to be inserted;  
 $V_i$ : user metadata vector;  
 $d$ : random probe range;  
**Result:**  $b$ : boolean *true* or *false*.

```
begin
// Randomly select a hash table.
1  for  $j \leftarrow 1$  to  $l$  do
2      for  $\delta \leftarrow 1$  to  $d$  do
3           $\text{pos} \leftarrow f(k_j, V_i[j] \parallel \delta)$ ;
4          if  $T_j[\text{pos}] = \text{NULL}$  then
5               $T_j[\text{pos}] \leftarrow L_i$ ;
6              return true;
7  return false;
```

---

practice, PRF is implemented by cryptographic hash functions, and their outputs should be mod the table capacity to determine the item buckets in hash tables. Non-committing private key encryption, i.e., XORing identifier  $L_i$  with a random mask  $r_i$  is applied to achieve the adaptive security. We will give formal security proofs of our design in Section IV. The correctness is guaranteed because PRF is deterministic. Thus, the locality-sensitivity of LSH values are preserved so that similar data items can be still grouped together in the secure index.

The kick-away operation may cause multi-round Insertion phases when most of buckets in index are occupied. Therefore, we assign a large number for the maximum allowed Insertion rounds defined as *MaxLoop* at line 10 in Algorithm 1. In the worst case, the rehash operation at line 17 will be performed to generate new LSH function set and rebuild the index. But in practice, a good choice of parameters (e.g., load factor  $\tau$ , random probe range  $d$ , LSH parameters) can greatly reduce the number of kick-ways which will be demonstrated in Section V.

### C. Privacy-preserving Image-centric Social Discovery Service

Our system delivers privacy-preserving image-centric social discovery service via secure similarity search over encrypted image profiles. The methodology is built on the concept of secure index construction to search and retrieve a constant yet small amount of encrypted near image profiles. *SF* only needs to perform a light-weighted distance ranking locally to obtain the final recommendation results. We state our service flow for a target user with image profile  $S$  and metadata  $V$  as follows:

1) *SF* calls  $\text{GenTpdr}(K, V)$  to one-way transform the user metadata  $V$  to a trapdoor  $t = \{t^1 = (t_0^1, \dots, t_d^1), \dots, t^l =$

$(t_0^l, \dots, t_d^l)$ . Here  $t_i^j = (pos_i^j, r_i^j)$ , where  $r_i^j = g(k_j, j || pos_i^j)$ , and  $pos_i^j$  is specified by  $f(k_j, V)$  for  $i = 0$  and  $f(k_j, V || i)$  for  $1 \leq i \leq d$ .

2) *CS* calls  $\text{SecRec}(t, I)$  to first locate  $l * (d + 1)$  encrypted buckets  $\{B_{i_c}\}$  specified by  $\{pos_{i_c}\}$ , and unmask each  $\{L_{i_c}\}$  via  $r_{i_c} \oplus B_{i_c}$ , for  $1 \leq c \leq l * (d + 1)$ . With  $\{L_{i_c}\}$ , *CS* then finds and returns the referenced  $l * (d + 1)$  encrypted image profiles  $\{S_{i_c}^*\}$ , as the secure match  $M$ , to *SF*.

3) *SF* calls  $\text{GetRec}(M)$  to recover  $\{S_{i_c}\}$  via  $\text{Dec}(k_s, S_{i_c}^*)$  for distance ranking and filtering, then output the recommended users  $R$  to the target user.

**Remark:** Our design requires *SF* to retrieve all matched results back for two reasons. Firstly, those image profiles will be decrypted and ranked according to their Euclidean distances with regard to the query image profile. The ranking results can be integrated with other friend recommendation algorithms for high quality social discovery. For example, one can use Friend-of-Friend [23] approach to further filter the ranking results. The derived final candidates can be recommended to the target user, who may want to only make friends with acquaintances in addition to common image-centric interests [7].

Secondly, the retrieved data are indeed small-sized (less than a few hundred for properly chosen  $l$  and  $d$ ), and the resulting distance ranking cost is marginal. We further note that this small bandwidth cost is constant when the data set grows, and will be sufficient for good social discovery accuracy. We will demonstrate these features in our experiment in Section V.

Note that our design can be combined with existing encryption techniques, such as [24], for image profile encryption, which is expected to further support encrypted cloud side distance ranking. We leave the comparison of these combinations and related evaluations as our future tasks.

#### D. User Profile Update

In social media sites, some users may occasionally upload new pictures to represent their real-time updated social interests. Accordingly, our system needs to further support the corresponding update procedure on the secure index, while guaranteeing the correctness and security. When user revises the preferred images, the image profile  $S$  and metadata  $V$  will be updated. Before inserting the new profile, the outdated image profile should be deleted. Thus, we must require *SF* to first retrieve the encrypted original image profile for deletion, and then inserting the new image profile back to the index  $I$ .

For secure and correct deletion and insertion operations, we need to slightly revise our design, which now introduces interactions between *SP* and *CS*. Our design is inspired from [13]. Specifically, we revise the way of bucket encryption as  $B = (G(r) \oplus (L || V), \text{Enc}(k_r, r))$ , where  $r$  is a random value,  $G(\cdot)$  is PRF,  $k_r$  is another secret key kept by *SF*.  $r$  should be newly picked each time whenever  $L$  needs to be re-masked at bucket  $B$ . The reason we include the masked user's metadata  $V$  in  $B$  is for secure insertion considerations. This is because if any kick-away operation happens due to insertion, *SF* should be able to use the related metadata to determine

where to re-insert the newly kicked bucket item in index  $I$ . Next, we introduce the secure deletion and secure insertion.

**Secure deletion:** To empty bucket  $B_i$ , *SF* first generates the delete trapdoor  $t$  via metadata  $V_i$ , which is similar as the search trapdoor but only contains the position  $pos$  as defined in Section III-C. After retrieving  $l * (d + 1)$  buckets  $\{B_{i_c}\}$ , the  $\{r_{i_c}\}$  are recovered via  $\text{Dec}(k_r, \cdot)$ , and identifiers  $\{L_{i_c}\}$  can be unmasked via  $\{G(r_{i_c})\}$ , for  $1 \leq c \leq l * (d + 1)$ . From recovered identifiers, *SF* will locate the target  $B_i$  which should be emptied. For security reason, this is done by *SF* replacing the target bucket  $B_i$  with masking of a special label  $\perp$ :  $B_i = (G(r_i) \oplus \perp, \text{Enc}(k_r, r_i))$  for newly picked  $r_i$ . Meanwhile, each  $B_{i_c}$  of the remaining  $l * (d + 1) - 1$  buckets needs to be re-masked too by using freshly and independently picked random  $r_{i_c}$ . Finally, *SF* sends  $l * (d + 1)$  re-masked buckets to *CS* to replace the old buckets at the same positions, which also hides the actual emptied bucket  $B_i$ . The identifier  $L_i$  is also passed to *CS* to remove the encrypted  $S_i^*$ . After deletion, *CS* sees all  $l * (d + 1)$  buckets are freshly re-masked.

**Secure insertion:** To insert  $S_i$ , *SF* generates the insertion trapdoor  $t$  for insertion of identifier  $L_i$  and metadata  $V_i$  to the index  $I$ . Like deletion trapdoor, the insertion trapdoor retrieves  $l * (d + 1)$  encrypted buckets first. If there are empty buckets found, i.e.,  $\perp$  is revealed after unmasking,  $L_i$  can be inserted directly. But if there are no empty buckets available, then one of the  $l$  primary insertion buckets, say  $B_k$ , will be kicked. *SF* first inserts  $V_i$  into  $B_k$  via  $B_k = (G(r_k) \oplus (L_i || V_i), \text{Enc}(k_r, r_k))$ . *SF* then re-masks all remaining  $l * (d + 1) - 1$  buckets as in **Secure Deletion**, and sends them back for *CS* to replace the old buckets at the same positions, which also hides the actual kicked bucket  $B_k$ . The encrypted  $S_i^*$  is also sent to *CS*. Note that  $B_k = (G(r) \oplus (L_k || V_k), \text{Enc}(k_r, r_k))$ . To re-insert  $L_k, V_k$  back to index  $I$ , *SF* first unmask  $B_k$ , uses  $V_k$  to generate the re-insertion trapdoor, and repeats the above **Secure Insertion** operations until no kick-away. Finally, *SF* passes the encrypted  $S_i^*$  passed to *CS* for storage.

**Remark:** Since the secure update essentially conforms to the same procedure as secure social discovery, the deletion and insertion do not leak extra sensitive information, except the fact that user profile is being updated and its possible position lies in all re-masked buckets. To further reduce the information leakage from update, one can leverage the batch update [12], [13] to perform multiple image profiles update simultaneously. We leave the detailed analysis as our future work.

#### E. System Compatibility

Our system can be compatible with image sharing services at social media sites. Encrypted data sharing in social networks has been proposed before, such as Persona [17], which adopts attribute-based encryption (ABE). Our design can utilize similar approaches for sharing encrypted images. Specifically, each user first generates ABE secret keys for her friends, based on the attributes derived from existing access control policy. For image encryption at *U<sub>sr</sub>*, a set of access attributes must be specified so that friends with the right ABE secret keys can

decrypt the image. Thus, users are able to manage friendship and mandate access over outsourced encrypted images.

#### IV. SECURITY ANALYSIS

In this section, we prove the security of the social discovery scheme between *SF* and *CS*. We follow the simulation based adaptive security definition, extended from searchable symmetric encryption (SSE) [12], [14]. The proof is to demonstrate that the attacker cannot infer any extra sensitive information from its views, i.e., a sequence of secure search queries and results. Specifically, we will first formally quantify the allowed information leakage from the view, including search pattern, access pattern, and intersection pattern. With the leakage, we show that a simulator can build a simulated index and output the simulated search trapdoors and results. To prove the security guarantee, we further show that for a sequence of adaptive queries, the views from the real search interactions and the simulated views cannot be distinguished. We define the information leakage as follows:

**Definition 3** (Access Pattern **AP**). *Let the metadata set  $\{V_1, \dots, V_q\}$  be a sequence of  $q$  queries,  $\{t_{V_1}, \dots, t_{V_q}\}$  be the trapdoor set. **AP** is defined as  $q$  sets of identifiers  $(\{L_{V_1}\}, \dots, \{L_{V_q}\})$  for  $\{t_{V_1}, \dots, t_{V_q}\}$ , where for each  $t_{V_i} = \{t^1, \dots, t^l\}$ ,  $t^j = \{t_0^j, \dots, t_d^j\}$  is the trapdoor for hash table  $T_j$  in  $I$  and associated to  $\{L_0^j, \dots, L_d^j\}$ , for  $1 \leq j \leq l$ .*

In our search scheme, we consider the similarity search pattern from the overlap of queries and the intersection pattern as the overlap in access pattern. Because the encrypted image profiles are semantically secure and search requests are protected by one-way trapdoors, such leakage only indicates the retrieved image profiles are close for our search correctness.

**Definition 4** (Similarity Search Pattern **SSP**). *For a sequence of  $q$  queries, **SSP** is defined as a symmetric matrix  $\Pi_{q \times q}^{SSP}$ , where the element  $\Pi^{SSP}[i][j]$  is a binary vector  $\nu$  that indicates the intersections  $V_i \cap V_j$  for  $1 \leq i, j \leq q$ , if  $V_i[m] = V_j[m]$ , then  $\nu[m] = 1$ , otherwise  $\nu[m] = 0$ , for  $1 \leq m \leq l$ .*

**Definition 5** (Intersection Pattern **IP**). *For a sequence of  $q$  queries, **IP** is defined as a symmetric matrix  $\Pi_{q \times q}^{IP}$ , where the element  $\Pi^{IP}[i][j]$  contains  $l$  arrays that records the access intersection for  $q_i$  and  $q_j$ . Array  $a_j$  stores the identifier intersection  $\{L_j\}$ , the trapdoor intersection  $\{t_j\}$  for hash table  $T_j$  in  $I$ .*

**Definition 6** (Trace  $\mathcal{T}$ ).  $\mathcal{T}$  is defined as  $(\mathbf{AP}, \mathbf{SSP}, \mathbf{IP}, I)$  for a sequence of  $q$  queries.

**Definition 7** (View  $\mathcal{V}$ ).  $\mathcal{V}$  is defined as  $(\{L_1\}, \dots, \{L_q\}, \{S_1^*\}, \dots, \{S_q^*\}, t_1, \dots, t_q)$  for a sequence of  $q$  queries.

Trace  $\mathcal{T}$  is the allowed information leakage, and view  $\mathcal{V}$  is observed by *CS* for a sequence of queries. The security of our similarity search scheme is stated in the Theorem 1.

**Theorem 1.** *Our proposed scheme satisfies the adaptive semantic security, if there exists a P.P.T. simulator  $\mathcal{S}$  and P.P.T. adversary  $\mathcal{A}$ , for a sufficiently large  $\gamma$  and every positive polynomial function  $p(\cdot)$ , we have  $P[\mathcal{A}(\mathcal{V}) = 1] - P[\mathcal{A}(\mathcal{V}_s) = 1] < \frac{1}{p(\gamma)}$ , i.e.,  $\mathcal{V}$  and  $\mathcal{V}_s$  are computationally indistinguishable.*

*Proof:* We will prove that the P.P.T.  $\mathcal{S}$  can first simulate an index  $I_s$  from the trace  $\mathcal{T}$ , then adaptively simulate the view  $\mathcal{V}_s$ , which is indistinguishable with the real view  $\mathcal{V}$ . The steps of simulating  $\mathcal{V}_s$  are presented as follows:

- To build  $I_s$ ,  $\mathcal{S}$  generates  $N$  random buckets, where  $N$  is the size of  $I$ . The bucket in  $I$  is  $B_i = r_i \oplus L_i$  or equal length random padding, so  $\mathcal{S}$  generates the random string  $B'_i$ , where  $|B'_i| = |B_i|$ . As a result,  $I_s$  and  $I$  are indistinguishable.
- To simulate the distinguishable  $\mathcal{V}_s$ ,  $\mathcal{S}$  first uses the identifiers  $\{L'\}$ , which are identical as the identifiers  $\{L\}$  from  $\mathcal{T}$ . Thus they are indistinguishable. Then  $\mathcal{S}$  generates set  $\{S^{*'}\}$  with the same size of  $\{S^*\}$ , where each  $S_i^{*'}$  is a random string and subject to  $|S_i^{*'}| = |S_i^*|$ , thus  $\{S^{*'}\}$  and  $\{S^*\}$  are computationally indistinguishable.
- To adaptively simulate the trapdoors,  $\mathcal{S}$  should create them consistently with **SSP** and **IP**, otherwise, the simulation will be detected by adversary. For the first query  $q_1$ ,  $\mathcal{S}$  simulates the trapdoors for  $l$  hash table one by one. For  $T_j$  in  $I$ , the trapdoor is  $t^{j'} = \{t_0^{j'}, \dots, t_d^{j'}\}$ , where  $t_i^{j'} = (pos', r')$ .  $\mathcal{S}$  will uniformly sample  $pos' \in [0, w)$  and locate  $B'_i$ . For an identifier  $L'_i$ , there must exist a random  $r'$  where  $L'_i = r' \oplus B'_i$ , so  $\mathcal{S}$  outputs  $r' = L'_i \oplus B'_i$ . For subsequent query  $q_{i+1}$  ( $1 \leq i \leq q-1$ ),  $\mathcal{S}$  adaptively simulates  $t'_{i+1}$  from **SSP** and **IP** for  $i$  queries. Similarly,  $\mathcal{S}$  simulates the trapdoors for  $l$  hash table one by one. If  $t_i^{j'}$  has not appeared before,  $\mathcal{S}$  adopts the same approach to simulate  $t_i^{j'}$ . Otherwise,  $\mathcal{S}$  copies the same  $t_i^{j'}$ .

The correctness of  $\mathcal{V}_s$  is easy to demonstrate by searching over  $I_s$ . Explicitly, the similarity search pattern matches the intersection pattern. For hash table  $T_m$  of  $q_i$  and  $q_j$ , if  $\nu[m] = 1$ , then the trapdoor  $t_i$  for  $q_i$  is the identical as  $t_j$  for  $q_j$ . For security, random  $pos'$  is of equal length to  $pos$ , where  $pos$  is produced via PRF  $f(k, \cdot)$ . Thus they are computationally indistinguishable. And  $r'$  is random due to the randomness of  $B'_i$ , so  $r'$  and  $r$  are computationally indistinguishable. Thus, the simulated  $t'$  and real  $t$  are computationally indistinguishable. We claim that there is no P.P.T. adversary who can distinguish between  $\mathcal{V}_s$  and  $\mathcal{V}$ . ■

**Remark:** Similar to existing work on SSE [12], [14], [25], our leakage on similarity search pattern and intersection pattern is from the fact that all search trapdoors are deterministic. It is not clear how to build probabilistic trapdoors for efficient symmetric key based search design. But to mitigate such statistical information leakage, one trick is to batch the social discovery requests for multiple randomly selected target users at once. However, more bandwidth cost will be incurred.

#### V. EXPERIMENTAL EVALUATION

##### A. Experimental Setup

We implement our system prototype by C++ in Linux. We deploy the *Usr* client on a PC with Intel i7-3930 CPU and 16G RAM, and deploy *SF* and *CS* on a server with Intel E5-2620 CPU and 120G RAM. Our system uses the computer vision library OpenCV 2.4.7 for visual word vocabulary training, image feature extraction, and image profile vector generation. Our experiment uses Speeded up Robust Features (SURF)

to extract features with a Determinant of Hessian interest point detector [26]. Here the interest point is referred to as a local salient patch, and each associated with a 64-dimensional feature vector. We use OpenSSL library to realize the symmetric encryption via AES-128 and the PRF via SHA-2. From the 1 million Flickr image collection, MIRFLICKR-1M [15], we choose 140,000 images with popular annotations as our experiment image dataset. We randomly pick 10% from our image dataset to build a 1000-word vocabulary via K-means clustering [9]. We further create 1 million users. For each user, we independently and randomly select 5 distinct images from the remaining 90% of our dataset.

### B. Image-centric Social Discovery Quality Evaluation

To evaluate our service quality of image-centric social discovery, we first select one target user, who has images of flowers and dogs to represent her social interests. We then try to find top five users with image profiles that are closest to this target user, among our created 1 million users. Our secure social discovery result is shown in Fig. 3. Those preferred images of recommended users are clearly visual similar with the images of the target user. They all have either flowers or dogs or both. Therefore, the recommended users can be considered as the potential social discovery candidates with mutual social interests to the target user. This demonstrates the consistency of our social discovery with human perception.

### C. Performance Evaluation

We now report our design from a number of performance factors. We will also compare our design with an existing work [21], which builds the secure index directly based on LSH hash tables, as we mentioned in Section III-B. We will denote it as KIK12, and show this approach does not work well with our large dataset of 1 million image profiles.

**User Client Overhead:** In our system, user client only processes a small number of preferred images. Thus, only a small computation cost (0.54sec for user image profile generation and 0.97msec for user metadata computation) and storage cost (1.03MB for visual word vocabulary) are introduced.

**Index Space Overhead Evaluation:** Fig. 4-(a) compares the space overhead between KIK12 and our proposed secure index. KIK12 suffers from the quadratic growing  $O(n^2)$  storage cost, while our index size  $O(n)$  is linearly increased. Here  $n$  is the number of users. Specifically, KIK12 contains the maximum number  $l * n$  buckets after random padding, where  $l$  is the number of LSH functions. Here each LSH bucket stores a  $n$ -bit binary vector to indicate which of the total  $n$  users are stored in the bucket. All  $n$ -bit LSH buckets are encrypted via symmetric encryption. Thus, the total size of the index in KIK12 is about  $l * n^2 / 8$  byte. The size of our index design is  $u * n / \tau$  byte, where  $\tau$  is the load factor and  $u$  is the size of one encrypted bucket  $B_i$ , 32byte as the output of SHA-2. In Fig. 4-(a), all space sizes are reported under parameters with  $l = 10$  and  $\tau = 80\%$ . For  $n = 1$  million, the index of KIK12 is around 1.13TB, while ours consumes only 38MB.































User	Preferred Images				
Query					
Top1					
Top2					
Top3					
Top4					
Top5					

Fig. 3: Social discovery results: the preferred images of one target user and the recommended top 5 users are exhibited respectively. It demonstrates the effectiveness of our “visual word” based image profile matching, which indicates that those users share mutual interests like flowers and dogs.

**Bandwidth Overhead Evaluation:** Fig. 4-(b) compares the service bandwidth cost between KIK12 and our design. In KIK12,  $SF$  sends one query with  $l$  trapdoors, and retrieves  $l$  encrypted binary vectors, each with  $n$ -bit, resulting in total  $l * n / 8$  byte. In our design, the search incurs constant bandwidth cost with respect to  $l * (d + 1)$  secure search trapdoors (32byte each) and retrieved encrypted image profiles (4KB each). For  $l = 10$  and  $d = 4$ , the bandwidth overhead is 201KB for the total 50 trapdoors and retrieved ciphertexts. As shown, even without the retrieved ciphertexts, KIK12 introduces 1220KB service bandwidth,  $6 \times$  more than ours, when  $n$  is 1 million.

**Query Latency Evaluation:** The query operation latency in our system is measured in Fig. 4-(c). The network delays are not considered. Fig. 4-(c) measures the operation latency under the revised index design which supports secure deletion and secure insertion. The search and deletion latency is both constant as they only require one round interaction between  $SF$  and  $CS$  with constant bandwidth cost determined by  $l$  and  $d$ . But as the secure deletion needs to further update the secure index via bucket re-masking (see Section III-D), its latency is slightly larger than the search operation. Since the insertion may involve multiple round interactions, the insertion latency is highly related to the number of kick-aways. Fig. 4-(c) also reports the total number of kick-ways for 100 data item insertions after we have already 1 million data in the index. Our experiment shows that the average number of kick-aways per insertion can be adjusted by setting the LSH parameters and index load factor. For the index with 1 million users already, only less than 1 kick-away per insertion is triggered on average when  $\tau \leq 80\%$ ,  $l = 10$ ,  $d = 30$ .

**Secure Index Building Cost:**  $SF$  can build our secure index in less than 1 minute for 1 million users as shown in Fig. 5-(a). But when the index load factor increases, more cuckoo kick-away operations will be triggered. Thus, the index building time becomes larger. Specific to our dataset, when we set load factor below 82%, we can get a reasonable performance in terms of both index building cost and secure operation latency.



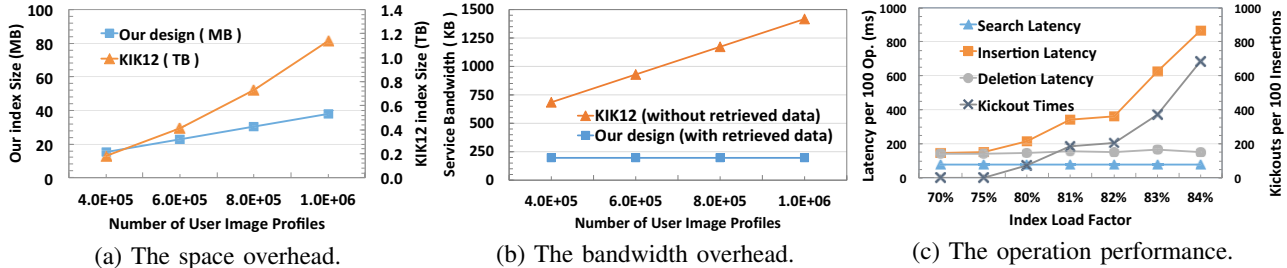


Fig. 4: Evaluation on space, bandwidth and operation overhead

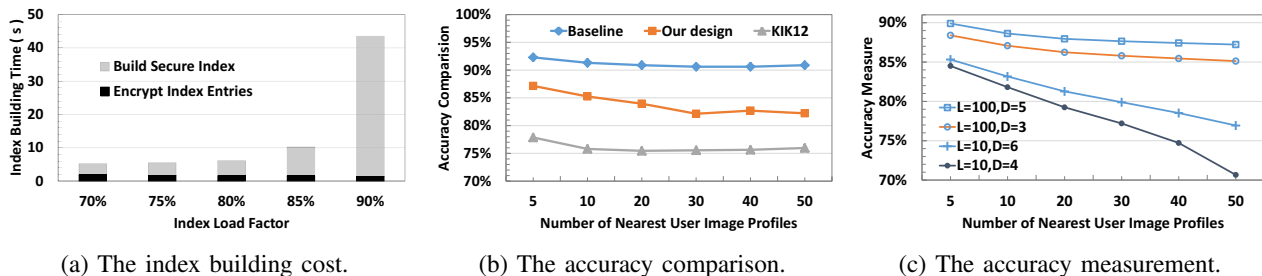


Fig. 5: Index building cost, accuracy measurement and analysis

**Accuracy Measurement:** We compare our discovery accuracy with KIK12 as well as a baseline approach. The baseline approach denotes the basic flow of using LSH for plaintext similarity search as introduced in Section II-C, while in KIK12, each item from the candidate set is directly ranked based on its occurrences in common LSH buckets. Our accuracy measurement from the average of 100 query image profiles  $\{S_q\}$  is shown in Fig. 5-(b). It is defined as  $\frac{1}{K} \sum_{i=1}^K \frac{\|S'_i - S_q\|}{\|S_i - S_q\|}$  [22]. It measures the closeness between the Euclidean distance of top- $K$  image profiles  $\{S_i\}$  and the ground truth  $\{S'_i\}$  via brute force method. Fig. 5-(b) shows that the accuracy of our results is higher than the accuracy of the score-based ranking method but lower than the accuracy of baseline for  $l = 10$  and  $d = 30$ . For baseline approach, it requires the distance comparison done over a very large set of potential candidates (5,000 according to our test). The larger set of candidates explains the better accuracy of the experiment result. For KIK12, the comparison shows that the number of occurrences in common LSH buckets is not always a good indicator for direct similarity ranking. We remark that our approach achieves reasonable accuracy measurement, yet always consumes much smaller and constant bandwidth. Fig. 5-(c) further demonstrates the impact of system parameters  $l$  and  $d$  on the accuracy of our design. In general, the more image profiles are retrieved, the more candidates can be obtained for ranking, leading to better accuracy measure but at the cost of bandwidth.

## VI. RELATED WORK

**Content-based Friend Recommendation.** Friend recommendation is one of the fundamental social network services. With the emerging popularity of social media, media content such as images have been used for self and social disclosure. One related approach defines the minimum cosine distance of two users' image features as user's visual similarity to assist

ranking friend candidates [16]. Another one called content-matching measures the distances of user feature vectors derived from user images [7]. Despite very useful, all these works do not consider the privacy protection on user images. In our research, we provide strong security guarantee for social discovery, which is conducted based on encrypted user images and the related encrypted image profile vectors.

**Privacy-preserving Social Profile Attribute Matching.** Our proposed research also relates to the branch of work on privacy-preserving profile matching [5], [6] that typically adopts one of two main approaches. One approach is private set intersection (PSI) and private cardinality of set intersection (PCSI) [5]. They treat users profiles as a set of attributes and compute the set intersection. When the intersection size is above a specific threshold, two users are identified as potential friends. The other approach considers a users profile as a vector [6]. Two users are considered potential friends when their secure scalar product is under a certain threshold.

However, all these approaches treat users' profiles as sets of numbers or even coarse-grained binary values, which do not necessarily express the versatile details of users' interests and/or experiences. Thus, it could be difficult to consistently provide good quality social discovery. Also, these approaches incur the computation/communication burden of friend finding directly to individual users, which might not be practically scalable with millions of users. Compared to them, our design focuses on image-centric approach with built-in security, and offload the storage and processing burden directly to cloud.

**Search over Encrypted Data.** Privacy-preserving image-centric social discovery is also closely related to the works on symmetric searchable encryption [12]–[14] and secure ranked search [27], [28], which support efficient and secure search over encrypted data. Yet, all these schemes only support textual files, but are not necessarily suitable for our image-centric

social discovery service. One recent work studied similarity search over encrypted data [21], which builds on direct combination of searchable encryption and locality-sensitive hashing based indexing. As mentioned in Section III-B, the indexing design suffers from the unscalable storage cost, and has very low space efficiency. It does not explicitly support data dynamics, and not always provide accurate search results.

Some related works [29], [30] on similar topic is built on metric index and special partition, both through computing distances with private pre-defined references or pivot objects. But the security strength is limited, as the private objects are subject to leakage when the attacker is able to adaptively exploit the search request generator. One arguably stronger design on encrypted hierarchical index was also proposed in [29], but it is subject to high bandwidth overhead and incurs considerable local burden for searching. Another work [24] finds nearest signals and similar data through secure multi-party computation on various distance metrics without knowing the contents. The model is different from ours. And these protocols introduce additional computation and bandwidth cost due to the relatively expensive public key encryption adopted.

## VII. CONCLUSION

We have designed and implemented a privacy-preserving image-centric social discovery system to expand user's friends with common interests effectively and securely. Our system is deployed under modern architecture, which leverages cloud as image storage back end. We first model user's social interests based on the image BoW representation, and then design a secure and compact similarity index to enable fast and scalable similarity search over millions of encrypted user image profile vectors. The security of our design is formally proved. Also, we present the service-driven applications on user image update and the compatibility with image sharing functionalities. Our evaluation demonstrates that our system is practical and efficient under huge image dataset and 1 million users. The illustrated social discovery results are of high quality and consistent with human perception.

In future, we plan to formalise and extend our secure user profile update protocols. We will also integrate other image related data like geolocation and tags to generate more comprehensive profiles for better social discovery services.

## ACKNOWLEDGMENT

This work was supported in part by Research Grants Council of Hong Kong under ECS grant CityU 138513, and by US National Science Foundation under grants CNS-1262277 and CNS-1262275. Portions of Dr. Squicciarini's work were supported by a Google grant.

## REFERENCES

- [1] A. Jeffries, "The man behind flickr on making the service 'awesome again'," 2013. <http://www.theverge.com/2013/3/20/4121574/flickr-chief-markus-spiering-talks-photos-and-marissa-mayer>.
- [2] R. Negoescu and D. Gatica-Perez, "Analyzing flickr groups," in *Proc. of Int'l Conf. on Content-based Image and Video Retrieval*, 2008.
- [3] Instagram Engineering Blog. Online at <http://instagram-engineering.tumblr.com/post/13649370142/what-powers-instagram-hundreds-of-instances-dozens-of>.
- [4] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.
- [5] M. Nagy, E. De Cristofaro, A. Dmitrienko, N. Asokan and A. Sadeghi, "Do I Know You? - Efficient and Privacy-Preserving Common Friend-Finder Protocols and Applications," in *Proc. of ACSAC*, 2013.
- [6] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *Proc. of INFOCOM*, pp. 1647–1655, 2011.
- [7] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy, "Make new friends, but keep the old: recommending people on social networking sites," in *Proc. of ACM CHI*, pp. 201–210, 2009.
- [8] M. McPherson, L. Smith-Lovin and J. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, pp. 415–444, 2001.
- [9] J. Yang, Y. Jiang, A. Hauptmann and C. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proc. of ACM Workshop on Multimedia Information Retrieval*, pp. 197–206, 2007.
- [10] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, pp. 117–122, 2008.
- [11] R. Pagh and F. Rodler, "Cuckoo hashing," *J. Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.
- [12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of CCS*, 2006.
- [13] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, "Computationally efficient searchable symmetric encryption," in *Proc. of the VLDB workshop on Secure Data Management*, pp. 87–100, 2010.
- [14] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. of CCS*, pp. 965–976, 2012.
- [15] M. Huiskes and M. Lew, "The MIR flickr retrieval evaluation," in *Proc. of ACM Int'l Conf. on Multimedia Information Retrieval*, 2008.
- [16] T. Yao, C.-W. Ngo, and T. Mei, "Context-based friend suggestion in online photo-sharing community," in *Proc. of ACM Multimedia*, 2011.
- [17] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: an online social network with user-defined privacy," in *Proc. of SIGCOMM*, pp. 135–146, 2009.
- [18] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access control in cloud computing," in *Proc. of INFOCOM*, pp. 1–9, 2010.
- [19] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," in *Proc. of VLDB*, pp. 950–961, 2007.
- [20] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between euclidean and cosine angle distance for nearest neighbor queries," in *Proc. of the ACM symposium on Applied computing*, pp. 1232–1237, ACM, 2004.
- [21] M. Kuzu, M. Islam and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proc. of IEEE ICDE*, pp. 1156–1167, 2012.
- [22] Y. Hua, B. Xiao, and X. Liu, "Nest: Locality-aware approximate query service for cloud computing," in *Proc. of INFOCOM*, 2013.
- [23] Florin Raiu, "People You May Know." Online at <https://blog.facebook.com/blog.php?post=15610312130>, 2009.
- [24] C. Blundo, E. De Cristofaro, and P. Gasti, "Espresso: Efficient privacy-preserving evaluation of sample set similarity," in *Proc. of Data Privacy Management*, pp. 89–103, Springer, 2013.
- [25] C. Wang, K. Ren, S. Yu and K. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. of INFOCOM*, pp. 451–459, 2012.
- [26] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [27] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467–1479, 2012.
- [28] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.
- [29] M. Yiu, I. Assent, C. Jensen, and P. Kalmis, "Outsourced similarity search on metric data assets," *IEEE Trans. on Knowledge and Data Engineering*, vol. 24, no. 2, pp. 338–352, 2012.
- [30] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. of IEEE ICDE*, pp. 733–744, 2013.