P2F2: Privacy-Preserving Face Finder

Nora Almalki, Reza Curtmola, Xiaoning Ding, Narain Gehani, Cristian Borcea
Department of Computer Science, New Jersey Institute of Technology
Email: {naa34, crix, xiaoning.ding, narain.gehani, borcea}@njit.edu

***Abstract*—Fueled by the explosive growth in the number of pictures taken using smart phones, people are increasingly using cloud photo storage services. Although many innovative apps have been developed to leverage this collection of photos in the cloud, users are concerned with the privacy of their photos. We have developed Privacy-Preserving Face Finder (P2F2), a system that allows cloud-based photo matching, while preserving the privacy of the photos from the cloud provider. P2F2 stores encrypted photos in the cloud and performs photo matching based on feature vectors extracted from the photos. At its core, P2F2 relies on a novel privacy-preserving protocol for computing the Chi-square distance between the feature vectors of two photos. To achieve its goal, P2F2 extracts two privacy-preserving components from a photo's feature vector and stores them at non-colluding cloud providers. Unlike previous privacy-preserving work, P2F2 is designed to work with feature descriptors that are optimized for face recognition. An authorized querier can match a target face photo with a set of encrypted face photos stored in the cloud and receive the k most similar encrypted photos, which can then be decrypted. We have implemented a prototype of P2F2 and evaluated its performance using smart phones and a small-size cloud. Our security analysis and experimental evaluation show that P2F2 successfully achieves the desired security guarantees and is feasible in practical conditions.**

I. Introduction

Over the last decade, the number of pictures taken by people has increased six-fold to 1.2 trillion. Smart phones are the primary capture devices and 79% percent of people said they use their phones regularly to take photos [1]. Many of these photos are stored in the cloud, leveraging its virtually infinite amount of storage and its reliability and availability to offer access to the photos anytime, anywhere.

The collection of photos taken by people provide a huge and valuable database for face recognition applications that could use this data to enable a user to find a specific person or connect with an old friend. Many mobile apps have been created to leverage this collection of photos taken by users of mobile devices. For example, the Lost Child app [2] allows parents to search for a lost child at a specific place and time. The LEEDIR [3] app enables eyewitnesses to submit photos and videos during emergency events to help law enforcement and relief agencies. The FBI used crowdsourced surveillance during the investigation of the Boston marathon bombings and analyzed photographic evidence provided by spectators. Facebook uses a Tag Suggestions tool to pre-tag newly uploaded photos. This tool uses facial recognition software to match these photos with photos that have been tagged elsewhere.

Unfortunately, the privacy of these photos, many of them of personal nature, can be violated by the cloud providers, cloud users, or external malicious entities. Once data is stored in the cloud, data owners relinquish control over it and have to fully trust cloud providers with the security and privacy of their data. Some cloud service providers would claim the data they store is encrypted and private, but they still hold the encryption keys and can decrypt it at any time. Furthermore, many cloud service providers such as Google, Microsoft, and Yahoo have been handing over data to government agencies in response to their requests. For example, in the first 6 months of 2015, Google has received $35,365$ government requests for information affecting $68,908$ user accounts, and it provided the government some data in response to at least 63% of its requests [4]. Due to the sensitive information these photos contain, privacy has become a major concern and people are more reluctant to outsource the photos to the cloud.

A straightforward solution to achieve privacy would require users to use traditional encryption techniques to encrypt the photos before storing them in the cloud. However, such an approach conflicts with many innovative apps that allow users to search through and share each other's photos. Another approach is to use fully homomorphic encryption (FHE) and allow the cloud to handle encrypted photos. Although research on FHE has made significant leaps recently, it is far from being practical by several orders of magnitude.

Recently, there have been several proposals to provision cloud providers with special features that allow users to share their photos in a privacy-preserving manner [5]–[10]. The cloud provider would act as a provider of storage and computation that facilitates sharing of photos between users, without having access to the actual photos that are being shared. Most of these proposals use specialized encryption methods or use partially homomorphic encryption.

Following this line of work, we introduce Privacy-Preserving Face Finder (P2F2), a system that allows cloud-based photo matching, while preserving the privacy of the photos from the cloud provider. P2F2 allows owners to store encrypted photos and their privacy-preserving feature vectors in the cloud and, at the same time, queriers authorized by the owners can perform photo matching using these vectors. An authorized querier receives the k most similar encrypted photos to its target photo, and it can then decrypt these matching photos. In this process, the cloud provider does not get access to the unprotected photos, so the privacy of owners' photos is preserved. At the core of our approach, we rely on a novel privacy-preserving protocol for computing the Chi-square distance between the feature vectors of two photos.

P2F2 is different from prior work in two fundamental aspects. *First*, prior work on privacy-preserving photo sharing

and searching focuses on visual descriptors such as SIFT [11] or SURF [12], which work well for general purpose image matching. However, these descriptors are not optimized for face recognition. We are not aware of any privacy-preserving work that targets visual descriptors suitable for face matching. P2F2 seeks to fill this gap and is designed specifically for face matching using the Local Binary Patterns (LBP) algorithm. *Second*, unlike visual descriptors designed for general purpose image matching, which are used in conjunction with the Euclidian distance, LBP works best with the Chi-square distance. Prior privacy-preserving work cannot be immediately extended from the Euclidian distance to the Chi-square distance.

In recent years, LBP has received increasing attention for facial representation due to its good performance in various facial image applications, including face detection and recognition [13], [14], gender/age classification [15], and expressions analysis [16]. In this work, we use LBP due to its simplicity, computation efficiency, and robustness to monotonic gray-scale changes caused by illumination variations.

Feature vectors can leak information about the image. Recent research shows that the output of a blackbox feature descriptor software, such as those used for image indexing, can be used to approximately reconstruct the original image [17] [18]. Another study proved that it is possible to reconstruct image parts from their binary local descriptors without any additional information [19]. Thus, it is clear that in order to ensure the privacy of photos, one must also ensure the privacy of their feature vectors. Our solution takes this into account by extracting two components from a photo's feature vector and storing each component at non-colluding cloud providers. Each component is carefully designed not to leak information about the original feature vector and, at the same time, to allow the matching protocol to correctly compute the Chi-square distance of feature vectors in a privacy-preserving manner.

Contributions. To the best of our knowledge, we propose the first privacy-preserving distance computation protocol designed for LBP features. Our contribution is twofold:

1) We design P2F2, a system that leverages cloud service providers and allows users to perform photo matching, while preserving the privacy of the photos from the cloud provider. This protocol could be applied to any system or face recognition algorithm that uses the Chi-square distance to measure the similarity between two photos.

2) We implement a prototype of P2F2 and evaluate its performance. Our findings show that P2F2 is feasible in practical conditions. For example, a data owner can pre-process 50 images/s on a laptop, and 1.5 images/s on a smart phone. Peforming photo matching for a target image through a dataset of 165 images takes 3.6s.

The rest of the paper is organized as follows. Section II discusses related work. Section III provides background for the LBP algorithm. Section IV presents the system and adversarial model. Section V presents the details of the P2F2 system. Section VI evaluates the performance of our system. Section VII concludes the paper.

II. Related Work

Different image matching algorithms use different image visual descriptors and different distance metrics. From the point of view of designing a privacy-preserving protocol, there is no general approach that fits all such algorithms: each visual descriptor and distance metric presents its own complexity and requires a different approach. We are not aware of any prior work that computes, in a privacy-preserving manner, the Chi-square distance between feature vectors generated by LBP.

POP [5] is a framework for outsourcing photo sharing and searching to untrusted servers. At its core, POP uses a privacy-preserving protocol based on the Paillier additive homomorphic encryption to compute the Euclidean distance for the SIFT and SURF visual descriptors of images. P2F2 computes a different distance (Chi-square) for LBP features, which requires an operation that is not supported by the Paillier cryptosystem: division over ciphertexts.

PIC [6] is a cloud-based privacy-preserving image search system, which uses a multi-level homomorphic encryption to encrypt images. Such encryption is more efficient than fully homomorphic encryption, but would not be practical for mobile devices with limited resources that would need to perform encryption. In addition, this approach cannot handle the Chi-square distance that requires division over ciphertexts.

P3 [7] is a system designed for privacy-preserving Photo Sharing Services (PSPs) such as Flickr or Picasa. In P3, the sender splits an image into a public component and a secret component. The PSP server only acts as a medium to transport these two components between the sender and the receiver. The receiver reconstructs the original image from these two components. Unlike P2F2, P3 is not designed to allow the PSP to perform the actual search process, i.e., match a target photo to a set of candidate photos based on a similarity metric.

Xia et al. [9] propose a searchable encryption scheme that supports content-based image retrieval over encrypted data. The scheme uses the earth mover's distance as a similarity metric for image retrieval, which involves an expensive computation with cubic time complexity. Furthermore, the scheme requires multiple rounds of interaction between the querier and the cloud server, and it only supports a static image database.

SecSIFT [10] uses three cloud entities to outsource the privacy-preserving extraction of SIFT features from images. P2F2 focuses on a different problem (outsourcing the photo matching operations to the cloud) and relies on image owners to pre-process their images before storing them in the cloud.

III. Background on LBP

Local Binary Patterns (LBP) is a visual descriptor used in computer vision for face recognition [20]. LBP is based on the assumption that texture has locally two complementary aspects, local spatial structure (patterns) and gray scale contrast (the strength of the patterns). It is a gray-scale invariant texture measure computed from the analysis of a local neighborhood over a central pixel. As shown in Fig. 1 [21], the LBP feature vector is calculated as follows: (1) Divide the image into local blocks, e.g., 8×8 blocks. (2) For each pixel in a block,

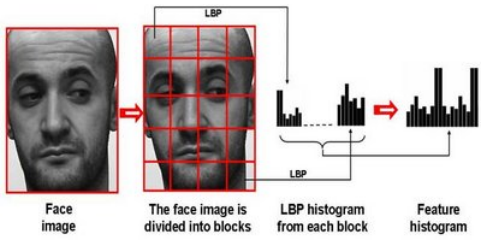


FIG. 1: Extracting the feature vector from image using LBP.

compare the pixel to each of its 8 neighbors. (3) Label each neighbor based on comparing its value to the center pixel's value: we label "1" if it is greater than the center, and "0" otherwise. These labels give an LBP code as an 8-digit binary number, which is converted to a decimal number and assigned to the center pixel. Steps 2 and 3 are repeated for each pixel in the block. (4) For each block, compute a histogram of the frequency of each number occurring in that block. This histogram is a local feature vector with 256 elements (since there can be 256 LBP codes). (5) Concatenate histograms of all blocks to get the *feature vector* for the entire image.

We use a version of LBP called *uniform patterns LBP* [22], which leverages the fact that some binary patterns occur more commonly than others. As a result, the number of elements in each histogram is reduced from 265 to 59. To take a concrete example, the image is divided into $8 \times 8 = 64$ blocks, and the uniform patterns LBP operator is applied to each block. This results in a 59-bin histogram for every block. The feature vector of the image is the histogram obtained by concatenating all block histograms into one histogram of size $64 \times 59 = 3776$.

Given two images that have feature vectors H_1 and H_2, the similarity between images is measured by calculating the Chi-square distance between their feature vectors [23]:

$$d(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i)} \quad (1)$$

The Chi-square distance was shown to perform better than two alternatives (histogram intersection and log-likelihood) [24] and, thus, it was our choice.

IV. System and Adversarial Model

A. System Model

As shown in Fig. 2, users of the P2F2 system can store their photos at a Cloud Service Provider (CSP) and can search through each other's photos while protecting the privacy of the stored photos from the CSP. There are two types of users in P2F2: *owners* and *queriers*. Each owner pre-processes her images using the LBP algorithm, generating a feature vector for each image. The owner then encrypts her images and stores both the encrypted images and the feature vectors at the CSP. Each querier has a target image and wants to retrieve other users' images that are most similar to the target image. The querier sends the feature vector of the target image to P2F2. The system then identifies the closest matching images to the target image based on the feature vectors and returns

FIG. 2: The P2F2 system.

the matching images to the querier. The similarity metric is based on the Chi-square distance (Equation 1).

The querier must have an authorization token in order to search through the images of other users. This authorization token will also allow the querier to decrypt the retrieved matching images (as P2F2 returns those images in encrypted format to the querier).

Since the feature vectors may leak information about the underlying images, they must be stored in a way that ensures their privacy. Owners extract two components from the feature vector of each image and store each component at non-colluding CSPs. Due to its construction, each component does not leak any information about the underlying feature vector.

B. Adversarial Model

Each CSP is considered to be "honest-but-curious". This means that each CSP follows the protocol specifications correctly, but may try to learn additional information about the photos (*i.e.*, break photo owners' privacy) by analyzing: (a) the data stored by photo owners, (b) the memory of the VMs run by P2F2 in the cloud, and (c) the messages exchanged between users during a search/match.

Owners upload the two components of the feature vectors at two CSPs (one component per CSP). We assume the CSPs do not collude and do not simply share with each other the data stored by owners (*i.e.*, the feature vector components). This non-collusion assumption about the CSPs is rooted in the reality that CSPs are often business competitors.

C. Guarantees

P2F2 seeks to achieve the following guarantees:

- *Match Accuracy*: The matching results returned by P2F2 are the same as in a system that performs the search over the unencrypted images.
- *Image Privacy:* The CSPs do not learn any information about the images stored by owners.

The match accuracy guarantee ensures that P2F2 preserves data privacy without sacrificing image match accuracy. The image privacy guarantee ensures confidentiality of the images stored by owners at CSPs.

We note that P2F2 does not attempt to hide the querier's target image. This matches application scenarios where law enforcement searches for a crime suspect, and the photo of the suspect does not need to be private. The CSP will then learn that the matching images are similar to the target photo, but will not learn anything about the other, non-matching images.

V. Privacy-Preserving Face Finder (P2F2)

P2F2 can be implemented at any CSP that supports virtualization and offers computing and storage services. P2F2

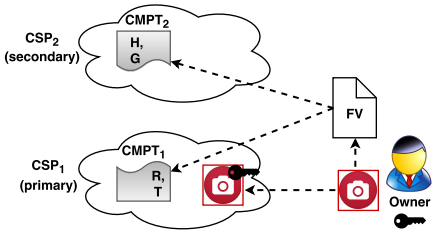


FIG. 3: The Setup protocol.

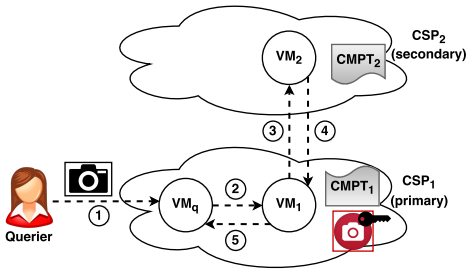


FIG. 4: The Search protocol.

consists of two phases: Setup and Search. During Setup, owners pre-process their images and store them along with their feature vectors at the CSPs. During Search, queriers authorized by photo owners search for photos that are similar to a target photo. Each owner instantiates two virtual machines, VM_1 (at CSP_1) and VM_2 (at CSP_2), and each querier instantiates a virtual machine VM_q at CSP_1. These VMs perform P2F2's functionality required in the Search phase.

A. *The* Setup *phase*

Each owner has a set of photos to be stored in the cloud. For each photo, the owner executes the Setup protocol shown in Fig. 3. The owner first pre-processes the photo and computes its feature vector FV using the *LBP* algorithm, where FV has $n = 3776$ elements: $FV = (x_1, x_2, \ldots, x_n)$. The owner then generates a prime modulus p and extracts two components from FV:

- *First Component* ($CMPT_1$): It contains two random vectors R and T whose elements are chosen at random from $[0, p-1]$ and have the same dimensions as the feature vector FV: $R = (r_1, r_2, \ldots, r_n)$ and $T = (t_1, t_2, \ldots, t_n)$.
- *Second Component* ($CMPT_2$): It contains two vectors H and G that have the same dimensions as the feature vector FV and are computed as follows:
  $H = (h_1, h_2, \ldots, h_n)$, where $h_i = (x_i^2 - r_i) \bmod p$
  $G = (g_1, g_2, \ldots, g_n)$, where $g_i = (-2x_i + r_i + t_i) \bmod p$

H and G are chosen like this such that they do not leak information about the original feature vectors and, at the same time, they allow computing the Chi-square distance between the underlying feature vectors.

The owner then encrypts the photo using a symmetric-key encryption scheme such as AES. Finally, the owner stores the encrypted image and $CMPT_1$ at CSP_1 (the primary CSP), and it stores $CMPT_2$ at CSP_2 (the secondary CSP).

B. *The* Search *phase*

In the Search phase, the querier's target photo is compared to the photos of the owners based on their feature vectors, and the k most similar encrypted photos are returned. The querier can then decrypt them if it has the encryption key from the owner. As shown in Fig. 4, the protocol that computes the Chi-square distance between the feature vector of the target photo and the feature vector of the owner's photo (defined by $CMPT_1$ and $CMPT_2$) consists of the following steps:

(Step 1) The querier sends the target photo to her virtual machine VM_q, which then computes Y the feature vector of the target photo based on the LBP algorithm, where $Y = (y_1, y_2, \ldots, y_n)$.

(Step 2) VM_q sends Y to VM_1. VM_1 generates a random vector S whose elements are chosen at random from $[0, p-1]$ and has the same dimension as the feature vector FV: $S = (s_1, s_2, \ldots, s_n)$. VM_1 then computes the vector $U = (u_1, u_2, \ldots, u_n)$, where $u_i = (r_i - r_i y_i - t_i y_i + y_i^2 + s_i) \bmod p$, and r_i and t_i are elements of the random vectors R and T computed previously by the owner as part of $CMPT_1$.

(Step 3) VM_1 then sends Y and U to VM_2. VM_2 computes the vector $V = (v_1, v_2, \ldots, v_n)$ as $V = G \times Y + H$, where "$\times$" denotes the dot product of two vectors. As a result, we have $v_i = (x_i^2 - r_i - 2x_i y_i + r_i y_i + t_i y_i) \bmod p$.

VM_2 then computes a new vector M by adding the two vectors U and V, and by dividing component-wise to Y: $M = (U + V)/Y$. Finally, VM_2 computes the masked value of the distance between the feature vectors of the two photos by adding all the elements in M: $\tilde{d} = \sum_{i=0}^{n} \frac{(x_i - y_i)^2 + s_i}{y_i}$.

(Step 4) VM_2 sends $\tilde{d}$ to VM_1, and VM_1 un-masks it to compute the Chi-square distance between the feature vectors FV and Y: $d(FV, Y) = \tilde{d} - \sum_{i=0}^{n} \frac{s_i}{y_i} = \sum_{i=0}^{n} \frac{(x_i - y_i)^2}{y_i}$.

VM_1 determines the identifiers of the top-k matching photos based on the distance between the feature vectors of the owner photos and of the target photo.

(Step 5) Finally, VM_1 sends to the querier (through VM_q) the encrypted photos that were determined in the previous step.

Authorization token: The querier needs an *authorization token* to perform searches through owners' photos. The authorization token also allows the querier to decrypt the retrieved photos that matched the search. There are several standard mechanisms that can be used to construct this authorization token. For example, the owners could encrypt the key that was used to encrypt the photos using attribute-based encryption [25], and only those queriers that possess certain specified attributes will be able to decrypt this key, and then decrypt the retrieved photos. As another example, the querier can retrieve the key needed to decrypt the photos directly from the owner.

C. *Analysis of Image Privacy and Match Accuracy*

We now show that P2F2 achieves the *image privacy and match accuracy guarantees* described in Sec. IV-B. The CSPs cannot access the content of the owners' images because these are stored encrypted with a semantically-secure encryption scheme. Thus, it remains to show that CSPs cannot learn the feature vectors of the images.

The VMs that perform the P2F2 protocol run in the untrusted CSP_1 and CSP_2. Each of these providers might try to individually learn information about the feature vectors of the images from the data stored by owners and from the data exchanged between VMs.

Analysis of data stored at CSPs: VM_1 stores $CMPT_1$ that contains two random vectors, R and T. We note that R and T are different for every image. VM_2 stores $CMPT_2$, which contains vectors H and G. These vectors mask the elements in the feature vector by adding modulo p large random numbers from the vectors R and T, where p is a large prime number chosen to have at least 80 bits. The security strength of this masking depends on the security of the pseudorandom generator used to generate the vectors R and T, and on the size of the modulus p. The corresponding components of vectors H and G, which are $h_i = (x_i^2 - r_i) \bmod p$ and $g_i = (-2x_i + r_i + t_i) \bmod p$, form a system of 2 equations with 3 unknowns, which has an infinitude of solutions.

Analysis of data exchanged between CSPs: In step 3, VM_2 receives the vectors U and Y from VM_1. Component-wise, this adds a new equation to the system of equations: $u_i = (r_i - r_i y_i - t_i y_i + y_i^2 + s_i) \bmod p$. However, this new equation also contains a new unknown (s_i), and thus the system remains underdetermined. We note that vector S is chosen at random for every search (for additional security, S can also be chosen at random by VM_1 when comparing the target image with every owner image). In step 4, VM_1 computes the distance between the target image and the owner's image. This distance is a function over all components of the owner's image, and the individual components remain protected.

It is immediate from step 4 of the Search protocol that P2F2 achieves the *match accuracy* guarantee, since the result is the exact Chi-square distance that is obtained in a system that performs the search over unencrypted data.

VI. Experimental Evaluation

We have implemented a prototype of the P2F2 system in Java. We used a large prime p of 80 bits and the BigInteger and BigDecimal classes for operating with large numbers. The random values were generated using the key derivation function proposed by Shoup [26]. The computer vision library OpenCV 3.0 was used to extract feature vectors of images.

We evaluated the performance of P2F2 in order to understand its feasibility. The evaluation focuses on: pre-processing time at the client, processing time in the cloud, and end-to-end delay to perform an image search.

We used two face image datasets for the evaluation, Yale A [27] and Yale B [28]. In each search request, we select randomly one of the images to be the query, and use the remaining images for our search dataset. The Yale A dataset contains 165 face images of 15 subjects. Each subject has 11 images in different conditions including with and without glasses, facial expression, and illumination variations. Each image is 240x240 pixels in size. The Yale B dataset contains 2,432 images of 39 subjects. Each subject has 64 images with illuminations from 64 different directions. Each image is 168x192 pixels in size.

TABLE I: Per image pre-processing time (ms).

	Laptop		**Smart phone**	
	Yale A	Yale B	Yale A	Yale B
Feature Extraction	0.66	0.36	14.26	9.83
Computing $CMPT_1$	2.32	2.45	237.28	238.83
Computing $CMPT_2$	4.67	4.53	355.31	355.74
Uploading to the Cloud	13.28	13.46	43.95	44.29
Total	20.93	20.8	650.8	648.69

To evaluate the performance of the pre-processing phase, we used a laptop and a smart phone because these are the typical personal devices storing photos. The laptop was a MacBook Pro with 2.5GHz Quad-core Intel Core i7 CPU and 16GB RAM. The smart phone was Motorola Nexus 6 XT1103 with Quad-core 2.7 GHz Krait 450 CPU and 3 GB RAM. On the cloud side, we used two servers in a small OpenStack-based cloud. Each server had Intel Xeon CPU E5-2630 CPU v3 and 80 GB memory, and the servers were connected by a 1 Gigabit network. VM_q and VM_1 were run on one of the servers and VM_2 was run on the other server. Each VM ran Linux, used 16GB memory, and was configured with 6 virtual CPUs.

Client Performance. The client is the owner of photos that need to be uploaded to the cloud. During pre-processing, the client overhead consists of the following actions for each image: extract the LBP feature vector, extract the two components $CMPT_1$ and $CMPT_2$, and upload the two components to the cloud. Table I shows the time required for each of these actions. Unlike on a laptop, the pre-processing time per image on a smart phone is not negligible, but this is a one-time operation before outsourcing the image. In addition, the pre-processing time can be reduced significantly by generating the $CMPT_1$ random values in advance and using them when needed. In our image datasets, the faces are already cropped. In general, it takes 3.6s per image to detect faces using a smart phone, and 1s using a Linux VM.

The size of the feature vector for each image is 18 KB. The size of each of the two components $CMPT_1$ and $CMPT_2$ is 106 KB. This provides insights into the time required to upload them to the cloud during pre-processing. It is also the additional storage size required at the cloud per image.

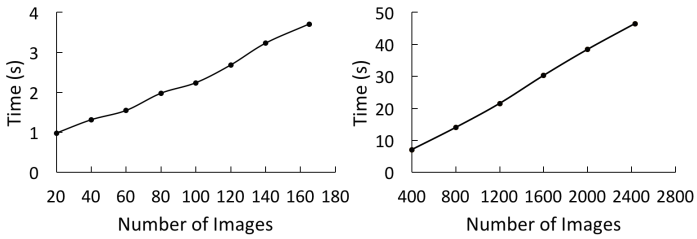
Cloud Performance. We measure the performance during a search. The communication overhead comes mainly from sending U from VM_1 to VM_2, which amounts to 53 KB for comparing to each owner image.

Regarding computation, we first measure the time taken to compute the key values required in the protocol. Table II shows that the overall time to compare a target photo to each owner image is less than 6ms. Note that the computation of U and V is done on two VMs provided by two different clouds and can be done in parallel if both VMs receive the query at the same time. However, our implementation simplifies the search procedure by sending the query to one VM.

End-to-end Delay. Figs. 5(a) and 5(b) show the end-to-end delay for a search with different number of images for the two datasets. We observe that the end-to-end delay to compare the

TABLE II: Processing time for image comparison.

Computing	Time (ms)
U	0.94
V	0.32
$\tilde{d}$	4.2
Chi-square Distance (d)	0.2



(a) Yale A dataset (b) Yale B dataset

FIG. 5: End-to-end delay for one search.

target image with each owner photo is about 20ms. Our current implementation has end-to-end delay linear with the size of the owner image dataset, but its performance can be substantially improved because the search protocol is fully parallelizable.

VII. Conclusion

We have proposed P2F2, a system that allows cloud-based photo sharing and searching in a privacy-preserving manner. P2F2 differentiates itself from previous work that tries to achieve similar privacy guarantees by accommodating feature descriptors that are optimized for face recognition.

We have built a prototype for P2F2 and evaluated its performance on a small cloud. The experimental results demonstrate P2F2's feasibility. P2F2 could be deployed today at any Cloud Service Provider (CSP) that supports virtualization and offers computing and storage services. The performance could be improved if the CSP provides natively the functionality necessary for a system like P2F2.

Acknowledgments

This research was supported by the National Science Foundation (NSF) under Grants No. CNS 1409523, CNS 1054754, DGE 1565478, and DUE 1241976, the National Security Agency (NSA) under Grant H98230-15-1-0274, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. A8650-15-C-7521. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, NSA, DARPA, and AFRL. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notice herein.

References

[1] C. Cheesman, "Report: Number of people taking photos swells eight-fold in 10 years," http://www.amateurphotographer.co.uk/.

[2] C. Borcea, X. Ding, N. Gehani, R. Curtmola, M. Khan, and H. Debnath, "Avatar: Mobile distributed computing in the cloud," in *The 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud '15)*, 2015.

[3] "Large Emergency Event Digital Information Repository: LEEDIR," http://www.leedir.com/.

[4] "Google transparency report," https://www.google.com/transparencyreport/userdatarequests/.

[5] L. Zhang, T. Jung, C. Liu, X. Ding, X.-Y. Li, and Y. Liu, "POP: Privacy-preserving outsourced photo sharing and searching for mobile devices," in *IEEE ICDCS*, 2015.

[6] L. Z. T. Jung, P. Feng, K. Liu, X.-Y. Li, and Y. Liu, "PIC: Enable large-scale privacy-preserving content-based image search on cloud," in *44th International Conference on Parallel Processing (ICPP)*, 2015.

[7] M.-R. Ra, R. Govindan, and A. Ortega, "P3: Toward privacy preserving photo sharing," in *Proc. of the NSDI*. USENIX, 2013.

[8] B. Ferreira, J. Rodrigues, J. Leitão, and H. Domingos, "Privacy-preserving content-based image retrieval in the cloud," *ArXiv e-prints*, November 2014.

[9] Z. Xia, Y. Zhu, X. Sun, Z. Qin, and K. Ren, "Towards privacy-preserving content-based image retrieval in cloud computing," *IEEE Trans. on Cloud Computing*, vol. PP, no. 99, p. 1, October 2015.

[10] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proc. of ACM International Conference on Multimedia*, 2014.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[13] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 28, no. 12, 2006.

[14] A. Hadid, M. Pietikainen, and T. Ahonen, "A discriminative feature space for detecting and recognizing faces," *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, pp. 797–804, June 2004.

[15] N. Sun, W. Zheng, C. Sun, C. Zou, and L. Zhao, "Gender classification based on boosting local binary pattern," in *Proceedings of the Third International Conference on Advances in Neural Networks*, 2006.

[16] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image and Vision Computing*, vol. 27, no. 6, pp. 803–816, May 2009.

[17] P. Weinzaepfel, H. Jgou, and P. Prez, "Reconstructing an image from its local descriptors," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 337–344.

[18] M. Daneshi and J. Guo, "Image reconstruction based on local feature descriptors," *Stanford University Technical Report*, 2011.

[19] E. d'Angelo, A. Alahi, and P. Vandergheynst, "Beyond bits: Reconstructing images from local binary descriptors," in *21st International Conference on Pattern Recognition (ICPR)*, 2012.

[20] T. Ojala, M. Pietikinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51 – 59, 1996.

[21] M. Pietikäinen, "Local Binary Patterns," *Scholarpedia*, vol. 5, no. 3, p. 9775, 2010, revision #137418.

[22] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray scale and rotation invariant texture classification with local binary patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 971 – 987, July 2002.

[23] "OpenCV: Histogram Comparison," http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html.

[24] T. Ahonen, A. Hadid, and M. Pietikäinen, *Proc. of 8th European Conference on Computer Vision (ECCV '04)*, 2004, ch. Face Recognition with Local Binary Patterns, pp. 469–481.

[25] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of the 2007 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2007, pp. 321–334.

[26] V. Shoup, "A proposal for an ISO standard for public key encryption (v. 2.1)," IBM Zurich Research Lab Technical Report, December 2001.

[27] "Yale face database," http://vision.ucsd.edu/content/yale-face-database.

[28] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643 – 660, August 2001.