Oblivious Image Matching

Shai Avidan[1], Ariel Elbaz[2], Tal Malkin[2], Ryan Moriarty[3]

[1] Adobe Research
[2] Columbia University
[3] UCLA

Abstract. We present the problem of *Oblivious Image Matching*, where two parties want to determine whether they have images of the same object or scene, without revealing any additional information. While image matching has attracted a great deal of attention in the computer vision community, it was never treated in a cryptographic sense.
In this paper we study the private version of the problem, oblivious image matching, and provide an efficient protocol for it. In doing so, we design a novel image matching algorithm, and a few private protocols that may be of independent interest. Specifically, we first show how to reduce the image matching problem to a two-level version of the fuzzy set matching problem, and then present a novel protocol to privately compute this (and several other) matching problems.

1 Introduction

The number of cameras today, in the modern world, surpass the size of its population. This, and the ease with which one can take pictures has led to an explosive growth in the number of digital pictures. Matching images, for the purpose of retrieval, analysis and visualization has therefore become an important topic in the computer vision community. This research is becoming increasingly more relevant with the popularity of photo sharing sites such as Flickr or PhotoBucket. These sites, and others, allow users to share images, but there are cases in which legal, copyright or privacy concerns might prevent users from sharing their images with others. Hence the need for a cryptographic treatment of the problem.

In this paper we address the problem of *Oblivious Image Matching*, where two parties, Alice and Bob, want to determine whether they have images of the same object or scene, without revealing any additional information. We first propose a new image matching protocol, which, while a bit less efficient than other image matching protocols in the vision community, we then show lends itself to an efficient private version.

The leading approach in the computer vision community to image matching effectively reduce images to a set of feature vectors, where each feature is considered as a point in some high dimensional vector space. Matching two images

now becomes the problem of matching two sets of feature vectors. We propose a new variant, that seeks to represent each image as a set of feature *strings* over a small alphabet. We tested this variant using standard computer vision techniques (see section 4), and showed that its accuracy is reasonably good.

When Alice and Bob are trying to privately match their images using this approach, their goal is to count the elements in one set that are *close* to the other set. This is a variant of the *fuzzy set matching* problem, which was defined in [FNP04]. There, Alice and Bob each have an input set of elements consisting of T attributes each. They want to compute the fuzzy intersection, consisting of all elements in one of the sets that match an element in the other set in at least a threshold t of attributes. The image matching variant requires a second level of matching (e.g, if we say that "Two words are similar if they match in at least t letters", and "Two sentences are similar if at least T of their words are similar", then Alice and Bob should be able to securely count the number of similar sentences they hold, without revealing any other information). This analogy is not accurate (e.g., for image matching the first level requires the attributes to be ordered, while the second level treats them as sets), but suffices for now (we give formal definitions in Section 3). This can further be extended to a multi-level cascade of matching problems.

We devise private and efficient protocols for oblivious image matching in the semi-honest model, by providing protocols for the matching problem described above (along the way providing private protocols for other variants and subtasks). Our image-matching protocol was specifically designed to correspond closely with fuzzy matching (so as it is amenable to an efficient private version), and our private fuzzy matching protocols were specifically designed with the image matching application in mind, achieving good performance for the typical parameters in that scenario.

Previous Work

There has been very little in the way of combining computer vision and cryptography. In fact, the only work that we are aware of, in the computer vision community to address this issue, is that of [AB06]. However, their work is a direct application of generic secure computation techniques [Yao82] to a standard computer vision algorithm and in that sense it falls short of providing new cryptographic tools. Below we briefly summarize the most relevant works in the vision community (without cryptography) and in the cryptography community (without vision).

Computer Vision. The problem of image matching and visual object recognition in general received a great deal of attention in recent years. Simply put, the goal is to determine if two given images are of the same object, or scene, or not. This problem is extremely difficult because change in the camera viewpoint,

illumination condition or pose of the object that is being photographed, result in different images. Simply matching images by measuring the difference in the pixel intensity values does not lead to satisfactory results. Recent advances in the field seek to break the image into a set of interest points, each of which is described with a short feature vector. The goal now becomes one of robustly matching two sets of features to determine if their corresponding images match. The quality of the various visual object recognition algorithms is measured on standard image sets, that were collected by the computer vision community. The interested reader is referred to [EZWea06,FFFP04,FPZ03,LLS04,OFPA04] for further details.

Cryptography In principle, all the problem mentioned above can be solved securely using generic solutions of secure computation for any circuit. However, these solutions are typically not efficient enough for practical use. There have been many works on efficient secure computation of various functions, starting with Lindell and Pinkas' *privacy preserving datamining* [LP00], and including many others . The most relevant to us are those of [FNP04] and [IW06]. These papers (among other contributions) introduced and solved (respectively) the fuzzy matching problem described above. Our protocol for the same problem (which we develop as a step toward our main protocol) achieves different bounds, which, depending on the setting of the parameters, may be better for some applications (e.g., the image matching application of our paper).

2 Preliminaries

2.1 Public Key Homomorphic Encryption Schemes

We use homomorphic public key encryption schemes, which are semantically secure. A homomorphic encryption allows to compute addition and multiplication by a constant using encryptions alone. We also use the fact that these operations take place in a finite field and thus are modulo the size of the field. This is achieved, among others, by Pailliers cryptosystem [Pai99], El Gamals's cryptosystem [ElG85]. Note that given homomorphic encryption, one can compute the encryption of a polynomial P at a point y, given the encrypted coefficients $a_o, \ldots, a_k$ and plaintext y.

2.2 Private Information Retrieval

Private Information Retrieval (introduced by Chor et al. [CGKS95] comes to protect a client's privacy when querying a database. We model the client's secret input as an index i into an n-bit string B held by the server, and want a communication-efficient protocol that enables the client to learn B_i without the server learning i. A trivial and *not* efficient solution is to have the server send

all of B to the client. Perhaps surprisingly, there are efficient PIR protocols that require communicating only $polylog(n)$ bits (where the index i is represented by $\log n$ bits).

The first PIR protocol that achieved $polylog(n)$ communication complexity (using two rounds) is that of Cachin, Micali and Stadler [CMS99]. In this protocol, the client runs in time $k \log n$ (where k is a security parameter), and the server runs in time $O(nk^c)$ for some c. (note, the server always "looks" at all his input bits; this must be the case, or else the server can tell which bits were surely not required by the client).

3 Secure Fuzzy Matching

Let $\mathcal{X} = \{\overline{X}_1, \ldots, \overline{X}_n\}$ and $\mathcal{Y} = \{\overline{Y}_1, \ldots, \overline{Y}_m\}$ be sets[4] of d-dimensional vectors over alphabet Σ;

1. We say that there is a *fuzzy match* between two vectors $\overline{X}_i, \overline{Y}_j$ if at least t of their coordinates are equal; that is
$$\overline{X}_i \sim_t \overline{Y}_j \quad \text{iff} \quad \#\{(\overline{X}_i)_k = (\overline{Y}_j)_k\} \geq t$$
2. We say that the vector $\overline{X}_i$ is *matched* in the set $\mathcal{Y}$ if there exists some $\overline{Y}_j \in \mathcal{Y}$ such that $\overline{X}_i$ fuzzy matches $\mathcal{Y}_j$; [5] that is
$$\overline{X}_i \sim_t \mathcal{Y} \quad \text{if} \quad \exists \overline{Y}_j \in \mathcal{Y} \quad \text{s.t.} \quad \overline{X}_i \sim_t \overline{Y}_j$$
3. We say that the sets $\mathcal{X}, \mathcal{Y}$ are *T-similar* if there are at least T vectors $\overline{X}_i \in \mathcal{X}$ that fuzzy-match the set $\mathcal{Y}$; that is
$$\mathcal{X} \approx_{t,T} \mathcal{Y} \quad \text{if} \quad \#\{i | \overline{X}_i \sim_t \mathcal{Y}\} \geq T$$

Note that a fuzzy match $\overline{X}_i, \overline{Y}_j$ matches between strings (where the order of each component matters), while at the outer level, a *T-similar* match $\mathcal{X} \approx_{t,T} \mathcal{Y}$ is between sets (where the order does not matter).

[FNP04] defined fuzzy-matching as the problem where a party that holds $\mathcal{X}$ learns all the elements from $\mathcal{Y}$ that are t-fuzzy matching to any element in X. We are more interested in determining securely if $\mathcal{X}$ and $\mathcal{Y}$ are T-similar, and in section 4 we show how to reduce the image matching problem to solving T-similarity.

In addition, we also give a secure protocol for determining fuzzy-matching over multiple levels of threshold checks (see section 5).

[4] With some modifications, one can allow repeating elements.

[5] An alternative definition requires that there is *exactly* one $\overline{Y}_j$ such that $\overline{X}_i \sim_t \overline{Y}_j$.

4 (Plain) Image Matching Protocol

In this section we will describe our (non-secure) image matching protocol. While this section does not involve any cryptography, the image matching algorithm should not be thought of as work independent of cryptography. Indeed, it was specifically designed with the goal of allowing private and efficient computation of this algorithm by two parties, without paying a big price in the accuracy of the image matching.

We start by describing the algorithm in the vision context. The reader who is only interested in the cryptographic task itself, may skip to Section 4.1, where we describe the final reduced version of the image matching algorithm, namely the fuzzy-matching flavored formulation. The following sections will focus on developing the private version for this algorithm.

Background. A gray scale image is a rectangular array of pixels where each pixel holds an integer value in the range $[0, 255]$. Matching two images by means of sum-of-square-differences of their pixel values does not lead to satisfactory results. For example, consider a check board image that is shifted one pixel to the right. Simply matching the sum-of-square-differences between the image and its shifted version will lead to a large number, indicating that the two images are not similar, while perceptually, we view these two images to be virtually identical. Hence, a different approach is needed.

The leading approach to image matching is the "bag-of-words" approach that represents an image as a collection of feature points. Specifically, dedicated computer vision algorithms detect interest points in the image (such as well defined corners in the image), extract a small patch (i.e., rectangle) of pixels around each such interest point and compute some statistics of this patch. This approach, formally known as probabilistic Latent Semantic Analysis (pLSA) [Hof99] and its hierarchical Bayesian form, Latent Dirichlet Allocation (LDA) [BNJ03] was imported, with great success [SZ03,NS06], from the textual analysis community. We will also follow the bag-of-words approach, but will depart from the standard paradigm as follows.

Our Approach. The typical (non-secure) computer vision algorithm treats the extracted features as vectors in some high dimensional vector space, and matches two images by matching their corresponding feature sets. Because vision is an inverse problem, an exact match can not usually be achieved, and one needs to use some distance metric to measure how similar two images are. The standard distance metric used in computer vision is the Euclidean distance, but we have focused on a less standard approach, of using Hamming distance and fuzzy matching. This gives a vision algorithm that is easier to turn into a cryptographically secure protocol. We have devised an image matching algorithm that relies on Hamming distance on strings, rather than Euclidean norm on vectors. In the following we refer to an array of features as vector, if we wish to emphasize that

the metric used is Euclidean, and we refer to other arrays as strings, when we wish to emphasize that we use Hamming distance as a metric.

The vision algorithm requires a pre-processing stage, which is done once before any image matching is required, and gives a set of parameters to be used in subsequent image matching instances. We describe the pre-processing stage below.

First, we collected a large number of images, detected interest points in them, extracted the patches and described each patch with the SIFT feature descriptor [Low04]. A SIFT descriptor consists of a $128D$ feature vector that corresponds to sixteen 8-bin histograms in small regions around the feature point[6]. After collecting a large sample of feature points, we used a clustering algorithm (k-means) to cluster each of the 16 histograms of the SIFT feature vector into a small number of codewords, and enumerated this small number of codewords. This gives a small alphabet (that is coordinate dependent) that will later allow us to reduce the alphabet size for feature points considerably. This is one of the main contributions in the vision algorithm we develop. We assume that this process is done only once and that the computed codewords are made available to all parties.

In the online stage, when a new image arrives, the relevant party converts it to a set of SIFT descriptors and then assign each element, in each SIFT feature vector, to its closest codeword that was found in the pre-processing stage using clustering. This way we have converted a SIFT feature *vector* into a SIFT feature *string* over some position dependent small alphabet. From now on, we will treat images as sets of strings. As our objective is to match an image held by Alice to an image held by Bob, the images can be converted into the SIFT feature strings before we begin our protocol. Then from the perspective of the protocol we are simply matching a SIFT feature string to a SIFT feature string and not two images.

Given the notations of subsection 3 we typically have about $n = m = 1000$ SIFT feature strings per image, where each SIFT feature string is of length $d = 16$ (because we have sixteen 8-bin histograms), and the number of codewords, per character in the SIFT string, (and hence the size of the alphabet Σ) is set to $|\Sigma| = 32$.

4.1 Presenting Image Matching as a Fuzzy Matching Problem

We will now describe the algorithm we use to match two sets of SIFT feature strings. This algorithm can be used to define our goal for the next sections, which is to compute this algorithm privately, when the sets are held by Alice and Bob respectively.

[6] These parameters were empirically evaluated in the computer vision community and found to give the best results on large collections of images.

Suppose we have two sets of SIFT feature strings we wish to match, $\mathcal{X} = \{\overline{X}_1, \ldots, \overline{X}_n\}$, where each $\overline{X}_i$ is comprised of the attributes $\overline{X}_{i1}, \ldots, \overline{X}_{id}$ and $\mathcal{Y} = \{\overline{Y}_1, \ldots, \overline{Y}_m\}$ where each $\overline{Y}_i$ is comprised of the attributes $\overline{Y}_{i1}, \ldots, \overline{Y}_{id}$. We give the SIFT feature string matching algorithm below in a way that makes it easy to understand the transformation of this algorithm to our secure oblivious matching protocol.

---

1. For each i, j, we check if the feature points $\overline{X}_i$ and $\overline{Y}_j$ have a t-fuzzy match.
2. For each i, we check if the feature point $\overline{X}_i$ had at least one fuzzy match.
3. We count the number of such feature points that have at least one fuzzy match, and if this number is greater than T, we declare the images to match.

Indeed, the image matching algorithm now boils down to determining whether the sets of Alice and Bob are T-similar. This can be viewed as computing three threshold matching schemes. First we find if the number of fuzzy matches between a pair of attributes is above a threshold t. Next we check if the number of fuzzy matches is above the threshold 1. And finally we find if the number of attributes to have at least one fuzzy match is greater than a threshold T.

As the protocol is essentially based on threshold comparisons, our goal is to find a way to compute these threshold comparisons such that inputs and outputs of the comparison functions are valid, and we want to do this while leaking no information. We accomplish this goal over the next sections.

5 Some Simpler Base Protocols

We start by defining two simple subset-selection functionalities and secure protocols for them. We then use these to get a multi-level threshold check functionality, and implement it securely. This gives as an easy corollary a secure protocol for two-level fuzzy-matching. In the next section, we use this two-level fuzzy matching protocol for image matching.

The reader may note that some of the techniques we use here are similar to those used by [FNP04]. As mentioned in the introduction, we extend their work to securely solve more general fuzzy-matching problems.

In what follows we assume the parties use an homomorphic encryption (see, eg, [ElG85,Pai99]) and that both parties publish their public keys.

5.1 Subset Selection with Encrypted Output

The Subset Selection with Encrypted output (SSE) functionality is as follows: Both parties share a finite field $\{0, \ldots, p-1\}$ and a *good subset* G of it. The

server's private input is some value v, encrypted under the client's public key. The functionality gives the client $ENC_S(1)$ if $v \in G$, and $ENC_S(0)$ otherwise. The following subprotocol securely implements this functionality.

---

1. The server selects a number $r \in \{0, \ldots, p-1\}$ uniformly at random, sets $v' = v + r$, and sends $Enc_C(v')$ to the Client.
2. The client decrypts it, gets v'.
3. The server creates p cyphertexts $C_0, \ldots, C_{p-1}$, such that $C_i = Enc_S(1)$ if $i - r \in R$, and $C_i = Enc_S(0)$ otherwise.
4. The client and the server run a PIR protocol, such that the client gets the v'-th cyphertext.

Note that if $v \in G \iff (v' - r) \in G \iff C_{v'} = Enc_S(1)$. Thus, the protocol is clearly correct.

We show security following the simulation paradigm, and claim the protocl if whatever a party can learn from the protocol, she could have learned just by looking at her input and output. Thus, we show a simulator that, given the party's input and output, creates a transcript that is undistinguishable from the party's view of the protocol.

The server's simulator selects v' at random and sends $Enc_C(v')$ to the client. Then it creates p cyphertexts, and runs the simulator of the PIR protocol, outputing whatever the PIR simulator outputs (which we assume by induction to be undistinguishable from the server's view in the real protocol, since we use a secure PIR protocol). It is easy to see this output of the simulator indistinguishable from the server's view during the protocol. The client's simulator, given the client's output from the protocol, $Enc_S(b)$, outputs a random number v', then it creates p cyphertexts using the server's public key. The client's simulator then replaces the v'-th cyphertext with its input, $Enc_S(b)$, and writes these p cyphertexts as a message coming from the server.

5.2 Subset Selection with Plain output

The Subset Selection with Plain output (SSP) functionality is as follows: Both parties share a finite field $\{0, \ldots, p-1\}$ and a *good subset* G of it. The server's private input is some value v, encrypted under the client's public key. The functionality lets the client know if $v \in G$.

The protocol will simply be an extension of the SSE protocol presented in the previous section. Recall at the end of the SSE protocol the client has $Enc_S(1)$ if $v \in G$, and $Enc_S(0)$ otherwise. To learn the value of this encryption without revealing the output to the server the client and server will run an extra two round protocol. Let $Enc_S(b)$ be the output of the client from the SSE protocol.

---

1. The Client server selects a number $r \in \{0, \ldots, p-1\}$ uniformly at random, sets $b' = b + r$, and sends $Enc_C(b')$ to the Server.
2. The Server decrypts b' and sends the value back to the client.
3. The Client calculates $b = b' - r$ and outputs "$v \in G$" if $b = 1$ and outputs "$v \notin G$" otherwise.

The simulator is strait forward and has been omitted.

5.3 Multi-Level Threshold Checks

In this subsection we give a secure protocol for solving the following functionality, which may be of independent interest: Given a known depth-k, tree-like hierarchy of threshold checks (where each threshold check is satisfied iff at least some threshold of its children "nodes" is satisfied), and the server's input is a set of boolean values (encrypted by the client's public key) in the leaves, the client learns if the root is satisfied or not.

A specific case is where all threshold checks at level i have the same number of variables v_i and the same threshold t_i. Let n_1 be the number of variables in the first level (thus, the number of variables in the second level is $n_2 = n_1/v_2$, and so on). The functionality thus checks, for each group of v_2 variables of the first level, if at least t_2 of them are satisfied, and marks their parent (at level 2) as satisfied if so. This repeats for levels $i = 2, 3, \ldots, k-1$ until the root's value is decided.

We assume, without loss of generality, that k is odd (otherwise, we add level $k+1$ with $v_{k+1} = 1$ and $t_{k+1} = 1$).

We give a secure protocol for this functionality.

---

1. For each node j at level 2 there are v_2 "children" variables. The server sums the v_2 values (encrypted with the client's public key) of these variables, and gets S_j^2.
2. For each j, The client and the server run the SSE protocol, where the server's private input is S_j^2, and the *good set* G is $\{t_2, \ldots v_2\}$.
3. The client now has n_2 variables for the second level (encrypted with the server's public key).
4. The client and the server repeat the previous steps, except now switching roles, until level $k-1$.
5. At the last level, the parties run the SSP protocol (and thus the client gets the result of the root threshold check unencrypted).

5.4 A Fuzzy Matching Protocol

In this section, we assume the client's input is a set of n d-dimensional vectors and the server's input is a set of m d-dimensional vectors, and both share a

threshold t. The Two-Level Fuzzy Matching functionality gives to the client the indices (i, j), where $i \in [n], j \in [m]$ of every pair of vectors that are t-similar.

This functionality is a variant of the secure fuzzy matching that was presented at [FNP04], and differs in that the client does not learn the elements that fuzzy-match his elements, but rather he learns the pairs of indices. This problem was left as an open problem in [FNP04], and was later solved by [IW06], who get communication complexity of $O(nd^2 + n^2)$ (they have $n = m$). The protocol we give below has communication complexity of $O(nd|\Sigma| + nm\log^c(d))$, which may be better for certain settings of parameters.

As in previous sections, denote the client's input $\mathcal{X} = \left\{\overline{X}_1, \ldots, \overline{X}_n\right\}$, and the server;s input by $\mathcal{Y} = \left\{\overline{Y}_1, \ldots, \overline{Y}_m\right\}$, where each $\overline{X}_i, \overline{Y}_j \in \Sigma^d$ for some finite alphabet Σ.

---

1. For each $i = 1, \ldots, m$ and $k = 1, \ldots, d$, the client encrypts $\ell_{ik}^{\overline{X}_{ik}}$ as one, and any other ℓ_{ik}^{σ} (for $\sigma \in \Sigma$) as zero.
2. The Client sends all $nd|\Sigma|$ encryptions to the server, who picks the $md|\Sigma|$ relevant encryptions of $\ell_{ik}^{\overline{Y}_{jk}}$ to work with.
3. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, the server sums to get $S_{ij} = \sum_{k=1}^{d} \ell_{ik}^{\overline{Y}_{jk}}$.
4. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, the client and the server run SSP protocol, where the server's input is S_{ij} (which is encrypted using the client's public key), and the *good set* G is $\{t, \ldots, d\}$.
5. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, the result of the SSP protocol is the client's output; (that is, if the SSP outputs "YES" for (i, j), the client learns that $\overline{X}_i$ and $\overline{Y}_j$ are matching).

The communication complexity of this protocol is $O(nd|\Sigma| + nm\log^c(d))$ where c is a parameter of the PIR protocol (see section 2.2).

Note that we use two-levels fuzzy matching, rather than multi-level, since this algorithm will be used in the next section.

We omit a length security discussion, as this protocol is very similar to the protocol in section 6.

6 Oblivious Image Matching Protocol

This is our main protocol for image matching, and is done on the set of "strings" that both Alice and Bob reduce their images to (see section 4). We use the same notation for the parties' private inputs as in previous sections.

The protocol proceeds as follows.

---

1. For each $i = 1, \ldots, m$ and $k = 1, \ldots, d$, Alice encrypts $\ell_{ik}^{\overline{X}_{ik}}$ as one, and any other ℓ_{ik}^{σ} (for $\sigma \in \Sigma$) as zero.
2. Alice sends all $nd|\Sigma|$ encryptions to Bob, who picks the $md|\Sigma|$ relevant encryptions of $\ell_{ik}^{\overline{Y}_{jk}}$ to work with.
3. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, Bob sums to get $S_{ij} = \sum_{k=1}^{d} \ell_{ik}^{\overline{Y}_{jk}}$.
4. For each $i = 1, \ldots, n$ and $j = 1, \ldots, m$ Alice and Bob perform the SSE protocol (Alice plays the client, Bob plays the server). For application (i, j) of the protocol, Bob's input is S_{ij} and the *good subset* G is $\{t, t+1, \ldots, d\} \subset [0..p]$. At the end of this step, Alice holds $\hat{S}_{ij}$, which are encryptions (by Bob's public key) of zero or one. (Note that $Dec_B(\hat{S}_{ij}) = 1$ if and only if $S_{ij} \in [t..d]$, that is if and only if X_i and Y_j fuzzy match.)
5. For each $i = 1, \ldots, n$, Alice computes $Z_i = \sum_{j=1}^{m} \hat{S}_{i,j}$.
6. For each $i = 1, \ldots, n$, Alice and Bob perform the SSE protocol (but now Alice plays the server and Bob plays the client). For application i of the protocol, Alice's input is Z_i and the *good subset* G is $\{1, \ldots, m\} \subset [0..p]$. At the end of this step, Bob holds $\hat{Z}_i$, which are encryptions (by Alice's public key) of zero or one. (Note that $Dec_A(\hat{Z}_i) = 1$ if and only if $Z_i \in [1..m]$, that is if and only if $\overline{X}_i$ had at least one fuzzy match).
7. Bob computes $W = \sum_{i=1}^{n} \hat{Z}_i$. Now Alice and Bob run the SSP protocol (Alice plays the client and Bob plays the server). Bob's input is W and the *good subset* G is $\{T, \ldots, n\} \subset [0..p]$.
8. If the output of the SSP is "YES", Alice concludes that $\mathcal{X}$ is T-similar to $\mathcal{Y}$. Otherwise, she concludes that $\mathcal{X}$ is not T-similar to $\mathcal{Y}$.

6.1 Complexity

The communication complexity of step 2 is $O(nd|\Sigma|)$. Using PIR, step 4 contributes $O(nm \log^c |d|)$, step 6 contributes $O(n \log m)$, and step 7 at most n. Therefore, the total communication complexity is $O(nd|\Sigma| + nm \log^c(d))$.

For the case image matching, we usually have $n \approx m \approx 1000$, and our choice of parameters for the experimental part was $d = 16$, $|\Sigma| = 32$.

Security We show security using simulators for Alice and for Bob.

A simulator SIM_A for Alice is given Alice's input and output and creates transcript that is undistinguishable from her view during the protocol.

1. SIM_A simulate Alice's step.
2. SIM_A generates $nd|\Sigma|$ encryptions (with Alice's public key) of zero and sends them to Bob.
3. (nothing to do)
4. For each of the interactions, Alice has no input for the Select-By-Range protocol, and gets as output an encryption of zero or one. SIM_A prepares

random output $\hat{S}_{ij} = Enc_B(0)$ (that is, a random encryption of zero, using Bob's public key) and then runs the simulator for Select-By-Range protocol, supplying $\hat{S}_{ij}$ as Alice's output from the protocol.
5. For each $i = 1, \ldots, n$, SIM_A computes $Z_i = \sum_{j=1}^{m} \hat{S}_{i,j}$.
6. For each $i = 1, \ldots, n$, SIM_A runs the simulator of Select-By-Range (where Alice plays the server's part now), supplying it with Alice's input to the protocol which is Z_i, and the range $R = \{1, \ldots, m\}$.
7. SIM_A selects $n - T + 1$ random values, $v_T, \ldots, v_n$. If Alice's output from the protocol is that there is a match, choose a random index i between T and n, and replace v_i with 0. SIM_A then encrypt these $n - T + 1$ values, using Alice's public key, and simulates Bob sending them.
8. SIM_A simulates Alice on the inputs received.

A simulator SIM_B for Bob is given Bob's input only, since Bob does not get any output from the protocol.

1. (nothing to do).
2. SIM_B creates $nd|\Sigma|$ encryptions of zero (using Alice's public key).
3. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, SIM_B sums to get S_{ij} as the sum of the appropriate encryptions from the previous step.
4. For each $i = 1, \ldots, n$ and $j = 1, \ldots, m$, SIM_B runs the Select-By-Range simulator. Bob's input for the protocol is S_{ij}, and the range is $R = \{t, t + 1, \ldots, d\}$. Bob does not get an output from this protocol.
5. (nothing to do).
6. For each $i = 1, \ldots, n$, SIM_B creates $\hat{Z}_i$ which is an encryption, using Alice's public key, of zero. SIM_B then provides $\hat{Z}_i$ as Bob's output to a simulator of the Select-By-Range protocol.
7. (a) SIM_B computes $W = \sum_{i=1}^{n} \hat{Z}_i$.
   (b) For each $h = T, \ldots, n$, SIM_B selects u_h uniformly at random, and computes $V_h = Enc_A((\hat{W} - h) \cdot u_h)$, where $\hat{W} = Dec_B(W)$.
   (c) Bob permutes these values and sends them to Alice.

7 Conclusions

We presented the problem of *Oblivious Image Matching*, where two parties want to determine whether they have images of the same object or scene, without revealing any additional information. This problem lies at the intersection of cryptography and computer vision and, as a result, received little or no treatment, so far.

We provide an efficient protocol for it and, in doing so, we design a number of private protocols that may be of independent interest. Specifically, we have shown how to reduce the image matching problem to a two-level version of the fuzzy set matching problem, and then presented a novel protocol to privately compute this (and several other) matching problems.

References

[AB06] S. Avidan and M. Butman. Blind vision. In *European Conference on Computer Vision*, pages 1–13, 2006.

[BNJ03] D. Blei, A. Ng, and M. Jordan. Latent dirchilet allocation. volume 3, pages 993–1022, 2003.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. *Lecture Notes in Computer Science*, 1592:402–414, 1999.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology — (CRYPTO 1984)*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[EZWea06] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool et al. The 2005 pascal visual object classes challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, pages 117–176, 2006.

[FFFP04] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples. In *Workshop on Generative-Model Based Vision, IEEE Proc. CVPR*, 2004.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology — (EUROCRYPT 2004)*, volume 3027, pages 1–19. Springer-Verlag, 2004.

[FPZ03] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, Madison, Wisconsin, June 2003.

[Hof99] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[IW06] P. Indyk and D. Woodruff. Polylogarithmic private approximations and efficient matching. In *Theory of Cryptography Conference (TCC)*, pages 245–264, 2006. Full version available at http://eccc.hpi-web.de/eccc-reports/2005/TR05-117/index.html.

[LLS04] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, (2):91–110, 2004.

[LP00] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology — (CRYPTO 2000)*, 2000.

[MS04] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. volume 60, pages 63–86, 2004.

[NS06] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[OFPA04] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Generic object recognition with boosting. Technical Report TR-EMT-2004-01, EMT, TU Graz, Austria, 2004. Submitted to the IEEE Transactions on Pattern Analysis and Machine Intelligence.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — (EUROCRYPT 1999)*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer-Verlag, 1999.

[SZ03] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, pages 1470–1477, 2003.

[Yao82] A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 160–164, Chicago, 1982. IEEE.

A Experimental Results

We implemented a non-secure version of our Oblivious Image Matching algorithm to see how well it works on real images.

For the pre-processing stage, described in subsection 4, we have collected approximately $100,000$ interest points, from the Caltech 100 image data set [FFFP04], that consists of 100 random images downloaded from the web. We use the Harris-affine detector [MS04] to detect them and the SIFT descriptor [Low04] to describe them. Recall that a SIFT descriptor consists of a $128D$ feature vector that corresponds to sixteen 8-bin histograms in small regions around the interest point. Then, we clustered each of the sixteen histograms independently into 32 clusters. This gives a position-dependent alphabet of size 32. Every $128D$ vector is mapped to a 16 characters long string. In the following, we assume that both parties have access to the clusters and so each party can transform his/her SIFT descriptor to a 16 character long string.

We first conducted a set of experiments that measure the stability of our algorithm. Stability is measured by taking multiple pictures of the same scene, while varying one camera parameter, say the camera zoom. In particular we measured the stability of our algorithm to changes in camera zoom, image rotation, image blur, JPEG compression, illumination changes and viewpoint changes. We found that our algorithm can robustly match, on average, a few hundred SIFT features, as opposed to about 1000 SIFT feature matches that non-secure vision algorithms can do. This means that our algorithm has enough samples to perform image matching, albeit in a less robust manner than standard vision algorithms.

To verify the assumption that indeed we can match enough SIFT features to allow for image matching, we conducted an experiment on a set of 9 pairs of images of different objects, taken from the CALTECH gadgets dataset. We split each pair, giving one image to Alice and the other to Bob, then we ran the algorithm and computed the confusion matrix. See Figure 1. The figure shows

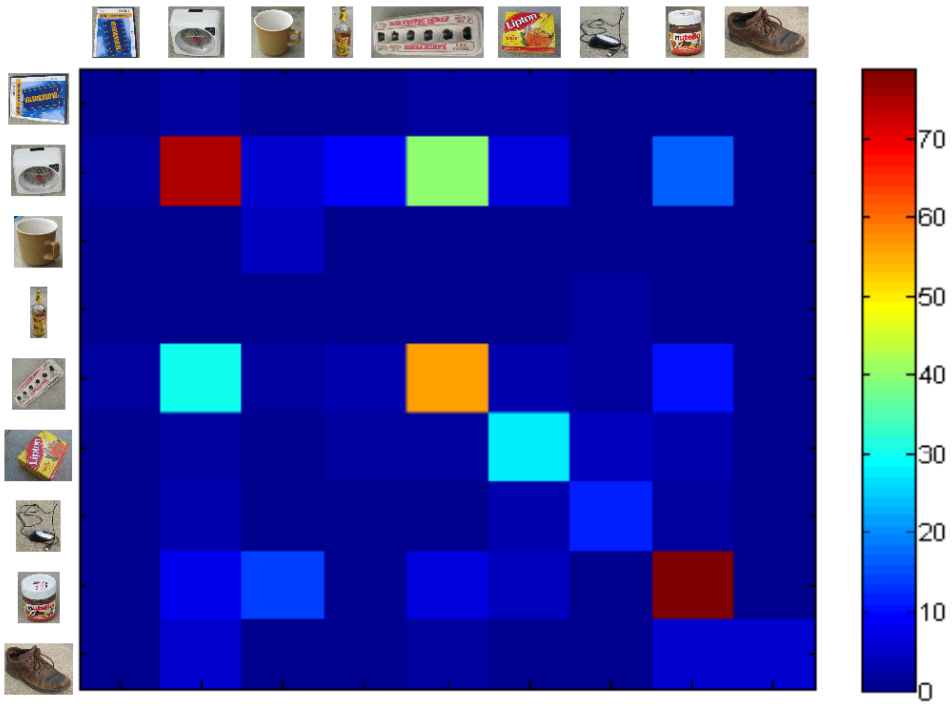

Fig. 1. Confusion Matrix on a subset of the Caltech gadgets data set. Images on the left are matched to image on the top. The confusion matrix is color coded with values shown in the color bar. The entries on the diagonal (indicating a correct match) are maximal in 7 out of the 9 objects.

the score (the number of fuzzy-matching interest points found) of the image on the left to the image on the top, where the colors indicate the number of matches. We empirically set the fuzzy matching threshold t to 0.8 (i.e., at least 13 out of the 16 characters must match) and, again, allowed for one-to-many matches. The correct match was found in the 7 out of the 9 pairs. Oblivious Matching failed on the first and fourth pairs. Note however that in four cases (pairs $1, 3, 4, 9$) the number of matches found was fairly low (below 10 matches) which means that the system should probably discard such a match. This low number of matches is due to the fact that a fairly tight threshold was chosen - having at least 13 out of the 16 characters match. Lowering the threshold will increase the number of false matches.