MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Blind Vision

Shai Avidan, Moshe Butman

TR2006-006 May 2006

Abstract

Alice would like to detect faces in a collection of sensitive surveillance images she own. Bob has a face detection algorithm that he is willing to let Alice use, for a fee, as long as she learns nothing about his detector. Alice is willing to use Bobś detector provided that he will learn nothing about her images, not even the result of the face detection operation. Blind vision is about applying secure multi-party techniques to vision algorithms so that Bob will learn nothing about the images he operates on, not even the result of his own operation and Alice will learn nothing about the detector. The proliferation of surveillance cameras raises privacy concerns that can be addressed by secure multi-party techniques and their adaptation to vision algorithms.

European Conference on Computer Vision (ECCV)

Blind Vision

Shai Avidan[1] and Moshe Butman[2]

[1] Mitsubishi Electric Research Labs,
201 Broadway, Cambridge, MA 02139
avidan@merl.com
[2] Department of Computer Science,
Bar-Ilan University, Ramat-Gan Israel
butmanm@cs.biu.ac.il

Abstract. Alice would like to detect faces in a collection of sensitive surveillance images she own. Bob has a face detection algorithm that he is willing to let Alice use, for a fee, as long as she learns nothing about his detector. Alice is willing to use Bob's detector provided that he will learn nothing about her images, not even the result of the face detection operation. ***Blind vision*** is about applying secure multi-party techniques to vision algorithms so that Bob will learn nothing about the images he operates on, not even the result of his own operation and Alice will learn nothing about the detector. The proliferation of surveillance cameras raises privacy concerns that can be addressed by secure multi-party techniques and their adaptation to vision algorithms.

1 Introduction

The proliferation of surveillance cameras raises privacy concerns that must be addressed. One way of protecting privacy is to encrypt the images on their way from the camera to the remote server that controls it. However, in some cases this might not be enough. For instance, when the client does not wish to reveal the content of the image even to the server that runs the particular vision algorithm. Consider, for example, a service center offering face detection capabilities over the web. Clients might be interested in the service but reluctant to reveal the content of their images, even to the service provider, either because they don't want the service center to learn the content of the image or they are concerned that virus attacks on the service center will reveal the content of the images. With slight modification the proposed algorithm can be used for blind face recognition. For example, a government agency can have photos of suspects and compare them to images taken from private surveillance cameras without learning anything about the content of the images (so as not to invade privacy), and without revealing the photos of the suspects. The only answer the government agency will learn is either a given suspect appear in a particular image or not. Another application might be in camera phones that does not have the CPU power to run heavy vision algorithms and would like to run the application securely on a remote server. Yet another application is blind OCR

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part III, LNCS 3953, pp. 1–13, 2006.

in which the client is not willing to reveal the content of the document to the server. In these cases one can resort to secure multi-party protocols that allow two parties to execute a given algorithm without learning anything about the other party.

Here we investigate the use of secure multi-party protocols for image analysis. This is a challenging task because secure multi-party protocols are known to be computationally intensive and applying them to large data sets, such as images and video streams makes the task even harder. Domain-specific constraints allow us to devise new schemes that are faster to use but might not be applicable to general secure multi-party problems.

As a concrete setup we focus on a surveillance scenario in which Alice owns a surveillance camera and Bob owns a server that runs a face detection algorithm. In our hypothetical scenario Alice and Bob will engage in a protocol that will allow Alice to learn if, and where, are faces in her images without learning anything about Bobs' detector. Bob will learn nothing about the images, not even if faces were detected in them.

We adopt secure multi-party protocols to derive a secure classification protocol. The protocol allows Alice to send Bob a candidate detection window and get a yes/no answer to the question "Is there a face in this window?". This results in a secure protocol that leaks no information to either party, but is slow in practice because of the use of cryptographic primitives. Then we suggest ways to drastically reduce the number of detection windows that Alice needs to send to Bob by using a non-cryptographic protocol that is very fast in practice but is not as secure as the secure classification protocol.

2 Background

Secure multi-party computation originated from the work of Yao [16] who gave a solution to the two-party problem where two parties are interested in evaluating a given function that takes as input private input from each party. As a concrete example consider the millionaire problem: Two parties want to find which one has a larger number, without revealing anything else about the numbers themselves. Later, Goldriech *et al.* [7] extended the case to $n > 2$ parties. However, the theoretical construct was still too demanding to be of practical use. An easy introduction to Cryptography is given in [14] and a more advanced and theoretical treatment is given in [6].

Since then many secure protocols were reported for various applications. Of particular interest here are those dealing with oblivious transfer [2], secure dot-product [1] or oblivious polynomial evaluation in general [11, 3] and learning decision trees [9]. Oblivious Polynomial Evaluation (OPE) [11, 3] assumes that Bob has a polynomial $P(x)$ and Alice wants to evaluate the polynomial for a particular x, unknown to Bob, without learning anything about the polynomial coefficients. This was later used by [9] to devise an ID3 decision tree learning algorithm where each party holds part of the training data, yet both parties are interested in learning a decision tree that uses all the available training data.

In the end both parties learn the parameters of the decision tree, but nothing about the training data of the other party.

Secure multi-party protocols are often analyzed for correctness, security and complexity. Correctness is measured by comparing the proposed protocol to the ideal protocol where the parties transfer their data to a trusted third party that performs the computation. If the secure protocol is identical to the ideal protocol then the protocol is declared correct (note that one might come up with secure approximation to an ideal algorithm). In security one needs to show what can and cannot be learned from the data exchange between the parties. One often assumes that the parties are *honest but curious*, meaning that they will follow the agreed-upon protocol but will try to learn as much as possible from the data-flow between the two parties. Put another way, one party is willing to trust the other party but is concerned that virus attacks on the other party will reveal the information. Finally, in complexity, one shows the computational and communication complexity of the secure algorithm.

3 Notations

All computations must be done over some finite field F that is large enough to represent all the intermediate results. One can approximate float numbers with fixed arithmetic and represent it as integer numbers in this field. Denote by $\mathbf{X}$ the image that Alice owns. A particular detection window within the image $\mathbf{X}$ will be denoted by $\mathbf{x} \in F^L$ and $\mathbf{x}$ will be treated in vector form. Bob owns a strong classifier of the form

$$H(\mathbf{x}) = sign(\sum_{n=1}^{N} h_n(\mathbf{x})), \tag{1}$$

where $h_n(\mathbf{x})$ is a threshold function of the form

$$h_n(\mathbf{x}) = \begin{cases} \alpha_n & \mathbf{x}^T\mathbf{y_n} > \Theta_n \\ \beta_n & \text{otherwise,} \end{cases} \tag{2}$$

and $\mathbf{y_n} \in F^L$ is the hyperplane of the threshold function $h_n(\mathbf{x})$. The parameters $\alpha_n \in F$, $\beta_n \in F$ and $\Theta_n \in F$ of $h_n(\mathbf{x})$ are determined during training; N is the number of weak classifiers used.

4 Secure Classification

In this section we develop a secure classifier that is based on a linear combination of simple threshold function ('stumps'). However, the ideas presented here can be used to develop other classifiers as well. For example, one can use the OPE protocol mentioned earlier to construct a polynomial-kernel SVM. Work still needs to be done to construct RBF-kernel SVM, or sigmoid-based neural network.

There is an inherent tension between secure multi-party methods and machine learning techniques in that one tries to hide and the other tries to infer. In the extreme case, Alice can use Bob to label training data for her so that she can later use the data to train a classifier of her own. The best we can hope for is to ensure that Bob will not learn anything about Alice's data and that Alice will not help her own training algorithm, other than supplying it with labeled examples, by running the secure classification protocol.

The cryptographic tool we will be using is Oblivious Transfer. Oblivious Transfer allows Alice to choose one element from a database of elements that Bob holds without revealing to Bob which element was chosen and without learning anything about the rest of the elements. In the following we will denote OT_1^M to indicate that Alice needs to chose one out of M elements. We will use OT to develop a series of secure sub-protocols that result in a secure classification protocol.

4.1 Oblivious Transfer

Oblivious Transfer allows Alice to choose one element from a database of elements that Bob holds without revealing to Bob which element was chosen and without learning anything about the rest of the elements. The notion of oblivious transfer was suggested by Even, Goldreich and Lempel [5] as a generalization of Rabin's "oblivious transfer" [13].

Bob privately owns two elements M_0, M_1 and Alice wants to receive one of them without letting Bob know which one. Bob is willing to let her do so provided that she will not learn anything about the other elements. The following protocol, based on RSA encryptions can be used to solve the problem in a semi-honest (i.e. honest but curious) setting.

Algorithm 1. Oblivious Transfer

Input: Alice has $\sigma \in \{0, 1\}$
Input: Bob has two strings M_0, M_1
Output: Alice learns M_σ.

1. Bob sends Alice two different public encryption keys K_0 and K_1.
2. Alice generates a key K and encrypts it with K_0 or K_1. For the sake of argument, let's say she chooses K_0. She sends Bob $E(K, K_0)$; that is, she encrypts K with one of Bob's public keys.
3. Bob does not know which public key Alice used, so he decrypts with both of his private keys. He thus obtains both the real key K, and a bogus one K'.
4. Bob sends Alice $E(M_0, K)$ and $E(M_1, K')$, in the *same* order he sent the keys K_0 and K_1 in step 1. Alice decrypts the first of these messages with the key K and obtains M_0.

Can Alice cheat? She would need to be able to find K', but she cannot do this unless she knows how to decrypt messages encrypted with the public key K_1.

Can Bob cheat? He would have to be able to determine which one of K and K' was the key Alice generated. But K and K' both look like random strings.

4.2 Secure Dot Product

Before diving into the technical details, let us give an intuitive introduction. Our goal is to break the result of the dot product operation $\mathbf{x}^T\mathbf{y}$ into two shares a and b, where a is known only to Alice, b is known only to Bob and it holds that $\mathbf{x}^T\mathbf{y} = a + b$. We do this by breaking the product of every pair of elements $\mathbf{x_i} * \mathbf{y_i}$ into two shares $\mathbf{a_i}$ and $\mathbf{b_i}$ and then letting Alice and Bob sum the vectors $\mathbf{a}$ and $\mathbf{b}$, respectively to obtain shares of the dot product. Observe that $\mathbf{a_i}$ and $\mathbf{b_i}$ must sum to $\mathbf{x_i} * \mathbf{y_i}$ where $\mathbf{x_i}$ is in the range $[0, 255]$ and $\mathbf{y_i} \in \{-1, 0, 1\}$ so the size of the field F should be at least 512 to accommodate all possible cases. The details are given in protocol 2.

Algorithm 2. Secure dot-product

Input: Alice has vector $\mathbf{x} \in F^L$
Input: Bob has vector $\mathbf{y} \in F^L$
Output: Alice and Bob have private shares a and b s.t. $a + b = \mathbf{x}^T\mathbf{y}$

1. Bob generates a random vector $\mathbf{b} \in F^L$
2. For each i=1...L, Alice and Bob conduct the following sub-steps
   (a) Bob enumerates all possible $\mathbf{x_i}$ values and constructs a $256D$ vector $\mathbf{a}$, s.t.

$$\mathbf{a_i} = \mathbf{y_i} * \mathbf{x_i} - \mathbf{b_i} \qquad \mathbf{x_i} \in [0...255]$$

   (b) Alice uses OT_1^{256} with $\mathbf{x_i}$ as her index, to choose the appropriate element from the vector $\mathbf{a}$ and stores it as $\mathbf{a_i}$.
3. Alice and Bob sum their private vectors $\mathbf{a}$ and $\mathbf{b}$, respectively, to obtain the shares $a = \sum_{i=1}^{L} \mathbf{a_i}$ and $b = \sum_{i=1}^{L} \mathbf{b_i}$ of the dot-product $\mathbf{x}^T\mathbf{y}$.

Correctness. The protocol is clearly correct.

Security. The protocol is secure for both parties as we will show next

- From Alice to Bob
  - In step 2(b) Alice uses OT with $\mathbf{x_i}$ as an index to choose an element from the vector $\mathbf{a}$. Because OT is secure, Bob can not learn which element she chose and hence can learn nothing about the vector $\mathbf{x}$.
- From Bob to Alice
  - For each element, Bob lets Alice pick one element from the vector $\mathbf{a}$ and since $\mathbf{a}$ is the sum of the vector $\mathbf{y}$ with some random vector $\mathbf{b}$, Alice can learn nothing about $\mathbf{y}$ from $\mathbf{a}$.

Complexity and Efficiency. The protocol is linear in L - the dimensionality of the vectors $\mathbf{x}$ and $\mathbf{y}$.

4.3 Secure Millionaire

Alice and Bob would like to compare and find which one has a larger number, without revealing anything else about their number [16]. We show here a solution to the problem based on the OT primitive. The idea is to have Alice and Bob represent their numbers in binary format, scan it one bit at a time from left (most significant bit) to right (least significant bit) and then get the result. For each bit Bob should prepare a lookup table that is based on his current bit value and the two possible bit values of Alice. Alice will use OT_1^2 to obtain some

Algorithm 3. Secure Millionaire

Input: Alice has a number $x \in F$
Input: Bob has a number $y \in F$
Output: Alice and Bob find out if $x > y$

1. Bob defines three states $\{\mathcal{A}, \mathcal{B}, \mathcal{U}\}$ that correspond to: $\mathcal{A}$lice has a larger number, $\mathcal{B}$ob has a larger number and $\mathcal{U}$ndecided, respectively. For each bit, Bob encodes $\{\mathcal{A}, \mathcal{B}, \mathcal{U}\}$ using a different permutation of the numbers $\{1, 2, 3\}$.
2. For the left most bit, Bob constructs a 2-entry lookup table $\mathbf{z}^{(n)}$ using the following table.

	$y_n = 0$	$y_n = 1$
$x_n = 0$	$\mathcal{U}$	$\mathcal{B}$
$x_n = 1$	$\mathcal{A}$	$\mathcal{U}$

   where x_n, y_n are the left most (most significant) bit of the numbers x, y, respectively. If $y_n = 0$ then Bob should construct a table from the left column, otherwise he should use the right column.
3. Alice uses OT_1^2 with x_n as her index to obtain $s^{(n)} = z^{(n)}(x_n)$
4. For each $i = n-1, ..., 1$, Alice and Bob conduct the following sub-steps
   (a) Bob constructs a 6-entry lookup table $\mathbf{z}^{(i)}$ that is indexed by $s^{(i+1)}$ and x_i, s.t.

	$y_i = 0$	$y_i = 1$
$s^{(i+1)} = \mathcal{A} \wedge x_i = 0$	$\mathcal{A}$	$\mathcal{A}$
$s^{(i+1)} = \mathcal{B} \wedge x_i = 0$	$\mathcal{B}$	$\mathcal{B}$
$s^{(i+1)} = \mathcal{U} \wedge x_i = 0$	$\mathcal{U}$	$\mathcal{B}$
$s^{(i+1)} = \mathcal{A} \wedge x_i = 1$	$\mathcal{A}$	$\mathcal{A}$
$s^{(i+1)} = \mathcal{B} \wedge x_i = 1$	$\mathcal{B}$	$\mathcal{B}$
$s^{(i+1)} = \mathcal{U} \wedge x_i = 1$	$\mathcal{A}$	$\mathcal{U}$

      where $s^{(i+1)}$ is the state variable from the previous bit. If $y_i = 0$ then Bob should construct a table from the left column, otherwise he should use the right column.
   (b) Alice uses OT_1^6 with $s^{(i+1)}$ and x_i as her indices to obtain $s^{(i)} = \mathbf{z}^{(i)}(s^{(i+1)}, x_i)$
5. Bob sends Alice the meaning of the three states of $s^{(1)}$ of the least significant bit. Alice now knows which number is larger.
6. If she wants, Alice can send the result to Bob.

intermediate result and they both will continue to the next bit. The problem with this approach is that comparing least significant bits is meaningless if the most significant bits were already used to determine which number is larger. Note, also, that Alice and Bob should not abort in the middle of the scan as this might reveal some information about the numbers themselves. To solve this problem we will use a state variable s that can take one of three states: $\mathcal{A}$ Alice has a larger number, $\mathcal{B}$ Bob has a larger number or $\mathcal{U}$ Undecided yet. For each bit Bob constructs a 6-way lookup table that consists of the 3 states of s and the two possible values of the next bit of Alice, the output is the new state after evaluating the current bit. For example, if $s = \mathcal{A}$, Bobs' current bit is 1 and Alice's' current bit is 0 then the output should be $s = \mathcal{A}$ and they both move to the next bit. To prevent Alice from interpreting the state s Bob can use different numbers to represent $\mathcal{A}, \mathcal{B}, \mathcal{U}$ for each bit so, for example, for the first bit $\mathcal{A}$ is represented as the number 1 but for the second bit 1 might represent the symbol $\mathcal{B}$. The details are given in protocol 3.

Correctness. The protocol is clearly correct.

Security. The protocol is secure for both parties as we will show next

- From Alice to Bob
  - In steps 3 and 4b Alice uses x_i as her index in the OT operation. Since OT is secure, Bob can learn nothing about the number x.
- From Bob to Alice
  - For each bit, Bob lets Alice pick one element from the lookup table $\mathbf{z}$ and returns the state s. Since the values of the state s are represented using random numbers for each bit, Alice cannot determine what does a change in s mean and can not learn anything about the number y, other than learning, in the end, if $x > y$.

Complexity and Efficiency. The protocol is linear in the number of bits of the numbers x and y.

4.4 Secure Classifier

We are now ready to present the secure classifier protocol. The protocol relies on the secure dot-product and Millionaire protocols and the details are given in protocol 4.

Correctness. The protocol is clearly correct.

Security. The protocol protects the security of both parties.

- From Alice to Bob
  - In step 2(a) Alice and Bob engage in a secure dot-product protocol so Bob learns nothing about the vector $\mathbf{x}$.
  - In step 2(b) and 3 Alice and Bob engage in secure Millionaire protocol so Bob can learn nothing about Alice's data.

Algorithm 4. Secure Classifier

Input: Alice has input test pattern $\mathbf{x} \in F^L$
Input: Bob has a strong classifier of the form $H(\mathbf{x}) = sign(\sum_{n=1}^{N} h_n(\mathbf{x}))$
Output: Alice has the result $H(\mathbf{x})$ and nothing else
Output: Bob learns nothing about the test pattern $\mathbf{x}$

1. Bob generates a set of N random numbers: $s_1, ..., s_N$, such that $s = \sum_{n=1}^{N} s_n$
2. For each $n = 1, ..., N$, Alice and Bob conduct the following sub-steps:
   (a) Alice and Bob obtain private shares a and b, respectively, of the dot product $\mathbf{x}^T\mathbf{y_n}$ using the secure-dot-product protocol.
   (b) Alice and Bob use the secure Millionaire protocol to determine which number is larger: a or $\Theta_n - b$. Instead of returning $\mathcal{A}$ or $\mathcal{B}$ the secure Millionaire protocol should return either $\alpha_n + s_n$ or $\beta_n + s_n$. Alice stores the result in $\mathbf{c_n}$.
3. Alice and Bob use the secure Millionaire protocol to determine which number is larger: $\sum_{n=1}^{N} \mathbf{c_n}$ or $\sum_{n=1}^{N} \mathbf{s_n}$. If Alice has a larger number then $\mathbf{x}$ is positively classified, otherwise $\mathbf{x}$ is negatively classified.

– From Bob to Alice
  - In step 2(a) Alice and Bob engage in a secure dot-product protocol so Alice learns nothing about Bobs' data.
  - In step 2(b) Alice and Bob engage in a secure Millionaire protocol so Alice only learns if $a > \Theta_n - b$ but since she does not know b she can not learn anything about the parameter Θ_n. Moreover, at the end of the Millionaire protocol Alice learns either $\alpha_n + s_n$ or $\beta_n + s_n$. In both cases, the real parameter (α_n or β_n) is obfuscated by the random number s_n.
  - In step 3 Alice learns if her number $\sum_{n=1}^{N} \mathbf{c_n}$ is greater than Bob's number $\sum_{n=1}^{N} \mathbf{s_n}$. Since $\mathbf{s}$ is a random vector, she can gain no knowledge about the actual parameters of Bobs' strong classifier.
  - Alice can learn the number of weak classifier N from the protocol. This can easily be fixed if Bob will add several fake weak classifiers $h_n(\mathbf{x})$ whose corresponding weights α_n, β_n are zero. This way Alice will only learn an upper bound on N and not N itself.

Complexity and Efficiency. The complexity of the protocol is $O(NLK)$, where N is the number of weak classifiers used, L is the dimensionality of the test vector $\mathbf{x}$ and K it the number of bits in the dot-product $\mathbf{x}^T\mathbf{y_n}$.

Applying the secure classification protocol to face detection is straightforward. Alice scans her image and sends each detection window to Bob for evaluation. Bob learns nothing about the image and Alice only gets a binary answer for every detection window. The problem with the protocol is speed. As we discuss in the experimental section, it might take from a few seconds to a few minutes to classify a detection window (depending on the number of levels in the rejection cascade, see details in the experiments section). This means that the protocol is prohibitively expensive to compute in practice. Therefor we investigate methods to accelerate it.

5 Accelerating Blind Vision

There are three methods to accelerate the above protocol. The first relies on cryptographic methods that leverage a small number of OT operations to perform a large number of OT [12, 8]. We will not explore these methods here.

A second approach would be for Bob to reveal a stripped-down version of his classifier to Alice. This way, Alice can run the stripped-down classifier on her data. This stripped-down classifier will effectively reject the vast majority of detection windows and will allow Alice to use the expansive secure protocol on a relatively small number of detection windows.

Finally, the last method of acceleration is to develop one-way hash functions that will allow Alice to quickly hash her data but still let Bob correctly classify the patterns without learning much about the original image itself. This will be used as a filter to quickly reject the vast majority of detection windows, leaving the "difficult" examples to be classified using the secure classification protocol.

5.1 Image Hashing Using Histograms of Oriented Gradients

There is a large body of literature on one way hash functions [14]. These functions take the input message (detection window in our case) and map it to some hashed vector in such a way that the original message can not be recovered. These one way hash functions are not suitable for our purpose because they map nearby patterns to different locations in hash space. So, two images that are nearby in image space might be mapped to far-apart vectors in the hash space. There is little hope then that a classifier will be able to learn something in the hash space, because the basic assumption that nearby patterns should have similar labels is violated.

We therefor use a domain-specific hash function. Specifically, we use the Histogram of Oriented Gradients (HoG) as our hash function. HoG was proved very useful in a variety of object recognition and detection applications [10, 4], yet it destroys the spatial order of pixels, as well as their absolute values, and is coarsely binned so we assume that recovering the original image patch from a given HoG is impossible. Figure 1 show some examples of face and non-face image patches and their corresponding HoGs.

In our system, Alice computes the HoG for each detection window and store each bin in a response image. We use 18 bin HoG so there are 18 response images used to represent the HoG for every detection window. That is, the 18 bins of the HoG of a particular detection window are stored at the central pixel location of that detection window, across all 18 response images.

By scrambling the order of pixels in the response images we effectively destroy the spatial relationship between the HoGs so Bob can not use this information to reconstruct the original image (the same scrambling permutation must be performed on all 18 response images). Figure 2 show how the response image that corresponds to one of the bins of the HoG looks like with and without scrambling the order of its pixels. Specifically, figure 2b shows a response image that corresponds to one bin in the HoG. Scrambling the order of the pixels (figure 2c)

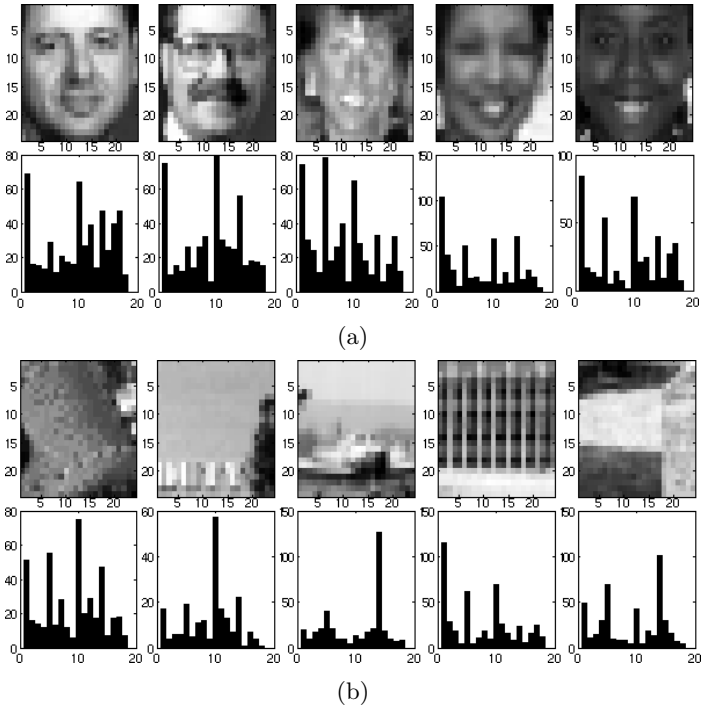
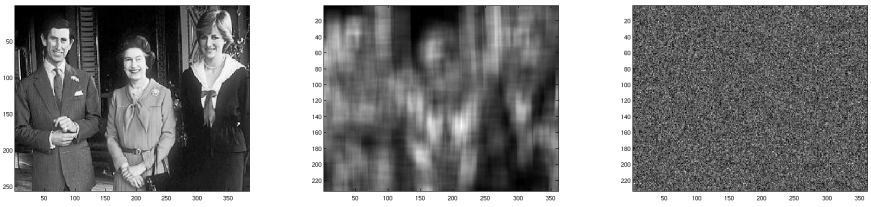


Fig. 1. Image to Histogram of Oriented Gradients (HoG) Hashing. (a) some examples of face images and their corresponding HoG. (b) some example of non-face images and their corresponding HoG. We assume that it is impossible to reconstruct an image from its HoG.



(a) (b) (c)

Fig. 2. The importance of scrambling. (a) original image. (b) Image of the first bin of the Histogram of Oriented Gradients (HoG). (c) Same as (b) after pixel scrambling.

destroys the spatial relationship between HoGs. In addition, Alice can bury the scrambled image in a larger image that contain random values (not shown here).

The inclusion of fake HoGs, by burying the response images in a larger image, prevents Bob from recovering the original image, because he does not know if he is using HoGs that came from the original image. Moreover, it prevents Bob from knowing the result of his classification, because he does not know if the

HoGs that he classified as positive (i.e. originated from a detection window that contains a face) correspond to real or fake image patches.

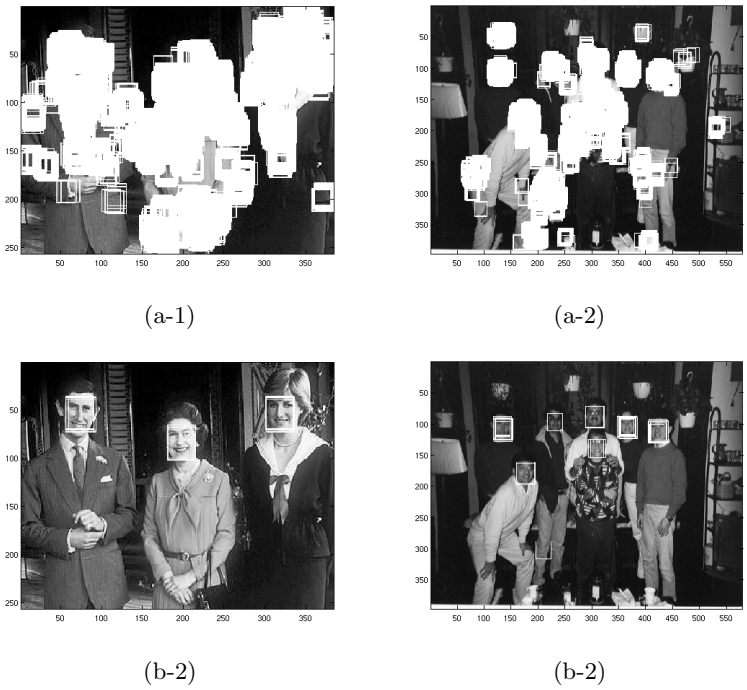
6 Experiments

We implemented the secure classification protocol in C++ using the NTL[1] package for large numbers and used RSA encryption with 128-bit long encryption keys. The HoG detector was implemented in MATLAB. We simulated Alice and Bob on one PC so communication delays are not reported here.

We converted our Viola-Jones type face detector [15] to a secure detector. In the process we have converted the integral-image representation to regular dot-product operation, a step that clearly slowed down our implementation as we no longer take advantage of the integral image representation. Also, we shifted the values of the filters from the range $[-1, 1]$ to the range $[0, 2]$ to ensure that all values are non-negative integers. We then converted all the thresholds to non-negative integers and updated them to reflect the shift in the filter values. The face detector consists of a cascade of 32 rejectors, where each rejector is of the form presented in equation 1. The first rejector requires 6 dot-product operations and the most complicated rejector require 300 dot-products. There is a total of 4303 dot-products to perform. Instead of computing the secure dot-product for each filter, we use OT to compute the secure dot-product for all the weak classifiers in a given level and allowed Alice and Bob to make a decision after every level of the cascade. This clearly reveal some information to Alice, as she knows at what level of the cascade a pattern was rejected but it greatly accelerates the performance of the program. We found that a single 24×24 detection window can be classified in several minutes using all the levels of the cascade. In most cases the first two levels of the cascade are enough to reject a pattern and they can be processed in a few seconds per detection window. As expected, the main bottleneck of the protocol is the extensive use of the OT operation.

To accelerate performance we used the HoG based image hashing. Each 24×24 detection window was mapped to HoG as follows. Alice first computes the gradient of every pixel and ignores every pixel whose x and y gradients were below 5 intensity values. Then she binned the gradient orientation into 18 bins and stored the result in a histogram. She then sends the HoGs, in random order and together with some fake HoGs, to Bob. Bob's HoG detector consists of a cascade of 45 levels. Each level of the cascade consists of a feed-forward neural network with 5 hidden units that was trained to reject as many negative examples as possible, while maintaining 98% of its positive examples. The unoptimized HoG detector takes several seconds to process a single 240×320 image. We found that on average the HoG detector rejects about 90% of the detection windows in an image. The remaining 10 percent are classified using the secure classifier protocol described earlier. In a typical case, about $15,000$ detection windows (out of a total of about $150,000$ detection windows) will be passed to the secure classification protocol. This approach accelerates secure classification

[1] Downloaded from http://www.shoup.net/ntl/index.html



(a-1) (a-2)

(b-2) (b-2)

Fig. 3. Blind Face Detection. (a) result after running the HoG detection. (b) Final detection result.

by an order of magnitude, at the risk of revealing some information. There is clearly a trade-off between the quality of the HoG detector and the amount of information revealed.

Figure 3 show some typical results. The top row shows the result of the HoG detection, as Alice sees them. The bottom row shows the result, as Alice sees it, after the secure classification. A couple of comments are in order. First, note that after the HoG detection the only thing that Bob knows is that he detected several thousands candidates. He does not know their spatial relationship, how they actually look or if they came from the original image or are simply chaff designed to confuse him. Second, the HoG detector is performed in a multi-scale fashion. In our case Alice uses a 3 level pyramid with a scale factor of 1.2 between scales. Finally, all the detection windows that were positively classified by the HoG detector are then scaled to 24×24 windows and fed to the secure classifier protocol.

7 Conclusions

Blind Vision applies secure multi-party techniques to image related algorithms. As an example we have presented a blind face detection protocol that reveals no information to either party at the expanse of heavy computation load. We

then suggested image hashing technique, using Histogram of Oriented Gradients (HoG) to accelerate the detection process, at the risk of revealing some information about the original image. There are several extensions to this work. First is the need to accelerate the detection process. Second is the need to develop secure versions to other popular classifiers such as RBF or sigmoid function. Third, we are investigating information theoretic approaches to analyze the amount of information leaked by the HoG hash function, as well as developing better and more secure image hashing functions. Finally we are exploring ways to extend Blind Vision to other vision algorithms such as object tracking or image segmentation.

References

1. M. J. Atallah and W. Du. *Secure multi-party computational geometry.* In WASDS2001: 7th International Workshop on Algorithms and Data Structures, pp 165-179, Providence Rhode Island, USA, August 8-10 2001.
2. C. Cachin. *Efficient private bidding and auctions with an oblivious third party.* In Proceedings of the 6th ACM conference on Computer and Communications Security, pp 120-127, Singapore, November 1-4 1999.
3. Y.C. Chang and C.J. Lu. *Oblivious polynomial evaluation and oblivious neural learning.* In AsiaCrypt: Advances in Cryptology. LNCS, Springer-Verlag, 2001.
4. N. Dalal and B. Triggs. *Histograms of Oriented Gradients for Human Detection.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005.
5. S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM 28, pp. 637-647, 1985.
6. O. Goldreich, *Foundations of Cryptography*, 2004.
7. O. Goldreich, S. Micali and A. Wigderson. *How to play any mental game - a completeness theorem for protocols with honest majority.* In 19th ACM Symposium on the Theory of Computing, pp 218-229, 1987.
8. Y. Ishai, J. Kilian, K. Nissim and E. Petrank. *Extending Oblivious Transfers Efficiently.* In CRYPTO 2003, pp 145-161.
9. Y. Lindell and B. Pinkas, *Privacy preserving data mining.* In Advances in Cryptology - Crypto2000, LNCS 1880, 2000.
10. D.G. Lowe, *Distinctive image features from scale-invariant keypoints.* International Journal of Computer Vision, 60(2):91-110, 2004.
11. M. Naor and B. Pinkas, *Oblivious Polynomial Evaluation.* In Proc. of the 31st Symp. on Theory of Computer Science (STOC), Atlanta, GA, pp. 245-254, May 1-4, 1999.
12. M. Naor and B. Pinkas, *Efficient Oblivious Transfer Protocols.* In Proc. of the twelfth annual ACM-SIAM symposium on Discrete algorithms , Washington, D.C., USA pp. 448-457, 2001.
13. M. O. Rabin, *How to exchange secrets by oblivious transfer*, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.
14. B. Schneier, *Applied Cryptography*, 1996.
15. P. Viola and M. Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features.* In IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, 2001.
16. A. C. Yao, *How to generate and exchange secrets*, 27th FOCS, pp. 162-167, 1986.