SECURE IMAGE FILTERING

Nan Hu, Sen-ching S. Cheung

University of Kentucky
Electrical & Computer Engineering
Lexington, KY, 40506
nan.hu@uky.edu, cheung@engr.uky.edu

Thinh Nguyen

Oregon State University
EECS Department
Corvallis, OR 97330
thinhq@eecs.oregonstate.edu

ABSTRACT

In today's heterogeneous network environment, there is a growing demand for *distrusted* parties to jointly execute distributed algorithms on private data whose *secrecy* needed to be safeguarded. Protocols that support such kind of joint computation without complete sharing of information are called Secure Multiparty Computation (SMC) protocols. Applying SMC protocols in image processing is a challenging problem. Most of the existing SMC protocols are implemented based on cryptographic primitives like Oblivious Transfer that are too computational intensive for pixel-based operations. In this paper, we develop two efficient SMC protocols for distributed linear image filtering between two parties, one party with the original image and the other with the image filter. The first protocol is based on a combination of rank reduction and random permutation. The second one uses random perturbation with the help of a non-colluding third party. Experimental results show that both of them execute significantly faster than oblivious-transfer based techniques.

Index Terms— Communication system security, Image Processing, Distributed Algorithms, Cryptography

1. INTRODUCTION

The proliferation of imaging and storage devices and the ubiquitous presence of computer networks make sharing of visual data easier than ever. Such casual exchange of visual data, however, has increasingly raised questions on how sensitive visual information can be protected. Consider the scenario in which a user of a cellular-phone camera wants to send his/her pictures to an online photo-processing laboratory for image enhancement such as red-eye removal. The user would be concerned about the privacy of his/her pictures while the online store would need to protect the proprietary enhancement technologies against reverse-engineering. Consider another scenario that the police wants to search for possible suspects in a surveillance video owned by private company A, using a proprietary software from yet another private company B. The three parties involved (police, company A, company B) all have information they do not want to share with each other (criminal biometric database from the police, surveillance tape from company A and proprietary software from company B). To support such kinds of applications, one needs to establish a joint computation and communication platform at the *pixel level* that can guarantee the secrecy of private data and algorithms, and at the same time achieve a well-defined objective that benefits all parties involved.

This is, however, not a new problem. Cryptographic protocols that support joint computation among multiple distrusted parties without complete sharing of information are called Secure Multiparty Computation (SMC) protocols. The general problem of SMC can be traced back to the classical paper by Yao [1]. The biggest challenge to any SMC technique is how to satisfy the security requirement. There are two types of security models - the *ideal* model and the *real* model [2]. In the ideal model, the computation is performed by a separate party trusted by all participants. This is perfectly secure as none of the participants gain information beyond the knowledge of their own data. However, the ideal model is not practical because all critical data reside at a single party and can easily be compromised. In the real model, the computation is performed via a *SMC protocol* agreed upon by all parties. The protocol is said to be secure in a *semi-honest* environment if all parties respect the protocol and are not able to derive more information than what can be deduced from the final results. Most of the SMC protocols, including those described in this paper, are developed under this model. Many SMC problems can be solved by using the Oblivious Transfer (OT) protocol in which one party (Alice) can select a particular item from a set owned by another party (Bob) without Bob knowing Alice's selection or Alice knowing other items from Bob's set [2]. The OT Transfer is typically implemented using different public-key encryption systems [3].

There has been little work in applying SMC to image processing problems. The only work known to us is by S. Avidan et al. on applying classical SMC protocols for two-party face detection [4]. In a typical classification task such as face detection, a significant portion of an image is transformed into feature vectors, which in most cases cannot be used to recover the original image. The manipulation of feature vectors is thus secure by definition and no special SMC protocols are required. As a result, the complex SMC protocols do not significantly affect the overall performance of the classification task. On the other hand, many common image processing applications require pixel-by-pixel processing. The sheer number of pixels in common images render most of the classical SMC techniques useless. Thus, it is imperative to develop fast computation techniques for these applications. Among all the image processing techniques, linear filtering is arguably the most basic and useful one. It is used in almost all image processing and computer vision applications such as enhancement, denoising, and feature extraction. Even though linear filtering by itself is inherently insecure as we will demonstrate in Section 2, we focus in this paper on "secure" and efficient linear filtering algorithms that can be used in conjunction with other types of non-linear processing. Our contributions are two novel secure linear filtering protocols that are significantly faster than those based on OT. These protocols are capable of processing mega-pixel images in less than one second using common desktops.

The organization of the paper is as follows: our starting point in Section 2 is the problem definition and some notations of linear filter-

ing. The classic two-party protocol based on OT is briefly reviewed in Section 3.1. In Section 3.2, we review the secure inner product protocol in [5] and introduce our two-party linear filtering protocol based on this inner product protocol. One drawback of this new protocol is that it cannot be used successively to implement multi-stage linear filtering. To solve this problem, we need the help from a third party. As we show in Section 3.3, the protocol guarantees that none of the parties have full knowledge of the data as long as no two parties collude to share data externally. The efficiency of the proposed techniques are demonstrated by the experimental results in Section 4. Finally, we conclude the paper and describe ongoing work in Section 5.

2. PROBLEM DEFINITION AND NOTATIONS

Given an image $\{x(m,n) : 0 \leq m \leq M, 0 \leq n \leq N\}$ and a filtering operation $f(\cdot)$ described by a set of parameters θ, we define the output $y(m,n)$ of applying this filter to x as follows:

$$y(m,n) = f(x;\theta) \tag{1}$$

In the secure image filtering model, we have two users, Alice and Bob, who own the image data x and the filter parameter θ respectively. Our goal is to establish a computation protocol between Alice and Bob so that

1. Alice obtains $f(x;\theta)$ without any knowledge of θ, and
2. Bob does not know anything about x.

For linear filtering, the filter parameters are specified as a filter mask $\mathbf{h}$ defined as $\{h(m,n) : -\frac{l}{2} \leq m \leq \frac{l}{2} and -\frac{l}{2} \leq n \leq \frac{l}{2}\}$. The linear filtering operation can then be written as

$$y(m,n) = x * h = \sum_{i=-l/2}^{l/2} \sum_{j=-l/2}^{l/2} h(i,j)x(m-i,n-j). \tag{2}$$

It is easy to see that Equation (2) is a scalar product between two $(l+1)^2$ dimensional vectors.

All of our secure linear filtering protocols use the following conceptual model: Alice first forms a $MN \times (l+1)^2$ matrix X_w whose rows are the signal data needed for the inner product operation. The total number of rows of X_w is the total number of pixels in the output image[1]. If we denote the i-th row of X_w as $X_w(i,:)$ and the output image as a vector $\mathbf{y} = [y_1 \cdots y_{MN}]^T$, the image filtering could be written as

$$\mathbf{y} = X_w\mathbf{h} \tag{3}$$

A secure linear filtering protocol decomposes every $y_i = X_w(i,:)\cdot\mathbf{h}$ into $y_i = a + b$ such that Alice computes a without any knowledge of $\mathbf{h}$ and Bob computes b without any knowledge of $X_w(i,:)$.

If linear filtering is the end goal of the processing, Bob sends back his portion to Alice to compute the output image y. Using both the input x and y, Alice can estimate $\mathbf{h}$ using the least square estimate

$$\hat{\mathbf{h}} = (X_w^T X_w)^{-1} X_w \mathbf{y}.$$

In other words, linear filtering is *intrinsically insecure* to Bob no matter how secure the protocols are. General non-linear filtering, on the other hand, is much harder to invert based on limited number of input and output pairs. If linear filtering is used as part of a secure non-linear image processing system, it is thus important that *neither Alice nor Bob has the entire output of the linear filtering protocol.*

[1] X_w can be made to have MN rows with appropriate boundary handling.

3. SECURE LINEAR FILTERING

In this section, we will develop two types of secure filtering protocol: 1) a two-party protocol based on rank reduction and random permutation and 2) a three-party protocol based on random perturbation of the data. Before introducing the new protocols, we first review the classic two-party protocol based on OT. A detailed explanation of OT could be found in [2], and an application of OT in face detection could be found in [4].

3.1. Classical Two-party Solution

Oblivious transfer allows Alice to select one element from the whole dataset Bob holds without revealing to Bob which element Alice has selected and without knowing any other element in the dataset rather than the one selected. Our current notion of oblivious transfer was originally suggested by Even, Goldreich and Lempel [6] as a generalization of Rabin's "oblivious transfer" [7]. We will denote OT_1^m to indicate the protocol that Alice choose one out of the m elements from Bob's dataset. The state-of-the-art OT_1^m requires computing at least two long integer modular exponential, typically in the range of 512 bit to 1024 bit, per selection [3].

Secure scalar product can be implemented based on oblivious transfer as shown in [4]. Alice and Bob have $\mathbf{x}, \mathbf{h} \in F^n$ respectively, where F is a finite field (0-255 for grayscale image pixels). Let the total number of element in the finite field F be N_F. To compute $y = \sum_{i=1}^n x_i \cdot h_i$, Bob computes for each h_i a table of N_F entries. The j-th entry of the table is $j \cdot h_i - r_i$ where $r_i, 1 \leq i \leq n$ are random numbers known only to Bob. Alice and Bob then engage in N rounds of $OT_1^{N_F}$ protocols in which Alice selects the j-th entry of the table in the i-th round if $x_i = j$. Finally, Alice computes $a = \sum_{i=1}^n x_i \cdot h_i - r_i$ and Bob has $b = \sum_{i=1}^n r_i$. It is obvious that $y = a + b$ while Bob and Alice know nothing about each other's vector. To prevent Alice from inferring Bob's data based on successive execution of OT, Bob must use unpredictable encryption keys in each OT which also adds to the complexity of the algorithm.

3.2. Two-party Solution

Our first protocol on linear filtering is based on the secure inner protocol proposed in [5]. We describe this protocol using the pseudocode `InnerProductAlice` and `InnerProductBob` listed below. Alice has a n-dimensional vector $\mathbf{x}$ and Bob has a n-dimensional vector $\mathbf{y}$. They both know an invertible matrix P and its inverse P^{-1}. P is broken down into top and bottom halves $T \in \mathbb{R}^{\lfloor \frac{n}{2} \rfloor \times n}$ and $B \in \mathbb{R}^{(n-\lfloor \frac{n}{2} \rfloor)\times n}$, while P^{-1} into left and right halves $L \in \mathbb{R}^{n \times \lfloor \frac{n}{2} \rfloor}$ and $R \in \mathbb{R}^{n \times (n-\lfloor \frac{n}{2} \rfloor)}$. The inner product $\mathbf{x}^T\mathbf{y}$ can then be decomposed as follows:

$$\mathbf{x}^T\mathbf{y} = \mathbf{x}^T P^{-1} P\mathbf{y} = \mathbf{x}^T LT\mathbf{y} + \mathbf{x}^T RB\mathbf{y} \tag{4}$$

Alice then sends $\mathbf{x}^T R$ to Bob who computes $\mathbf{x}^T RB\mathbf{y}$ while Bob sends Alice $T\mathbf{y}$ so that she can compute $\mathbf{x}^T LT\mathbf{y}$.

The security of the protocols comes from the observation that $\mathbf{x}^T R$ and $T\mathbf{y}$ project $\mathbf{x}$ and $\mathbf{y}$ into lower-rank subspaces, and thus the components of the original vectors inside the null spaces of the matrices are irrecoverably lost. In [5], the authors proposed a design of P based on decoding matrices used in error control coding so as to spread the projections of neighboring vectors as far as possible. In this paper, we adopt this method to generate an invertible matrix G and compute the actual P by multiplying G with a random orthogonal matrix. This approach allows us to generate different P if the protocols require multiple invertible matrices. Unlike the OT-based

Algorithm 1 `InnerProductAlice`$(\mathbf{x}, P^{-1})$

Require: $\mathbf{x} \in \mathbb{R}^n$. $P^{-1} = (L \quad R)$ is a $n \times n$ invertible matrix where $n \geq 2$; $L \in \mathbb{R}^{n \times \lfloor n/2 \rfloor}$ and $R \in \mathbb{R}^{n \times (n - \lfloor n/2 \rfloor)}$.

1: $\mathbf{x_1} \leftarrow L^T \mathbf{x}$
2: $\mathbf{x_2} \leftarrow R^T \mathbf{x}$
3: Transmit $\mathbf{x_2}$ to Bob.
4: Receive $\mathbf{y_1}$ from Bob.
5: **return** $\mathbf{x_1}^T \mathbf{y_1}$

Algorithm 2 `InnerProductBob`$(\mathbf{y}, P)$

Require: $\mathbf{y} \in \mathbb{R}^n$. $P^T = (T^T \quad B^T)$ is a $n \times n$ invertible matrix where $n \geq 2$; $T \in \mathbb{R}^{\lfloor n/2 \rfloor \times n}$ and $B \in \mathbb{R}^{(n - \lfloor n/2 \rfloor) \times n}$.

1: $\mathbf{y_1} \leftarrow T\mathbf{y}$
2: $\mathbf{y_2} \leftarrow B\mathbf{y}$
3: Receive $\mathbf{x_2}$ from Alice.
4: Transmit $\mathbf{y_1}$ to Alice.
5: **return** $\mathbf{x_2}^T \mathbf{y_2}$

protocol which requires complex long-integer exponential and random key generation, this protocol requires only the highly optimized matrix multiplications.

It may seem intuitive to implement secure linear filtering by applying the inner product algorithm on X_w row by row. However, it is not secure as adjacent rows in X_w overlap with each other. As a result, the redundancy in the rank-reduced data sent to Bob allows him to form a least-square estimation of the original image. This least square problem involves solving a least-square data matrix of size $MN\lfloor \frac{(l+1)^2}{2} \rfloor \times MN$. To achieve secrecy, Alice needs to first randomly scramble the order of the rows in X_w before carrying out the inner product. Without the proper row order, Bob has no means of formulating the least-square estimation. The protocol is described in `FilterAlice` and `FilterBob`.

Algorithm 3 `FilterAlice`$(\mathbf{x}, P^{-1})$

Require: $\mathbf{x} \in \mathbb{R}^m$. P^{-1} is a $n \times n$ invertible matrix as described in `InnerProductAlice` where n is the length of the linear filter.

1: Reformat $\mathbf{x}$ into X_w.
2: Scramble the row order of X_w.
3: **for** $i \leftarrow 1$ to m **do**
4: $\quad \mathbf{r}_A(i) \leftarrow$ `InnerProductAlice`$(X_w(i,:), P^{-1})$
5: **end for**
6: **return** $\mathbf{r}_A$

Algorithm 4 `FilterBob`$(\mathbf{h}, m, P)$

Require: $\mathbf{h} \in \mathbb{R}^n$. m is the number of inner product operations required. P^T is a $n \times n$ invertible matrix as described in `InnerProductBob`.

1: **for** $i \leftarrow 1$ to m **do**
2: $\quad \mathbf{r}_B(i) \leftarrow$ `InnerProductBob`$(\mathbf{h}, P)$
3: **end for**
4: **return** $\mathbf{r}_B$

At the end of `FilterAlice` and `FilterBob`, the filtered result $\mathbf{y}$ is decomposed into $\mathbf{r}_A + \mathbf{r}_B$ where Alice has $\mathbf{r}_A$ and Bob has $\mathbf{r}_B$, all in a scrambling order known only to Alice. As discussed earlier, $\mathbf{r}_B$ cannot be sent back to Alice for unscrambling as this will allow Alice to estimate Bob's filter. This creates a problem for multiple stages of linear filtering as Bob does not know the scrambling order of $\mathbf{r}_B$. Multiple stages of linear filtering are often used in image processing such as separable filtering (horizontal and vertical filtering) or wavelet transform (multiple stages of subband filtering). Thus, it is highly desirable to have a secure protocol that can support multiple stages of linear filtering. This is the subject of Section 3.3.

3.3. Three-party Solution

In this section, we show how to implement the secure linear filtering with the help of a third party Clark, who we assume will not collude with either Bob or Alice. This protocol does not use any scrambling and thus allow multiple stages of linear filtering. On the other hand, its application is comparatively limited as a non-colluding third party may not be always present. The basic idea is that, instead of scrambling the rows of X_w, we randomly inject random noise into the rows of X_w and $\mathbf{h}$ for each inner product operation. The dependency between successive rows vanishes as random noise is used each time.

The proposed protocol is shown in `3PartyInnerProductAlice`, `3PartyInnerProductBob`, and `3PartyInnerProductClark`. The problem notation is the same as in Section 3.2 where Alice holds $\mathbf{x}$ and Bob holds $\mathbf{y}$ and they want to jointly compute $\mathbf{x}^T\mathbf{y}$. Alice generates a random n-dimensional vector $\mathbf{x}_a$ and computes $\mathbf{x}_b = \mathbf{x} - \mathbf{x}_a$. Similarly, Bob generate a random n-dimensional vector $\mathbf{h}_a$ and computes $\mathbf{y}_b = \mathbf{y} - \mathbf{y}_a$. Then the inner product can be rewritten as

$$\mathbf{x}^T\mathbf{y} = \mathbf{x}_a^T\mathbf{y}_a + \mathbf{x}_a^T\mathbf{y}_b + \mathbf{x}_b^T\mathbf{y}_a + \mathbf{x}_b^T\mathbf{y}_b, \tag{5}$$

$\mathbf{x}_a$ or $\mathbf{x}_b$ alone provides no information about $\mathbf{x}$. Neither does $\mathbf{y}_a$ or $\mathbf{y}_b$ alone about $\mathbf{y}$. Unfortunately, it is impossible for Bob and Alice to compute all the four terms in (5) by just receiving one component of the vector from each other. For example, if Alice sends Bob $\mathbf{x}_a$, he can compute the first and the second terms $r_b = \mathbf{x}_a^T\mathbf{y}_a + \mathbf{x}_a^T\mathbf{y}_b$. If Bob sends Alice $\mathbf{y}_a$, Alice can compute the third term but not the fourth. If Bob sends $\mathbf{y}_b$, Alice can compute the fourth but not the third. To solve this conundrum, we introduce a third party Clark. If Bob sends Alice $\mathbf{y}_a$ and Alice computes $r_a = \mathbf{x}_b^T\mathbf{y}_a$, Alice and Bob can send Clark $\mathbf{x}_b$ and $\mathbf{y}_b$ so that Clark can compute the remaining term in (5), i.e. $r_c = \mathbf{x}_b^T\mathbf{y}_b$. Provided that no two parties collude with each other, the information Alice, Bob and Clark have are all random data which disclose no information about either $\mathbf{x}$ or $\mathbf{h}$.

Algorithm 5 `3PartyInnerProductAlice`$(\mathbf{x})$

Require: $\mathbf{x} \in \mathbb{R}^n$.

1: Generate random vector $\mathbf{x}_a$.
2: $\mathbf{x}_b \leftarrow (\mathbf{x} - \mathbf{x}_a)$
3: Transmit $\mathbf{x}_a$ to Bob.
4: Transmit $\mathbf{x}_b$ to Clark.
5: Receive $\mathbf{y}_a$ from Bob.
6: **return** $r_a = \mathbf{x_b}^T\mathbf{y_a}$

To use this new protocol for linear filtering, we can simply modify `FilterAlice` and `FilterBob` by removing the scrambling and replacing `InnerProductAlice` with `3PartyInnerProductAlice` and `InnerProductBob` with `3PartyInnerProductBob`. At the end of the protocol, Alice, Bob and Clark will have $\mathbf{r}_A$, $\mathbf{r}_B$ and $\mathbf{r}_C$ respectively such that the output image $y = \mathbf{r}_A + \mathbf{r}_B + \mathbf{r}_C$. To perform a second stage linear filtering with, say Bob's $\mathbf{g}$, we can simply apply the distribution rule for convolution $y * g = \mathbf{r}_A * g + \mathbf{r}_B * g + \mathbf{r}_C * g$ and compute each term using the secure linear filtering protocol between Bob and each of Alice and Clark.

Algorithm 6 `3PartyInnerProductBob`($\mathbf{y}$)

Require: $\mathbf{y} \in \mathbb{R}^n$.

1: Generate random vector $\mathbf{y}_a$.
2: $\mathbf{y}_b \leftarrow (\mathbf{y} - \mathbf{y}_a)$
3: Receive $\mathbf{x}_a$ from Alice.
4: Transmit $\mathbf{y}_a$ to Alice.
5: Transmit $\mathbf{y}_b$ to Clark.
6: **return** $r_b = \mathbf{x}_a^T \mathbf{y}_a + \mathbf{x}_a^T \mathbf{y}_b$

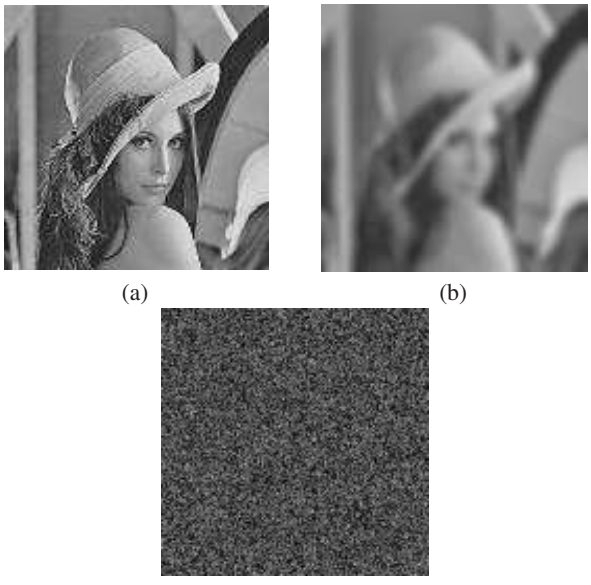
Algorithm 7 `3PartyInnerProductClark`()

1: Receive $\mathbf{x}_b$ from Alice.
2: Receive $\mathbf{y}_b$ from Bob.
3: **return** $r_c = \mathbf{x}_b^T \mathbf{y}_b$

4. EXPERIMENTS

All the algorithms described in Section 3 are implemented and tested with MATLAB version 7.0. To ensure the accuracy of the design, all data exchange between Alice and Bob are conducted using actual TCP/IP connections. To encourage future development and independent verification, we have also made all our source code available at http://www.vis.uky.edu/∼nhu2/secureImage.

Figure 1(a) shows the original 128×128 Lena image. Figure 1(b) shows the result of filtering Lena using a 7×7 Gaussian denoising filter. This image is the result of combining Bob's and Alice's partial copies. We use $P \in \mathbb{R}^{49 \times 49}$ to completely hide the details of the filter and the output of `FilterBob` is shown in Figure 1(c). As shown, all the details from the original image are scrambled.



(a) (b)

(c)

Fig. 1. *(a)* 128×128 *Lena; (b) Lena smoothed by a* 7×7 *Gaussian filter using secure filtering with a disguise matrix* $M \in \mathbb{R}^{49 \times 49}$*; (c) is the partial result held by Bob*

Our protocols are also computationally efficient. As a comparison, we have implemented a classic two-party protocol based on the description from [4], using our own 512-bit RSA public-key cryptosystem (PKCS). We then compare its performance with the algorithm described in Section 3.2 and 3.3 on a dual Wintel CPU (P4-3.4GHz) desktop with 1GB memory. The reason we did not test the classic protocol on real images is because it will take hours to do a linear filtering on a single image. The oblivious transfer based technique takes about 20 minutes to compute the inner product of two 20-dimensional vectors while our two-party protocol uses only 30 *milliseconds* and our three-party protocol uses 47 milliseconds. Despite our non-optimal implementation of the oblivious transfer protocol, its slow performance can be attributed to the handling of very long integers in the encryption/decryption process as well as the large amount of information exchanged between Alice and Bob. For linear filtering using a 7×7 Gaussian mask on the same computing platform, our two-party solution takes on average 0.7 seconds to denoise a 128×128 image and our three-party solution takes around 0.6 seconds.

5. CONCLUSION

This paper presents two different solutions for the secure linear filtering problem — a two-party solution based on rank reduction and random permutation, and a three-party solution based on random perturbation. A drawback of the two-party solution is that the scrambling prevents repeated use of the protocols. To avoid scrambling, our third-party solution relies on a third party to participate in independent random perturbation of each inner product computation. The comparison between our proposed protocols and the classic protocol based on OT shows that our protocols are much faster and more appropriate for pixel-wise computation. A more detailed evaluation of the security of these algorithms, especially under malicious attack, is among the most pressing issue we are currently addressing. Another important area is to assess the bandwidth requirement for these algorithms, which could quickly become a bottleneck as we move from image to video applications..

6. REFERENCES

[1] A. C. Yao, "How to generate and exchange secretes," *proceeding of 27th FOCS*, pp. 162–167, 1986.

[2] O. Goldreich, *Foundations of Cryptography: Volume II Basic Applications*, Cambridge, 2004.

[3] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," *Proc. 12th Ann. Symp. Discrete Algorithms*, pp. 448–457, 2001.

[4] S. Avidan, "Blind vision," *ECCV2006*, May 2006.

[5] W. Du et al., "Privacy-preserving multivariate statistical analysis: Linear regression and classification," *proceeding of the 4th SIAM International Conference on Data Mining*, pp. 222–233, 2004.

[6] A. Lempel S. Even, O. Goldreich, "A randomized protocol for signing contracts," *Communications of the ACM 28*, pp. 637–647, 1985.

[7] M. O. Rabin, "How to exchange secrets by oblivious transfer," *Tech. Memo TR-81, Aiken Computation Laboratory*, 1981.