# Privacy-friendly Photo Capturing and Sharing System

**Lan Zhang**[1,2], **Kebin Liu**[2], **Xiang-Yang Li**[1], **Cihang Liu**[2], **Xuan Ding**[2], **Yunhao Liu**[2]
[1]Department of Computer Science and Technology, University of Science and Technology of China
[2]School of Software and TNLIST, Tsinghua University, China
{zhanglan03,xiangyang.li}@gmail.com,{kebin,cihang,xuan,yunhao}@greenorbs.com

## ABSTRACT

The wide adoption of smart devices with onboard cameras facilitates photo capturing and sharing, but greatly increases people's concern on privacy infringement. Here we seek a solution to respect the privacy of persons being photographed in a smarter way that they can be automatically erased from photos captured by smart devices according to their requirements. To make this work, we need to address three challenges: 1) how to enable users explicitly express their privacy protection intentions without wearing any visible specialized tag, and 2) how to associate the intentions with persons in captured photos accurately and efficiently. Furthermore, 3) the association process itself should not cause portrait information leakage and should be accomplished in a privacy-preserving way. In this work, we design, develop, and evaluate a system, called *COIN* (Cloak Of INvisibility), that enables a user to flexibly express her privacy requirement and empowers the photo service provider (or image taker) to exert the privacy protection policy. Leveraging the visual distinguishability of people in the field-of-view and the dimension-order-independent property of vector similarity measurement, COIN achieves high accuracy and low overhead. We implement a prototype system, and our evaluation results on both the trace-driven and real-life experiments confirm the feasibility and efficiency of our system.

### Author Keywords

Photo Capturing and Sharing; Portrait Privacy; Smart Camera

### ACM Classification Keywords

K.4.1 Computers and Society: Public Policy Issues; K.6.5 Computers and Society: Security and Protection; C.2 Computer-communication Networks

## INTRODUCTION

Nowadays, smart devices with onboard cameras *e.g.*, smart phones and glasses [29], are pervasive in our daily lives. Current smart devices can capture and even share photos anytime anywhere without informing the parties in the photo, thus raising many concerns on people's privacy infringement. Secretive photographing without clear warning beforehand is privacy violation. Even worse, if the secretively taken photos which contain information beyond what users want to reveal are shared on Internet, it will make users extremely susceptible to various attacks.

To protect people's portrait privacy from unwilling photo-taking and publication, many photo service providers or users have taken actions in different ways. For example, some Glass wearers whip their device off in inappropriate situations, such as in gym locker rooms or work meetings; some business bans smart glasses inside their buildings to respect customers' privacy; and the Glass manufacturer (*e.g.*, Google) does not allow developers to create applications that take photo silently. Those methods, however, are broad-brush and blunt which can significantly hurt the applications of smart devices. Therefore, it is appealing to consider how one might build a system which respects people's portrait privacy while guaranteeing a comfortable usage of smart glasses/cameras.

Instead of discarding the smart glasses/cameras due to privacy concerns, in this work, we seek a solution for reaching an ultimate goal of privacy-aware Glass/camera, operating transparently to end users. Our solution will let end users to express their privacy requirements and glasses/cameras or photo service providers will exert the privacy protection mechanisms. When taking a photo/video, the smart device will detect who (in the picture/video) requested privacy protection, and then remove them from the image automatically. Our protocol can also improve the social augment application by automatically tagging a user in a photo when he/she expresses an interest to share his/her information with surrounding people.

To implement such a privacy-aware camera, we need to address several critical challenges. **First**, we should enable the user efficiently and flexibly to express his/her intention/requirements conveniently. Some methods require the user to wear visible specialized tag (*e.g.*, QR code [3]), which is inaesthetic and inconvenient. In this work, we propose a method to encode the user's portrait feature in the request and transmit it using wireless devices. Then, the **second** challenge is that we should accurately and efficiently associate each privacy-seeking user with an image region in the photo taken by another user (the photographer ). Furthermore, the association process itself should not cause portrait information leakage and should be accomplished in a privacy-preserving way. Face recognition is widely used to identify people in photos, but in practice it suffers when there is a lack of a clear front view of faces due to the camera's view angle and dis-



*UbiComp '16,* September 12-16, 2016, Heidelberg, Germany
 ISBN 978-1-4503-4461-6/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971662

tance. Sophisticated but complicated matching schemes may cause high overhead and long delay. The matching problem itself is difficult due to the accuracy and efficiency requirements, let alone completing matching process in a private and non-interactive manner with untrusted server. Matching a user's privacy-expression with a possible people in a photo can be reduced to some sort of vector matching. Many private vector matching protocols use multi-party computation techniques, which require frequent interactions among participants. Most existing private vector matching methods (in both multi-party computation and outsourced manner) use homomorphic encryption [12, 20] or garble circuit [20], and cause high computation cost for both client and cloud. The **third** challenge is that the privacy-friendly Glass/camera should be transparent to all users and minimize extra overhead to mobile devices. An ad hoc approach may lead to requirements for "always-on" neighbour discovery, frequent information exchanging as well as complex image matching computation on user devices. To reduce the overhead of users, our protocol will outsource most of these tasks to cloud with a well-designed strategy to prevent privacy leakage to untrusted cloud and other users.

The main contributions of this work are as follows:

- To the best of our knowledge, we are the first to present a portrait privacy preserving photo capturing and sharing system. We comprehensively analyze the privacy issues during the photo capturing and sharing and define three types of threats. With our approach, people who do not want to be captured in photo will be automatically erased from the photo and verification of the removal is also supported in case the photographer ignores the request.

- For accurate and efficient matching between people's privacy intentions and people in the photo, we design a graph-based portrait profile and a robust matching algorithm. The graph representation of portrait is sufficiently distinguishable and better-formed for storage and matching than the image. Moreover, our matching mechanism is resistant to pose changes of people and camera, and also compatible for the future development of vision feature description.

- We propose a novel highly efficient *encryption-free* privacy-preserving vector distance protocol in a non-interactive manner with untrusted server. Based on our observation, the dimension-order-independent property of distance between vectors, we enable the distance computation on transformed vectors other than cypher blocks, which significantly reduces the computation complexity and communication cost than existing crypto system based solutions. With our protocol, most computation tasks are transferred from smart devices to the cloud in a privacy-preserving way, meanwhile the interaction cost is significantly saved.

- We design and implement a prototype system and verify the effectiveness of our scheme by extensive experiments as well as case studies.

## RELATED WORK

Ubiquitous availability of smart devices with onboard cameras has resulted in photos being captured and shared online at an unprecedented scale. It is important to respect people's portrait privacy and protect them from unwilling photo-taking and publication. Some actions have been taken in industry and business areas, *e.g.*, forbidding silent photo taking or camera usage in inappropriate situations, which are broad-brush and blunt. There are very few existing work addressing this issue in academic area. The most related work to ours is [3], which requires users to wear visible specialized tags (e.g., QR code) to express their requirements and blurs faces in photos accordingly. There are a few limitations of [3]: first, visible tags on cloth are inaesthetic and inconvenient; second, it considers only face as private information, but a portrait not only includes the user's face but also his/her body, since clothes and accessories could also reveal identification information; third, it assumes a trusted server and transmits each user's portrait image in its original form (visible form), which could cause privacy leakage to untrusted servers and eavesdroppers; besides, without verification mechanism, a photographer can simply ignore the request. In this work, we seek a solution to protect people's portrait privacy in a non-intrusive way while guaranteeing a comfortable usage of smart glasses/cameras. The whole protection process should be conducted in a privacy-preserving manner even with untrusted server. Also verification of the protection should be supported in case the photographer ignores the request. Our work are related to existing research in the following aspects.

### Image Privacy Protection

A number of approaches have been designed to protect image privacy. There is a trivial solution to protect image content. Blacking out private contents, *e.g.* human faces, thwarts any possible violation of owners' privacy. For example, systems like Blinkering Surveillance [21] use computer vision methods to hide sensitive contents from video frame. GigaSight [23] blacks out sensitive information from video frames. But in a photographing scenario, the challenge is how to match people's privacy requests with people in the photo. Face recognition is an alternative way to solve the matching problem. Many face descriptors are proposed for face recognition, *e.g.* Eigenfaces [24] and Fisherfaces [2]. To use face recognition approaches, it requires the people to face to cameras. Besides, during the information exchange process, face descriptors could be leaked to adversaries. There are some work providing privacy-preserving face recognition, by which a client can privately search for a specific face image in the image database. [9] leverages homomorphic encryption to recognize a face in a database of $M$ faces. [20] improves the scheme with cryptographic building blocks combing homomorphic encryption with garbled circuits. [27, 28] enable outsourced content-based image search against untrusted service provider. Those methods provide privacy protection to the requested images as well as the outcome of the matching algorithm, but the computation overhead is large.

### Privacy-Preserving Vector Distance Computation

A key step for privacy-preserving photo capturing is matching users in the photo with users who request privacy protection. The matching process can be considered as a sort of distance

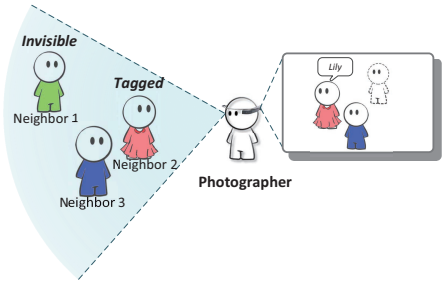


Figure 1: Example application scenario of COIN : the tagged person is labeled in the photo and the invisible person is erased from the photo.

computation between vectors. To achieve privacy preserving match, many existing protocols use multi-party computation (SMC) [12, 20] or garble circuit [20] to compute Euclidean distance between vectors. They, however usually require frequent online interactions among data owners. Moreover, their large computation cost and ciphertext size make them unsuitable for mobile applications. [30] designs a light weight symmetric encryption based vector matching protocol, but it cannot be adopted for distance computation. [19] proposes an approach using Fourier-related transforms to hide accurate data values and to approximately preserve Euclidean distances among them. It works well for some data mining purpose on large datasets, but the transformation is public and deterministic and it cannot prevent malicious user from dictionary attack. Instead, we propose a novel highly efficient *encryption-free* privacy-preserving vector distance protocol in a non-interactive manner with untrusted server.

## MOTIVATION AND PRIVACY REQUIREMENTS

### Motivation

In this work we seek a solution to respect the privacy of persons being photographed in a smarter way. As an example shown in Fig. 1, when someone uses his smart Glass to take a photo, people in the field of view (FOV) should be notified (or the photographer should know the privacy protection intentions of people in FOV). Then persons who are unwilling to be photographed, *e.g.* Neighbor 1, should have a convenient way to specify their privacy intentions, and thus be automatically erased from the photo. We refer to them as *invisible users*. Users who would like to make friends with the photographer, *e.g.*, Neighbor 2, can be automatically tagged on the photo and share information. We refer to them as *tagged users*. The photographer just takes other people into the photo as usual. The system can motivate the photographer in two aspects: (1) when respecting other invisible users' privacy, his/her privacy can also be protected by others; (2) the system supports tagging people automatically, which could be helpful and fun in many scenarios (*e.g.*, social applications). Photo sharing services can benefit from this system by attracting more users who would like to be invisible or tagged in photos. Besides, after one time setting the solution should be transparent to all users and avoid incurring high overhead to their smart devices. Finally, the portrait privacy protection strategy should be well designed and avoid further leakage of any type of private information.

### Threats to Portrait Privacy

People may be photographed nearly anytime anywhere without their consent. Photos contain rich information, including people's appearance, location, activities, *etc.*. Facing massive cameras and image analysis techniques [16,24] , people's portrait privacy is badly in need of protection. In this work, we focus on protecting users' portrait information. Here, a portrait not only includes the user's face but also his/her body, since clothes and accessories could also reveal identification information. We consider three types of threats to portrait privacy, which expose sensitive information to different extend.

- **Visual portrait privacy.** The most intuitive way to violate a user's portrait privacy is to capture and publish (*e.g.*, through photo sharing systems) a photo containing his/her visible portrait. Simply blurring all faces in images, *e.g.*, [23] and Google Street View, will disable the normal photographing function. In our protocol COIN we propose to match the people in the photo to their privacy protection intentions, and erase only people that should be invisible. As we will discuss in detail in Section System Design Overview, a user expresses his/her privacy requirement by encoding his/her portrait, which clearly cannot be transmitted in its original form (otherwise his/her portrait privacy is broken by himself/herself).

- **Portrait feature privacy.** This type of threats occur inside some image services, *e.g.* image matching or face recognition. These services don't use visible images directly, but take feature vectors of image as the descriptor, *e.g.*, Eigenfaces [24] and color histogram. But users can also be identified by features of portrait image. For example, face images can be reconstructed from face vectors [6]. During the process, the leakage of portrait features also violates users' privacy.

- **Inference privacy.** Even if an image system hides original images and other information such as their feature vectors, an adversary with a collection of images (an image dictionary) can infer the hidden content using the similarity measurement function of the system. Hence, we should prevent adversaries from obtaining the similarity measuring results to enhance the privacy protection.

There are also other types of user privacy should be respected, *e.g.* location privacy. A lot of methods have been proposed to provide privacy-preserving location services [17]. COIN leverages existing solutions to protect other user privacy since it is not the focus of this study.

### Adversary Model

Our approach defends a user's portrait privacy against both untrusted cloud server and malicious users. For the cloud, we apply the widely used "honest-but-curious" assumption. The cloud server will follow the protocol, but might conduct extra work to harvest portrait images of invisible people, reconstruct invisible portraits using feature vectors or infer the invisible content using image dictionary. This is a justifiable assumption because deviating from the protocol will lead to poor user experience, thus could cause revenue loss of the service provider. Also, we assume that the cloud won't collude

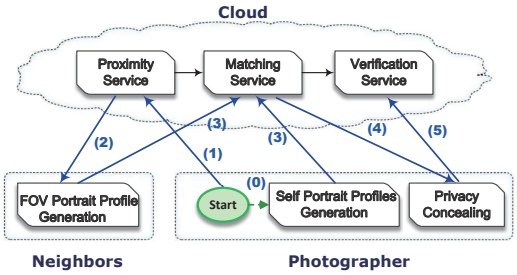


Figure 2: Baseline system architecture (COIN ).

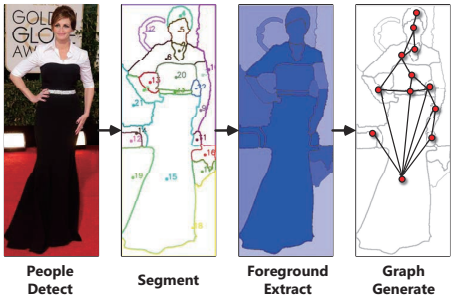


Figure 3: Portrait graph representation.

with any client to conduct an attack. A malicious user could participate in harvesting other users' portrait information by eavesdropping their communication with the cloud. All users except the photographer should be prevented from obtaining the portrait information of invisible users. Although the photographer already owns the photo of invisible people, he/she may misbehave to preserve the invisible people who should be erased and publish the photo through Internet. So, we also need a verification scheme against dishonest photographers.

## SYSTEM DESIGN OVERVIEW

Facing critical challenges introduced in Introduction. we design our system to achieve both the privacy and system efficiency goals. We firstly present our baseline system design without privacy-preserving computing. With our graph-based portrait matching algorithm, the baseline approach COIN is effective to protect users' visual portrait privacy by accurately matching invisible people in photos and erasing them automatically. COIN is sufficient when the server is trusted and the communication channel is secure. But in practice, there is a risk of exposing portrait features to untrusted cloud and other participants. Furthermore, we propose an efficient privacy-preserving outsourced vector distance protocol, based on which an advanced approach COIN++ provides portrait feature privacy and inference privacy protection with little extra overhead for the client. In this section, we will sketch our system architecture.

### Overview of Baseline System COIN

There are three parties involved in our system: the photographer, who takes the photo; the neighbors, who could be in the FOV of the photographer (as presented in Fig. 1); the cloud, who takes charge of location, communication and computation services. The architecture and workflow of our baseline system are illustrated in Fig. 2.

We would like to take a typical photographing process as an example to describe the functionality of each component and the system workflow. Each user creates his/her personal portrait profile using the *Self Portrait Profile Generation* component and encode his/her portrait profile to express his/her privacy requirement, *e.g.*, he/she can choose his/her status from "invisible me" (or "tag me") in the mobile phone app to make himself/herself an invisible (or tagged) user. In our design, a set of vision feature vectors are extracted from subregions of a portrait image. Both face and body features are extracted, in case that there may be a lack of clear front view of face. We design a graph structure to encode the extracted vision features (feature vectors are properties of nodes) and use the graph as the portrait profile, as the examples in Fig. 3. Compared with uploading the original image, the feature graph shows a low risk on privacy leakage without lose of matching functionality, and are much more efficient for both computation and communication. Besides, the graph representation is highly robust against pose changes of people and cameras. For each invisible user, his/her self portrait profile is generated once and for all until he/she updates it. While the face features of a user remains the same, the user could change his/her outfits. The portrait profile could be automatically updated when the user takes a selfie or while he/she uses the phone with the frontal camera facing himself/herself. In our advanced approach, transformed version of portrait graph are used to improve privacy protection, and we convert the people matching problem to graph matching problem.

Triggered by a photo shooting action, the *Proximity Service* on cloud will automatically start to check if there are invisible people in the FOV of the photographer. If any, the cloud will inform them and start the next step. Proximity service can be realized easily using existing location service and onboard compass. For example, Wi-Fi based localization can achieve 1-2 meter level accuracy [18]. It is not always convenient to recognize an accurate FOV, which needs parameters and the current orientation of the camera. In practice, the cloud could efficiently search invisible neighbors (located within a certain distance) of the photographer and take them as potential people appearing in the photo. With limited number of potential matched invisible users, our system can achieve high matching accuracy and low overhead.

In the next step, after being informed by cloud, invisible neighbors upload their self portrait profiles to cloud (this could be done in advance to reduce the delay). Meanwhile, the photographer detects all people in the photo and extracts their portrait profiles with the *FOV Portrait Profiles Generation* component, which works similarly to self portrait profile generation. These profiles will be uploaded to cloud as well. Then the *Matching Service* will match portrait profiles of invisible users to portrait profiles from the photo and determine people that should be erased from the photo. The graph matching algorithm will be discussed in detail in the following section. The matched results will be sent to photographer, and then the *Privacy Concealing* component will erase the corresponding image regions of invisible people from the photo automatically by blurring or other more sophisticated

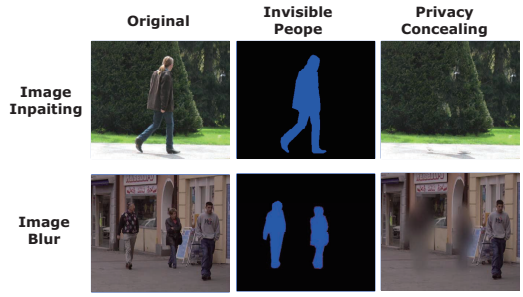


Figure 4: Example of automatic privacy concealing (erase invisible people from photos) by image inpainting [5] or blur.

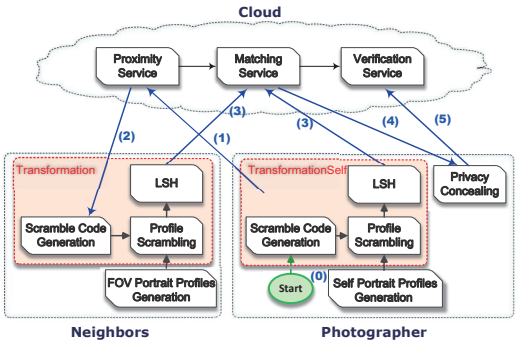


Figure 5: Advanced system architecture (COIN++ ).

techniques, like image inpainting [5,13], to maximize the aesthetics. We show an example of removing invisible people from photo in Fig. 4. For the inpainting, we use the code from Criminisi's work [5]. After the removal, the photographer can store or share the photo using the cloud service. Note that, based on the personal specification, the whole procedure works the same way for tagged users and the "erase" operation can be alternated to "tag" to augment many social applications.

In case there are dishonest photographers who ignore the request or don't complete the removal, COIN supports verification of removal. All invisible users' portrait profiles have been uploaded to the cloud in the previous stage. Once the photo is shared through Internet, the *Verification Service* will check the photo as follows the cloud first conducts a people detection on the photo and extracts all portrait profiles; then the cloud matches these profiles with the cached profiles of invisible neighbors, if there is a matching, it can tell that the photographer didn't follow the protocol. The verification process can be completed alone by the cloud without any interaction with users.

### Overview of Advanced System COIN++

The baseline protocol protects the visual privacy of people's portraits, but exposes users' portrait profile (*i.e.* feature vectors) to the cloud and even the eavesdroppers. With some feature vectors an adversary could have a chance to match them with existing photos or even reconstruct the photo. In the advanced approach, we retain the visual portrait privacy protection and improve the system to protect users' portrait feature privacy and inference privacy. In other words, the graph based profile matching scheme should be conducted in a privacy-preserving manner. The core of the portrait profile matching algorithm is to measure the distance between vision feature vectors. We cannot directly adopt existing privacy-preserving vector distance protocols based on homomorphic encryption [12, 20] and garbled circuit [20] due to their large computation and communication cost. In our system we propose a highly efficient outsourced vector distance protocol (see Section Privacy Enhanced System). As shown in the red blocks in Fig. 5, based on our observation, the dimension-order-independent property of distance between vectors, we propose a well designed scrambling scheme and combine it with locality sensitive hash. With our design, all invisible neighbors can secretly transform their vectors in a distance preserving way. Then the cloud can measure vector distances using the transformed vectors and match transformed portrait graph with the same algorithm as in the baseline system. Our scheme protects invisible users' portrait profiles (from both himself/herself and the photographer) with little extra cost on the client side. But, it saves computation cost for the cloud, because the distance computation of high-dimensional real number vectors is converted to distance of low-dimensional binary hash code.

The architecture of the advance system is presented in Figure 5, except vector transformation and verification, other components are the same as that in the baseline system. Here, the verification is more challenging, because the cloud only knows the transformed portrait graphs of invisible users. Without knowing the secret transformation, the cloud cannot compare them with portrait graphs extracted from the uploaded photo directly. When an invisible user needs to check if his/her portrait has been removed, he/she need to start a verification and participate in the process as follows: the cloud sends all extracted feature vectors in a random order to the invisible user. Note that, these feature vectors are supposed to belong to preserved visible people if the photographer is honest. And the invisible user transforms them in the same way as his/her self feature vectors and sends the results transformed vectors to the cloud. Then the cloud can compare preserved people in the photo and invisible neighbors using transformed portrait graphs, and detect the dishonest photographer.

## PORTRAIT PROFILE GENERATION AND MATCHING

### Portrait Profile Generation

In this work, we design a distinguishable graph representation of portrait. Compared with the original pixel image, portrait graph extracts components of the portrait and describes their connectivity, hence it is more robust to changes of the person's pose and the camera's view angle. Portrait graph is also more efficient for storage, transmission and matching, and shows a lower risk on privacy leakage. After applying people detection on a photo [16, 25], we obtain portrait images (including both people faces and bodies) from the photo as shown in the first subfigure of Fig. 3. Then portrait image can be segmented into adjacent regions by different colors and textures [1, 8]. Given one portrait image, a graph $G = (V, E)$ can be constructed, where $V$ is a set of nodes

representing segmented regions and $E$ are edges connecting any two regions that share a boundary. Then we measure each node's confidence of being a part of the person. The first cue is the extend that the region shares its boundary with the border of the portrait image, which gives an evidence that a region doesn't belong to the entity. The second cue is the distance from the mass center of a region to the center of the portrait image, which gives an evidence that a region belongs to the entity. The confidence of a node is obtained by fusing evidences using Dempster Shafer (DS) theory [22], and we remove the node with low confidence to eliminate the background. Fig. 3 shows examples of foreground extraction and portrait graph generation, which provide more accurate graph representation of people portrait. The result of foreground extraction can also be employed by the privacy concealing component as the accurate erase area to achieve better looking removal, as shown in Fig. 4. For each region of portrait, vision feature vectors, *e.g.*, face feature vector (*e.g.*, eigenfaces vector [24] ), color histogram and texture vector, are extracted as property of the corresponding node. We will give more details about node properties in the implementation section.

#### Portrait Graph Matching Scheme

To achieve accurate and efficient portrait graph matching, there are several challenging issues should be addressed with low computation cost: graph structure of the same person varies due to changing illumination condition and viewpoint; incomplete graphs could be produced due to occlusion; portrait profile could still contain some noisy nodes from background. As a result, the matching algorithm should be elastic to node/edge division, aggregation, insertion and deletion, and robust to noise nodes. Existing graph matching methods usually have application-oriented specifications [10, 11, 26], *e.g.*, assumptions about node numbers, graph structure and pre-knowledge of correspondences, making them difficult to be directly applied in this work. To meet the critical requirements of portrait profile matching, we design a voting based strategy in which both the node similarity and graph structure are considered.

Let graph $G^x = (V^x, E^x)$ denote portrait profile $X$ (say produced by a user) and $G^y = (V^y, E^y)$ denote portrait profile $Y$ (say produced by a photographer). Here $V^x = \{v_1^x, v_2^x, ..., v_p^x\}$ and $V^y = \{v_1^y, v_2^y, ..., v_q^y\}$. Each node owns some feature vectors as its property. In order to improve matching accuracy as well as speed up the computation process, we add a *label* to each node according to the result of face detection, which describes its type, for example, *human face* or *human body*. Only nodes of the same type can be matched. The similarity between two nodes of different types is zero. As human face is a strong feature to identify a person, our matching scheme will firstly consider the matching between nodes labeled with "human face" (*e.g.*, Node No.5 in Fig. 3), then invoke an integrated graph matching. In this way, our method provides more accurate and robust matching than existing face recognition based methods.

**Initialization.** Let the similarity between nodes $v_i^x$ and $v_j^y$ be $\mathbf{S}(v_i^x, v_i^y)$, which can be obtained through measuring the distances between feature vectors of two graph nodes. Note that, if $v_i^x$ and $v_i^y$ have different type labels, $\mathbf{S}(v_i^x, v_i^y)$ is set to zero. In the next section, we will discuss the details of privacy preserving vector distance computation. During the matching process, a matrix $M$ with $p$ rows and $q$ columns is built. Each entry $M_{ij} = \{f_{ij}, n_{ij}, c_{ij}\}$ of the matrix is a triple where $f_{ij}$ is a boolean flag indicating whether node $v_j^y$ is a possible match for node $v_i^x$, $n_{ij}$ caches the one-hop neighbor match information and $c_{ij}$ is a counter. Details of these parameters will be presented in the following parts. A match is represented as an assignment for all $\{f_{ij}\}$, where there is at most one $f_{ij}$ equaling $TRUE$ for every column $j$. All $\{f_{ij}\}$ are initiated to $TRUE$.

After the initialization, our graph matching scheme consists of three stages.

**Stage 1.** We eliminate wrong matches based on the similarities of node pairs. If $\mathbf{S}(v_i^x, v_i^y)$ is above a pre-specified threshold $\xi_s$, the corresponding flag $f_{ij}$ is set to $TRUE$, otherwise, we eliminate this match by setting $f_{ij}$ to $FALSE$. Note that, a node in $V^x$ does not necessarily have a possible match in $V^y$, thus there can be rows with all $FALSE$ flags. After this stage, all node pairs with $TRUE$ flags are considered as *candidate matches*.

**Stage 2.** We explore the one-hop neighbor matching for each candidate match. For each candidate match $(v_i^x, v_j^y)$, the neighbor sets of $v_i^x$ and $v_j^y$ are denoted as $NE(v_i^x)$ and $NE(v_j^y)$. We find the most likely mapping from $NE(v_i^x)$ to $NE(v_j^y)$. To achieve this, we firstly look for potential matches in matrix $M$ for each node in $NE(v_i^x)$. We then connect each node in $NE(v_i^x)$ with its matched nodes in $NE(v_j^y)$ with undirected edges. Nodes in both sets as well as the edges form a bipartite graph and the problem can be transformed to find a maximum match on the bipartite graph. To address this problem, we apply the Hungary algorithm [14] which outputs a mapping from $NE(v_i^x)$ to $NE(v_j^y)$. As mentioned above, the mapping is denoted as $n_{ij}$.

$$n_{ij}(v_a^x) = \begin{cases} v_b^y & \text{if } v_a^x \text{ matches } v_b^y \\ \Phi & \text{if there is no match in } NE(v_j^y) \text{ for } v_a^x \end{cases}$$

where $v_a^x \in NE(v_i^x)$ and $v_b^y \in NE(v_j^y)$.

**Stage 3.** We choose at most one assignment for each node in $V^x$ by a voting based scheme. For each candidate match $(v_i^x, v_j^y)$, we build two trees rooted at $v_i^x$ and $v_j^y$ on graph $G_x$ and $G_y$ respectively. The two trees are traced in parallel on two graphs with the BFS method. Here we restrict the tree growth to the constraint that, once a node $v_k^x$ on $G_x$ and its matched node $v_g^y$ are appended to the trees, the neighbors of $v_k^x$ which have not been included can be added to the tree only if they have matched nodes in $NE(v_g^y)$ according the recorded mapping $n_{kg}$. When two trees have grown to the maximum size, we get a possible match for the subgraphs. In this approach, we propose a voting scheme to determine the best match. That is, for each candidate match $(v_k^x, v_g^y)$ on the two trees, we increase the counter value of $c_{kg}$ in entry $M_{kg}$. After trees of all candidate matches $(v_i^x, v_j^y)$ voted, we check the $c_{ij}$ in each entry $M_{ij}$ and retain the largest one for each column. Then matrix $M$ indicates a most likely match of $G^x$

and $G^y$ and the similarity between the two portrait profiles are calculated by integrating similarities of all matched nodes and edges.

$$\mathbf{S}(G^x, G^y) = \frac{\sum_{f_{ij}=\text{TRUE}} \mathbf{S}(v_i^x, v_j^y)}{\parallel V^x \parallel + \parallel V^y \parallel} + \frac{\sum_{e_{ab}\in E^x} \sum_{e_{cd}\in E^y} \delta(e_{ab}, e_{cd})}{\parallel E^x \parallel + \parallel E^y \parallel}$$

where

$$\delta(e_{ab}, e_{cd}) = \begin{cases} 1 & \text{if } f_{ac} = TRUE \ \& \ f_{bd} = TRUE \\ 0 & \text{otherwise} \end{cases}$$

## PRIVACY ENHANCED SYSTEM

We enhance our system to protect users' portrait feature privacy and inference privacy by conducting the portrait graph matching in an outsourced and privacy-preserving manner, whose core is measuring the Euclidean distance between feature vectors privately and efficiently. As one of the main contributions, we propose a novel highly efficient *encryption-free* privacy-preserving vector distance protocol in a non-interactive manner against untrusted server. We gain the chance by observing that the distance between two vectors is independent to their dimension order, since the vector distance is measured in a dimension-wise way. Based on the observation, we propose to transform the original vectors to randomly ordered vectors in a distance preserving way, and keep the transformation a secret to adversaries. This design enables us to measure distance on transformed vectors as on original vectors (which means light-weight computation), but the adversary cannot obtain the original vectors nor compute the distance between the transformed vectors and vectors from a dictionary to infer the original ones.

Let the distance function of two vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots), \mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \cdots) \in \mathbf{V}$ be $\mathsf{d}(\mathbf{x}, \mathbf{y})$. As shown in Fig. 5, we design the transformation with two main building blocks: *Profile Scrambling* and *Locality Sensitive Hash (LSH)*. The profile scrambling module works based on our observation that vector distances are *dimension-order-independent*, that is when we randomly change the dimension order of both $\mathbf{x}$ and $\mathbf{y}$ consistently to obtain scrambled $\mathbf{x}'$ and $\mathbf{y}'$, we have $\mathsf{d}(\mathbf{x}, \mathbf{y}) \equiv \mathsf{d}(\mathbf{x}', \mathbf{y}')$. Once the scrambling order is kept secret, the original vectors are protected and a dictionary based inference is prevented. In case there may be some dimension-dependent characteristics of vision feature vectors, *e.g.*, in the color histogram the dimensions representing red component usually have large values, we employ the LSH module to transform the scrambled feature vectors into another low-dimensional vector space. LSH hides the scrambled feature vectors from all parties and makes the statistic analysis on scrambled vectors infeasible, meanwhile it also preserves the distance among vectors. Besides, lower-dimension vectors reduce the cost for vector distance computation and vector transmission. Moreover, changing the dimension order of $\mathbf{x}$ randomly to $\mathbf{x}'$ makes their hashes totally different, because there is a random distance between them. Hence, the vector scrambling works like a random salt, which strengths the security property of LSH and makes the dictionary attack against LSH infeasible. Combining vector scrambling and LSH, we protect feature vectors of invisible users from untrusted cloud and other parties, and outsource most computation to the cloud in a secure and noninteractive manner.

Before we present our privacy-preserving vector distance protocol, we firstly introduce the LSH based vector distance measurement as a preliminary. The key insight behind LSH is that it is possible to construct hash functions such that close vectors will have the same hash value with higher probability than vectors that are far apart. Different LSH functions are designed for various distance metrics, *e.g.*, Euclidean distance, Hamming distance, cosine distance. In our system, we use the commonly used Euclidean distance $\mathsf{d}_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (\mathbf{x}_i - \mathbf{y}_i)^2}$. Particularly, for high-dimensional vector space $\mathbf{V} = R^D$ with Euclidean distance, an LSH function is defined as follows [7]:

$$\mathbf{H}(\mathbf{x}) = < h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_m(\mathbf{x}) > \quad (1)$$

$$h_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \frac{\mathbf{a}\cdot\mathbf{x}}{W} \geq 1, i = 1, 2, \cdots, m \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $\mathbf{a} \in R^D$ is a random vector with each dimension chosen independently from the standard Gaussian distribution $N(0, 1)$. Here each $h_i$ is an atomic LSH function, and the LSH function $\mathbf{H}$ generates a hash vector of the input vector by concatenating $m$ scalar atomic hash values. The window size $W$ and $m$ control the distance range that the mapping is sensitive to. In the advance system, the cloud determines the hashing function $\mathbf{H}$ and publishes it to all participants. According to the definition of LSH, the differences between hash vectors indicate the distance between original vectors. In this work we apply the Hamming distance between hash vectors to approximate the distance between original vectors. Our experimental results show that the Hamming distance between hash vectors is nearly monotonic to distance measurement between original vectors.

### Outsourced Privacy-preserving Distance Computing

Here, we assume that each user shares a secure communication channel with the cloud. Then the transformed vectors are protected from other participants. In a specific round of photographing, to preserve distance between transformed vectors, the challenge is that all participants (photographer and invisible neighbors) must scramble their feature vectors in a consistent order individually and secretly. We refer to the scramble order as scramble code $\mathbf{SC}$. To achieve the same $\mathbf{SC}$, all participants first need to generate a same random seed $R$ secretly. The multi-user agreement protocol requires that the untrusted cloud cannot learn the random seed and the scramble code, although it controls all communications between users.

#### *Random Number Exchange*

There are many well-designed group key agreement protocols [4, 15], but most of them require multiple communication rounds among participants, which could cause long delay. Utilizing the honest-but-curious cloud and secure communication channels between the cloud and users, we adapt the practical distributed group key agreement protocol proposed in [4] to achieve round optimum and efficient random number

agreement. Let $U_1, \cdots, U_n$ be a dynamic subset of all users who want to generate a common random number and our protocol is presented in Algorithm 1. With this protocol, the photographer and his/her invisible neighbors can obtain the same random number, while the cloud learns nothing about the random number.

**Algorithm 1** Random Number Agreement.

**System Initialization:**

Cloud generates and publishes system parameters: 1) a large prime number $p = \Theta(2^{cN})$, a constant $c \geq 1$, $q = \Theta(2^N)$ and $g \in Z_p$ of order $q = \Theta(2^N)$.

Each user $U_i$ generates his private parameter $a_i \in Z_q$ and public parameter $b_i = g^{a_i} \mod p$ and sends $b_i$ to the cloud.

Cloud checks that $b_i^q \equiv 1 \mod p$ for all $i = 1, \cdots, n$.

**Runtime:**

1: Cloud arranges $n$ users' indices in a cycle and sends $b_{i-1}$ and $b_{i+1}$ to each $U_i, i = 1, \cdots, n$.

2: Each $U_i, i = 1, \cdots, n$ computes $c_i$ and sends it to cloud

$$c_i = (b_{i+1}/b_{i-1})^{a_i} \mod p. \quad (3)$$

3: Cloud sends $c_1, \cdots, c_n$ to each $U_i, i = 1, \cdots, n$.

4: Each $U_i, i = 1, \cdots, n$ computes the random number

$$R_i = (b_{i-1})^{na_i} \cdot c_i^{n-1} \cdot c_{i+1}^{n-2} \cdots c_{i-2} \mod p. \quad (4)$$

Although each user generates $R_i$ individually, all $R_i$ equal to the same random number

$$R = g^{a_1a_2 + a_2a_3 + \cdots + a_na_1} \mod p. \quad (5)$$

*Scramble Code Generation*

After obtaining consistent random seed $R$, each participant generates the scramble code using Algorithm 2, and rearranges the dimension order of each feature vector according to the scramble code.

**Algorithm 2** Scramble code generation.

**Input:** Vector dimension $N$; Random number $R$; Sorted set $S = \{1, 2, \cdots, N\}$;

**Output:** Scrambled dimension sequence $\mathbf{SC}$;

1: **for** $k = N - 1; k >= 0; k - -$ **do**
2: $i = R/k!$;
3: $\mathbf{SC}[N - k] = S[i]$;
4: Remove $S[i]$ from $S$;
5: $R = R \mod k!$;
6: **end for**
7: **return** $\mathbf{SC}$;

As illustrated in Fig. 5, after scrambling feature vectors, the photographer and invisible neighbors apply LSH to scrambled vectors to get transformed vectors for current round. The cloud can simply use the transformed vectors to compute distance and conduct the same graph matching algorithm as in the basic scheme. While the membership doesn't change, the random number remains the same. In this case, the photographer can use the same random number to generate transformed vectors for new photos, and all invisible neighbors do not need any recalculation. When the membership changes, the cloud can insert/remove users into/from the existing ring of Algorithm 1 to update the random number for a new round. Note that, in this case, most users do not need to recalculate the Step 2 in Algorithm 1. Based on our evaluation, the runtime of random generation is usually only 0.014s, which is negligible for human movement. Once the random number is updated, the system achieves randomized transformation outputs for the same feature vector in different rounds.

## PROTOTYPE IMPLEMENTATION AND EVALUATION

We implement prototype systems of both variants COIN and COIN++ . To support automatic people detection, we implement the most popular face detection [25] and pedestrian detection [16] algorithm based on OpenCV. To generate portrait graph, Many segmentation methods exist in the computer vision field [1,8]. In our prototype, we adopt JSeg [8] for image segmentation, because it achieves good results with acceptable computation cost. The color threshold of JSeg is set to 100 and the merge threshold is set to 0.6, which work fine in all evaluations. Leveraging the library of MPEG-7, a 48-byte eigenfaces vector [24] is extracted as the property of a node labeled with "face", and a 64-byte color histogram vector and a 20-byte texture vector (edge histogram with 4 blocks and 5 orientations) are extracted as the property of other nodes. Eigenface is a very popular face feature vector due to its high efficiency, but it can be affected by lighting, scale and rotation. Note that, COIN is compatible with any other vector-based feature descriptors, more advanced descriptors can also be adopted. Obtaining the matching results, invisible people are removed from the photo by blurring and inpainting [5], as shown in Fig. 4. Except the image processing, all other building blocks are realized using Java, including portrait graph matching, LSH, random number agreement, vector scrambling and the messaging module. The client side is developed as an app on Android platform for case study. A user starts this app by inputting his/her portrait profile via selfieing, and chooses his/her status from "invisible me", "tag me" or "do nothing".

### Case Study and Experiment Setup

To test the practicality and efficiency of COIN the evaluation is conducted in a crowded real-life scenario: a networking workshop with more than 50 attendees in a 200 $m^2$ meeting hall. 10 volunteers (4 female and 6 male) acted as invisible users and also photographers. Within one day, the volunteers took photos freely and our system recorded the cost and photos. After the experiment, we got 208 photos. 1326 pedestrians are detected which belong to 42 individuals (7 female and 35 male), but only 412 faces are detected. The reason is that, pedestrian detection is much more robust from different view points, but face detection requires the frontal face of people. It implies that a whole body detection and description (*e.g.*, our graph model) is more robust to changes of people's pose and the camera's view angle. Fig. 6 shows some sample pedestrian images and their portrait graphs extracted by our system. In our evaluations, we do not consider those people in photo who cannot be detected by our prototype, since they are usually very small or occluded badly. Besides, the detection rate could be improved with more sophisticated people

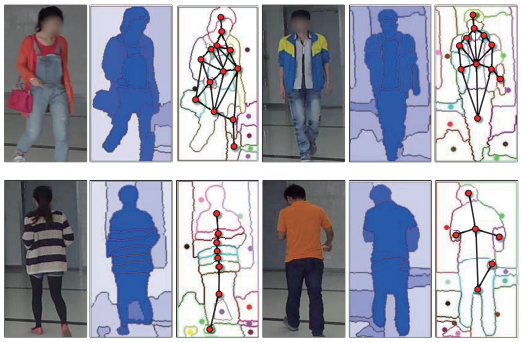

Figure 6: Sample portrait images and their portrait profile, including the original image, detected foreground and portrait profile graph. (The faces are blurred for the purpose of anonymity.)

detection algorithm, which is out of the scope of this work. For evaluation only, we manually labeled all captured people as the ground truth.

In the experiments, we use three types of phones as clients: HTC G10 (1024Hz CPU and 768M RAM), HTC G23 (1536Hz CPU and 1G RAM) and HTC New One (1741Hz CPU and 2G RAM). One laptop is used as the cloud: ThinkPad X1 with i7 2.7GHz CPU and 4GB RAM. Before experiments, we need to decide the parameter $\xi_s$ of the matching methods, which is the threshold to eliminate unmatched nodes in Stage 1. We conduct pair-wise portrait graph matchings on the dataset with different settings of $\xi_s$. When $\xi_s$ increases from 0 to 0.5, average matching time for a pair of portrait graphs is cut down by $30\%$, with little hurt to the similarity measurement. But while $\xi_s$ continues to increase, it eliminates more and more true match pairs. As a result, we set $\xi_s = 0.5$ in our experiments. We also need to determine the parameter of the LSH algorithm. The accuracy of distance measurement using LSH increases with bigger $m$ (longer hash code) and smaller $W$ (smaller window). According to the analysis in [7] and the statistics of our dataset, we set the $W$ to 3 and $m$ to 128, then the hashed vector is 128 bit. For the random number generation, $N$ is set to 512, which provides sufficient protection for the random number agreement protocol.

### Matching Accuracy

Here we investigate the most important metric, the portrait matching accuracy, which determines the correctness of invisible people removal and visible people tagging.

We start by examining the consistency and distinguishability of user's portrait graph by self-similarity (similarity between the same entity's portrait graphs) and cross-similarity (similarity between different entities' portrait graphs). In this evaluation, we remove the face property since it is highly distinctive but cannot always be obtained. Figure. 7 presents the evaluation results using the dataset. The upper blue line stands for mean self-similarity for each entity, and the lower red line is mean cross-similarity between this entity and all other entities. We notice that, generally portrait graph has a good consistency, *i.e.*, high self-similarity and small variance. And the obvious gap between self-similarity and cross-similarity shows a good distinguishability, which also shows that our graph representation is highly robust for pose changes of people and cameras. Hence, in most cases, portrait graph can provide accurate matching without face features, which also implies better privacy protection.

Then, we evaluate the matching correctness by analyzing all possible combinations using the dataset. A false negative (FN) happens when a user A is invisible, but not removed from the photo due to a match score lower than threshold $\theta_s$. A false positive (FP) happens when user A is invisible, but another visible user C is removed due to a higher match score than both the threshold and A's score. In COIN , matching is conducted on portrait graph with plain feature vectors. Fig. 8 illustrates the percentage of FN and FP changing with different threshold $\theta_s$. By selecting the threshold $\theta_s = 0.5$, COIN achieves $0.5\%$ false negative ($99.5\%$ recall) and $2.1\%$ false positive ($97.9\%$ precision) without using any face property. With face property, the false negative decreases to about $0.1\%$ and false positive is less than $1\%$. In COIN++ , feature vectors are transformed by scrambling and LSH. While the scrambling retains the accurate distance between vectors, LSH could cause some accuracy loss. Will the transformation reduce the matching accuracy? With appropriate parameters $m = 128$ and $W = 3$, COIN++ achieves comparable accuracy with COIN , as shown in Fig. 9. When $\theta_s = 0.5$, the false negative is about $0.7\%$ ($99.3\%$ recall) and the false positive is about $2.9\%$ ($97.1\%$ precision) without any face property. In summary, both variants support accurate matching and our vector transformation achieves good portrait feature privacy protection with little accuracy loss.

### Micro Benchmark

#### *Communication Cost*

In COIN , each face node takes 48B and every other node takes 84B. The size of the portrait graph depends on the node number $k$. For most applications, $k \leq 10$ is sufficient, so the communication cost for each portrait is 0.82KB. In COIN++ , after encoding, each vector is hashed to 128 bits, which reduces the size of a portrait to 0.15KB. Protocol COIN++ requires extra communication for random number agreement, which is only about 0.19KB. COIN costs each participant less than 1KB data transmission to enable portrait privacy protection. The cost for a photographer depends on the people number in the captured photo, but in most cases (with less than 10 people in photo), less than 10KB overhead is incurred, which is much less than a photo. In general, COIN achieves much smaller transmitted data size and better privacy protection than transmitting the image itself.

#### *Computation Cost*

In COIN , the computation cost is composed of portrait graph generation on the client side, and portrait matching on the cloud. COIN++ costs extra computation for random number agreement and vector transformation by scrambling and LSH. The runtime is only about 3 ms to transform ten 64-dimension feature vectors. Table. 1 presents all the decomposed computing time. It shows that, the major computation delay is caused by image processing. For a participant, it only needs

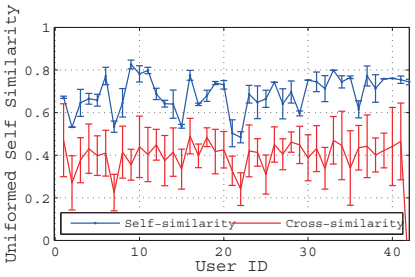


Figure 7: Portrait similarity variances.

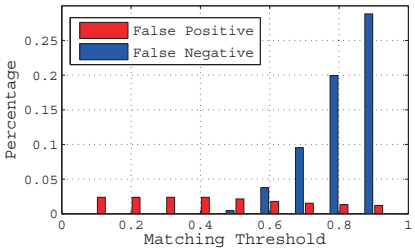


Figure 8: FP and FN in basic scheme

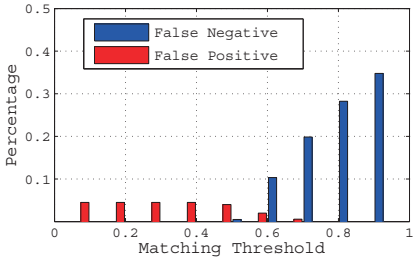


Figure 9: FP and FN in advanced scheme

to be executed once for the profile setup; for a photographer, it needs to be executed for every captured photo. The runtime of portrait detection and segmentation depends on the resolution and complexity of the photo, but the detection and segmentation results are not sensitive to scaling. Hence, in our prototype all images are scaled to about 240,000 pixels. For the photographer, on average it takes about 0.4s to conduct face and pedestrian detection. Given a portrait image/subimage, the processing time of segmentation and feature extraction increases with the image complexity, *i.e.* region number after segmentation. On average, there are 28.2 regions of each portrait, and it takes about 2.6s to process one image.

Table 1: Microbenchmarks of Runtime (in second)

| Client | | | |
|---|---|---|---|
| Segmentation | 0.5 | 2.4 | 8.1 |
| Extraction | 0.02 | 0.25 | 1.3 |
| Random-Gen | 0.012 | 0.014 | 0.017 |
| Cloud | | | |
| Matching (basic) | 0.015 | 0.037 | 0.079 |
| Matching (advanced) | 0.006 | 0.01 | 0.039 |
| Random-Init | 1.31 | 0.9 | 1.57 |

Compared with the image processing, the runtime of graph generation and matching is nearly negligible. On the client side, only extra 0.014s runtime is required for random number generation in COIN++ . On the cloud side, the time needed to match a pair of portrait graph is only about 0.04s in COIN and decreases to 0.01s in COIN++ due to the hashed feature vector. The cloud also needs 0.9s to generate system parameters for random number generation. The millisecond-level portrait graph transmission delay is negligible too. So the total computation delays for both variants are about 3s on the client and 1s on the cloud, which results a 4s system computation delay.

Now we've learned the magnitude of the time cost for each component, the overall delay also depends on the number of co-located invisible neighbors. With more active peers sending privacy requests, the matching cost will increase, but compared to the image processing cost, the matching cost on the cloud side is still quite small. Besides, the power consumption caused by our protocol (second-level computation) is much smaller than that caused by photo capturing itself.

### Case Evaluation

We conduct the case based evaluation in the forementioned experiments. When there are invisible users in the photo, the false negative rate is about $1.4\%$ ($98.6\%$ recall) and the false positive rate is $0.9\%$ ($99.1\%$ precision). But when there are no invisible users in the photo, the false positive rate raises to $4.9\%$ ($95.1\%$ precision) due to the absent of any true match users, and the threshold 0.5 is not high enough to exclude all false matches. And the average time for successful invisible people removal is about 4 seconds.

## DISCUSSION AND CONCLUSION

In this work, we present a new approach COIN to protect users' portrait privacy during photo taking and sharing. With our system, users that are unwilling to be photographed will be automatically erased from the pictures in a lightweight and privacy-preserving way. To achieve this goal, we propose the integrated system framework, a portrait graph matching scheme to match people in photos and an encryption-free privacy-preserving vector distance computation method. We have fully implemented our protocol, and thoroughly evaluated the protection performance and the overhead of the system.

Our work is a first step towards privacy-preserving photo capturing and sharing. There are still plenty of room for improvement, and also many open problems to solve. First, the accuracy and efficiency of our method is highly dependent on the performance of people detection and image segmentation methods. For example, a large crowd of people or a complicated environment may bring high error rate or computation cost. With the remarkable development of deep learning techniques in the computer vision area, more accurate and fast solutions can be adopted to facilitate our system. Besides, more sophisticated incentive mechanism should be designed to encourage people to use privacy-friendly camera apps.

## ACKNOWLEDGMENT

The research is supported in part by NSF China under Grants No. 61572281, No. 61472218, and China Postdoctoral Science Foundation under Grant No. 2015M580101, 2016T90096, No.2015M570101. The research of Li is partially supported by NSF ECCS-1247944, NSF CMMI 1436786, NSF CNS 1526638, NSF China under Grant No. 61520106007.

## REFERENCES


1. Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. 2011. Contour detection and hierarchical image segmentation. *IEEE TPAMI* 33, 5 (2011), 898–916.
2. Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE TPAMI* 19, 7 (1997), 711–720.
3. Cheng Bo, Guobin Shen, Jie Liu, Xiang-Yang Li, YongGuang Zhang, and Feng Zhao. 2014. Privacy. tag: Privacy concern expressed and respected. In *SenSys*. ACM, 163–176.
4. Mike Burmester and Yvo Desmedt. 1994. A secure and efficient conference key distribution system. In *EUROCRYPT*. 275–286.
5. Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing* 13, 9 (2004), 1200–1212.
6. Maryam Daneshi and Jiaqi Guo. 2011. Image reconstruction based on local feature descriptors. (2011).
7. Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*. ACM.
8. Yining Deng and BS Manjunath. 2001. Unsupervised segmentation of color-texture regions in images and video. *IEEE TPAMI* 23, 8 (2001), 800–810.
9. Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. 2009. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*. Springer, 235–253.
10. Nan Hu, Raif M Rustamov, and Leonidas Guibas. 2013. Graph Matching with Anchor Nodes: A Learning Approach. In *CVPR*. IEEE.
11. DK Isenor and Safwat G Zaky. 1986. Fingerprint identification using graph matching. *Pattern Recognition* 19, 2 (1986), 113–122.
12. Jonathan Katz, Amit Sahai, and Brent Waters. 2008. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*. 146–162.
13. Nikos Komodakis. 2006. Image completion using global optimization. In *CVPR*. IEEE.
14. Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.
15. Patrick PC Lee, John CS Lui, and David KY Yau. 2006. Distributed collaborative key agreement and authentication protocols for dynamic peer groups. *IEEE/ACM TON* 14, 2 (2006), 263–276.
16. Bastian Leibe, Edgar Seemann, and Bernt Schiele. 2005. Pedestrian detection in crowded scenes. In *CVPR*. IEEE.
17. Xiang-Yang Li and Taeho Jung. 2013. Search me if you can: privacy-preserving location query service. In *INFOCOM*. IEEE.
18. Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. 2012. Push the limit of wifi based localization for smartphones. In *MobiCom*. ACM.
19. Shibnath Mukherjee, Zhiyuan Chen, and Aryya Gangopadhyay. 2006. A privacy-preserving technique for Euclidean distance-based mining algorithms using Fourier-related transforms. *The VLDB Journal* 15, 4 (2006), 293–315.
20. Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. 2010. Efficient Privacy-Preserving Face Recognition. In *ICISC*.
21. Andrew Senior, Sharath Pankanti, Arun Hampapur, Lisa Brown, Ying-Li Tian, and Ahmet Ekin. 2005. Blinkering Surveillance: Enabling video privacy through computer vision. In *S&P*. IEEE.
22. Glenn Shafer. 1976. *A mathematical theory of evidence*. Vol. 1. Princeton university press Princeton.
23. Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, and Mahadev Satyanarayanan. 2013. Scalable Crowd-Sourcing of Video from Mobile Devices. In *Mobisys*. ACM.
24. Matthew Turk and Alex Pentland. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience* 3, 1 (1991), 71–86.
25. Paul Viola and Michael J Jones. 2004. Robust real-time face detection. *IJCV* 57, 2 (2004), 137–154.
26. Laurenz Wiskott, J-M Fellous, N Kuiger, and Christoph Von Der Malsburg. 1997. Face recognition by elastic bunch graph matching. *IEEE TPAMI* 19, 7 (1997), 775–779.
27. Lan Zhang, Taeho Jung, Puchun Feng, Kebin Liu, Xiang-Yang Li, and Yunhao Liu. 2015a. Pic: Enable large-scale privacy preserving content-based image search on cloud. In *ICPP*. IEEE, 949–958.
28. Lan Zhang, Taeho Jung, Cihang Liu, Xuan Ding, Xiang-Yang Li, and Yunhao Liu. 2015b. Pop: Privacy-preserving outsourced photo sharing and searching for mobile devices. In *ICDCS*. IEEE, 308–317.
29. Lan Zhang, Xiang-Yang Li, Wenchao Huang, Kebin Liu, Shuwei Zong, Xuesi Jian, Puchun Feng, Taeho Jung, and Yunhao Liu. 2014. It starts with igaze: Visual attention driven networking with smart glasses. In *MobiCom*. ACM, 91–102.
30. Lan Zhang, Xiang-Yang Li, Kebin Liu, Taeho Jung, and Yunhao Liu. 2015. Message in a sealed bottle: Privacy preserving friending in mobile social networks. *IEEE Transactions on Mobile Computing* 14, 9 (2015), 1888–1902.