STUDIA INFORMATICA
Nr 1/2(7) Systemy i technologie informacyjne 2006

# How To Construct Support Vector Machines Without Breaching Privacy

**Justin Zhan[1], LiWu Chang[2], and Stan Matwin[3]**
[1] School of Information Technology and Engineering, University of Ottawa, Canada, zhizhan@site.uottawa.ca
[2] Center for High Assurance Computer Systems, Naval Research Laboratory, USA, lchang@itd.nrl.navy.mil
[3] School of Information Technology and Engineering, University of Ottawa, Canada. Institute for Computer Science, Polish Academy of Sciences, Warsaw, Poland, stan@site.uottawa.ca

**Abstract.** This paper addresses the problem of data sharing among multiple parties in the following scenario: without disclosing their private data to each other, multiple parties, each having a private data set, want to collaboratively construct support vector machines using a linear, polynomial or sigmoid kernel function. To tackle this problem, we develop a secure protocol for multiple parties to conduct the desired computation. In our solution, multiple parties use homomorphic encryption and digital envelope techniques to exchange the data while keeping it private. All the parties are treated symmetrically: they all participate in the encryption and in the computation involved in learning support vector machines.

**Keywords.** Privacy, security, support vector machine

## 1 Introduction

In the modern business world, collaboration becomes especially important because of the mutual benefit it brings. In this paper, we address the following collaboration problem: multiple parties are cooperating on a data mining task. Each of the parties owns data pertinent to the aspect of the task addressed by this party. More specifically, the data consist of instances, all parties have data about all the instances involved, but each party has its own view of the instances - each party works with its own attribute set. The parties may be unwilling to release their attribute values to the other party due to privacy or confidentiality of the data. How can multiple parties structure information sharing between them so that the data will be shared for the purpose of data mining, while at the same time specific attribute values will be kept confidential by the parties to whom they belong? This is the task addressed in this paper. In the privacy-oriented data mining this task is known as data mining with vertically partitioned data [13]. The following scenarios illustrate situations in which this type of collaboration is interesting: (i) Multiple competing

supermarkets, each having an extra large set of data records of its customers' buying behaviors, want to conduct data mining on their joint data set for mutual benefit. Since these companies are competitors in the market, they do not want to disclose too much about their customers' information to each other, but they know the results obtained from this collaboration could bring them an advantage over other competitors. (ii) Vaidya and Clifton [13] provide the following convincing example in the area of automotive safety: Ford Explorers with Firestone tires from a specific factory had tread separation problems in certain situations. Early identification of the real problem could have avoided at least some of the 800 injuries that occurred in accidents attributed to the faulty tires. Since the tires did not have problems on other vehicles, and other tires on Ford Explorers did not pose a problem, neither side felt responsible. Both manufacturers had their own data, but only early generation of mining results based on all of the data may have enabled Ford and Firestone to collaborate in resolving this safety problem.

This paper studies how to learn support vector machines in the distributed scenario with private attributes as described above. In the last few years, there has been a surge of interest in Support Vector Machines (SVM) [23, 24]. SVM is a powerful methodology for solving problems in nonlinear classification, function estimation and density estimation which has also led to many other recent developments in kernel based learning methods in general [6, 20, 21]. SVMs have been introduced within the context of statistical learning theory and structural risk minimization. As part of the SVM algorithm, one solves convex optimization problems, typically quadratic programs. It has been empirically shown that SVMs have good generalization performance on many applications such as text categorization [12], face detection [16], and handwritten character recognition [14] based on the existing SVM learning technologies, we study the problem of learning Support Vector Machines on private data. More precisely, the problem is defined as follows: multiple parties want to jointly build support vector machines on their data set, but none of the parties is willing to disclose her actual data to each other or any other parties. The dataset is vertically partitioned in that all parties have data about all the instances involved, but each party has its own view of the instances - each party works with its own attribute set. We develop a secure protocol, based on homomorphic encryption and digital envelope techniques, to tackle the problem. An important feature of our approach is its distributed character, i.e. there is no single, centralized authority that all parties need to trust. Instead, the computation is distributed among parties, and the use of homomorphic encryption and digital envelope techniques ensures privacy of the data.

The paper is organized as follows: The related work is discussed in Section 2. We describe the SVMs training procedure in Section 3. We then present our proposed secure protocols in Section 4. We give our conclusion in Section 5.

# 2 Related Work

## 2.1 Secure Multi-Party Computation

A Secure Multi-party Computation (SMC) problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output. The SMC problem literature was introduced by Yao [26]. It has been proved that for any polynomial function, there is a secure multi-party computation solution [11]. The approach used is as follows: the function $F$ to be computed is firstly represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets corresponding shares of the input wires and the output wires for every gate. This approach, though appealing in its generality and simplicity, is highly impractical for large datasets.

## 2.2 Privacy-Preserving Data Mining

In early work on privacy-preserving data mining, Lindell and Pinkas [15] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by secure multi-party computation (SMC) research [11, 26]. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [13]. Randomization approaches were firstly proposed by Agrawal and Srikant in [2] to solve privacy-preserving data mining problem. Researchers proposed more random perturbation-based techniques to tackle the problems (e.g., [3, 8, 19]). In addition to perturbation, aggregation of data values [22] provides another alternative to mask the actual data values. In [1], authors studied the problem of computing the kth-ranked element. Dwork and Nissim [9] showed how to learn certain types of boolean functions from statistical databases in terms of a measure of probability difference with respect to probabilistic implication, where data are perturbed with noise for the release of statistics. In [25], Wright and Yang applied homomorphic encryption [17] to the Bayesian networks induction for the case of two parties. In [10], Goethals et.al. deal with secure scalar product computation for privacy-preserving data mining. In this paper, we develop a secure protocol based on homomorphic encryption and digital envelope techniques to learn support vector machines.

# 3 Learning SVMs On Private Data

Support vector machines were invented by Vapnik [24] in 1982. The idea consists of mapping the space of input examples into a high-dimensional feature space, so that the optimal separating hyperplane built on this space allows a good generalization capacity. The input examples become linearly or almost linearly

separable in the high dimensional space through selecting an adequate mapping [23]. Research on SVMs is extensive since it was invented. However, to our best knowledge, there is no effort on learning SVMs on private data. In this paper, our goal is to provide a privacy-preserving algorithm for multiple parties to collaboratively learn SVMs without compromising their data privacy.

### 3.1 Notations

We define the following notations for illustration purposes.

- n: the total number of parties. We assume $n \geq 3$.
- $P_j$: Party j.
- $m_i$: the total number of attributes of $P_i$ for $i \in [1, n]$.
- m: the total number of attributes.

### 3.2 Overview of Support Vector Machine

SVM is primarily a two-class classifier for which the optimization criterion is the width of the margin between the different classes. In the linear form, the formula for output of a SVM is

$$u = \vec{w} \cdot \vec{x} + b, \qquad (1)$$

where $\vec{w}$ is the normal vector to the hyperplane and $\vec{x}$ is the input vector. To maximize margin, we need minimize the following [4]:

$$\min_{w,b} \tfrac{1}{2} \|\vec{w}\|^2, \qquad (2)$$

subject to $y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1, \ \forall i$, where $\vec{x_i}$ is the *ith* training example, and $y_i$ is the correct output of the SVM for the *ith* training example. The value $y_i$ is +1 (resp. -1) for the positive (resp. negative) examples in a class.

Through introducing Lagrangian multipliers, the above optimization can be converted into a dual quadratic optimization problem.

$$\min_{\vec{\alpha}} \psi(\vec{\alpha}) = \min_{\alpha_i, \alpha_j} \tfrac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\vec{\alpha_i}, \vec{\alpha_j}) - \sum_{i=1}^{N} \alpha_i, \qquad (3)$$

where $\alpha_i$s are the Lagrange multipliers, $\vec{\alpha} = \alpha_1, \alpha_2, \cdots, \alpha_N$, subject to inequality constraints: $\alpha_i \geq 0, \forall i,$ and linear equality constraint: $\sum_{i=1}^{N} y_i \alpha_i = 0$.

By solving the dual optimization problem, one obtains the coefficients $\alpha_i, i = 1,2,\cdots,N$ from which the normal vector $\vec{w}$ and the threshold b can be derived [18].

To deal with non-linearly separable data in feature space, Cortes and Vapnik [5] introduced slack-variables to relax the hard-margin constraints. The modification is:

$$\min \tfrac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i \qquad (4)$$

subject to $y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_{i,} \forall i,$ where $\xi_i$ is a slack variable that allows margin failure and constant C > 0 determines the trade-off between the empirical error and the complexity term. This leads to dual quadratic problem involving Eq.(3) subject to the constraints $C \geq \alpha_i \geq 0, \forall i,$ and $\sum_{i=1}^{N} y_i\alpha_i = 0$.

To solve the dual quadratic problem, we apply sequential minimal optimization [18] which is a very efficient algorithm for training SVMs.

### 3.3 Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) [18] is a simple algorithm that can efficiently solve the SVM quadratic optimization (QO) problem. Instead of directly tackling the QO problem, it decomposes the overall QO problem into QO sub-problems based on Osunna's convergence theorem [16]. At each step, SMO chooses two Lagarange multipliers to jointly optimize, find the optimal values for these multipliers, and updates the SVM to reflect the new optimal values.

In order to solve for the two Lagrange multipliers, SMO firstly computes the constraints on these multipliers and then solves for the constrained minimum.
Normally, the objective function is positive definite, SMO computes the minimum along the direction of the linear constraints $\sum_{i=1}^{2} y_i\alpha_i = 0$ within the boundary $C \geq \alpha_i \geq 0, i = 1,2.$

$$\alpha_2^{New} = \alpha_2 + y_2(E_1 - E_2)\eta, \qquad (5)$$

where $E_i = y_i\alpha_i K(\vec{x}_i, \vec{x}) - y_i$ is the error on the $ith$ training example, $\vec{x}_i$ is the stored training vector and $\vec{x}$ is the input vector, and $\eta$ is the second derivative of Eq.(3) along the direction of the above linear constraints:

$$\eta = K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_2}, \overrightarrow{x_2}) - 2K(\overrightarrow{x_1}, \overrightarrow{x_2}). \qquad (6)$$

Next step, the constrained minimum is found by clipping the unconstrained minimum to the ends of the line segment: $\alpha_2^{new,clipped}$ is equal to H if $\alpha_2^{new} \geq H$, is equal to $\alpha_2^{new}$ if $L < \alpha_2^{new} < H$, and is equal to $\alpha_2^{n,clippedew} = L$ if $\alpha_2^{new} \leq L$. If the target $y_1$ is not equal to the target $y_2$, $L = \max(0, \alpha_2 - \alpha_1), H = \min(C, C + \alpha_2 - \alpha_1)$. If the target $y_1$ is equal to the target $y_2$, $L = \max(0, \alpha_2 + \alpha_1 - C), H = \min(C, \alpha_2 + \alpha_1)$.

The value of $\alpha_1$ is computed from the new, clipped, $\alpha_2$:

$$\alpha_1^{new} = \alpha_1 + s(\alpha_2 - \alpha_2^{new,clipped}), \qquad (7)$$

where $s = y_1 y_2$.

In the procedure of sequential minimal optimization, the only step accessing the actual attribute values is the computation of the kernel function K. Kernel functions have various forms. Three types of kernel functions are considered here: they are the linear kernel function $K = (\vec{a}, \vec{b})$, the polynomial kernel function $K = ((\vec{a}, \vec{b}) + \theta)$, where $d \in N, \theta \in R$ are constants, and the sigmoid kernel function $K = \tanh((\kappa(\vec{a}, \vec{b})) + \theta)$, where $\kappa, \theta \in R$ are constants, for instances $\vec{a}$ and $\vec{b}$.

To compute these types of kernel functions, the only computation involving private data is to compute the inner product between two instances. Since each party has partial attribute values, each of them can only compute partial inner product. The challenge is how to combine these partial inner products without disclosing each party's private data. Suppose that $P_1, P_2, \cdots,$ and $P_n$ get the partial inner products denoted by $v_1, v_2, \cdots,$ and $v_n$ respectively. The goal is to compute $\sum_{j=1}^{n} v_j$ without compromising data privacy. To achieve this goal, a secure protocol is developed in next section.

# 4 A Secure Protocol

## 4.1 Homomorphic Encryption and Digital Envelope

In our secure protocols, we use homomorphic encryption [17] keys to encrypt the parties' private data. In particular, we utilize the following characterizer of the homomorphic encryption functions: $e(a_1) \times e(a_2) = e(a_1 + a_2)$ where e is an encryption function; $a_1$ and $a_2$ are the data to be encrypted. Because of the property

of associativity, $e(a_1 + a_2 + \cdots\_ a_n)$ can be computed as $e(a_1) \times e(a_2) \times \cdots \times e(a_n)$ where $e(a_i) \neq 0$. That is

$$e(a_1 + a_2 + \cdots + a_n) = e(a_1) \times e(a_2) \times \cdots \times e(a_n) \quad (8)$$

*Digital envelope* A digital envelope is a random number (or a set of random numbers) only known by the owner of private data. To hide the private data in a digital envelope, we conduct a set of mathematical operations between a random number (or a set of random numbers) and the private data. The mathematical operations could be addition, subtraction, multiplication, etc. For example, assume the private data value is a. There is a random number R which is only known the owner of a. The owner can hide a by adding this random number, e.g., a+R.

### 4.2 Description of Protocol

Let's assume that there are two instance vectors, $\vec{x_1}$ and $\vec{x_2}$, which contain $m = m_1 + m_2 + \cdots + m_n$ number of attributes. $P_1$ has the attribute values of the first $m_1$ attributes, and $P_2$ has the attribute values of the second $m_2$ attributes, $\cdots, P_n$ has the attribute values of the last $m_n$ attributes. We use $x_{1i}$ to denote the *ith* element in vector $\vec{x_1}$, and $x_{2i}$ to denote the *ith* element in vector $\vec{x_2}$. In order to compute the $K(\vec{x_1}, \vec{x_2})$, the key issue is how the multiple parties compute the inner product between $\vec{x_1}$ and $\vec{x_2}$ without disclosing them to each other. Before applying our secure protocol, $P_1$ computes $\sum_{i=1}^{m_1} x_{1i} \cdot x_{2i}$ and gets a count $v_1$, $P_2$ computes $\sum_{i=m_1+1}^{m_1+m_2} x_{1i} \cdot x_{2i}$ and gets a count $v_2$, $\cdots, P_n$ computes $\sum_{i=m-m_n+1}^{m} x_{1i} \cdot x_{2i}$ and gets a count $v_n$. The goal is to securely compute $\sum_{j=1}^{n} v_j = v_1 + v_2 + \cdots + v_n$.

In our protocol, there is no single centralized authority that all parties need to trust. Instead, the computation is distributed among parties. There are four steps. In Step I, multiple parties randomly select a party as the key generator. Let's assume that $P_n$ is selected. Each party generates a digital envelope and $P_n$ also generates a cryptographic key pair (e, d). In Step II, $P_{n-1}$ computes $e(\sum_{j=1}^{n}(v_j + r_j))$ where $r_j$ is a digital envelope (See below for details). $P_{n-1}$ uses the property of homomorphic encryptions to combine the data received from other parties whose private data are

securely protected. In Step III, $P_1$ computes $e(-\sum_{j=1}^{n} v_j)$. $P_1$ also applies the property of homomorphic encryptions to combine the encrypted data from other parties. In Step IV, $P_1$ and $P_{n-1}$ compute $e(\sum_{j=1}^{n} v_j)$, then send it to $P_n$. $P_n$ computes $\sum_{j=1}^{n} v_j$ that is the desired output.

We describe this more formally as follows:

**Protocol 1.** *Secure Multi-Party Protocol*
INPUT: $P_1$'s input is a count $v_1$, $P_2$'s input is a count $v_2$, $\cdots$, $P_n$'s input is a count $v_n$. The counts are taken from the real number domain.
OUTPUT: $\sum_{i=1}^{n} v_i$.

**Step I:** Key and digital envelope generation.
1. $P_j$s for $j \in [1, n]$ randomly select a key generator, e.g., $P_n$.
2. $P_n$ generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. Let e(.) denote encryption and d(.) denote decryption.
3. Each party independently generates a digital envelope, i.e., $P_j$ generates a digital envelope $r_j$, for $j \in [1, n]$.

**Step II:** Computing $e(\sum_{j=1}^{n}(v_j + r_j))$.
1. $P_1$ computes $e(v_1 + r_1)$, and sends it to $P_2$.
2. $P_2$ computes $e(v_1 + r_1) \times e(v_2 + r_2) = e(v_1 + v_2 + r_1 + r_2)$, and sends it to $P_3$.
3. Repeat steps 1, 2 until $P_{n-1}$ obtains
$e(v_1 + v_2 + \cdots + v_{n-2} + r_1 + r_2 + \cdots + r_{n-2}) \times e(v_{n-1} + r_{n-1})$
$= e(v_1 + v_2 + \cdots + v_{n-1} + r_1 + r_2 + \cdots + r_{n-1})$.
4. $P_n$ computes $e(v_n + r_n)$, and sends it to $P_{n-1}$.
5. $P_{n-1}$ computes $e(v_1 + \cdots + v_{n-1} + r_1 + \cdots + r_{n-1}) \times e(v_n + r_n)$
$= e(v_1 + \cdots + v_n + r_1 + \cdots + r_n) = e(\sum_{j=1}^{n}(v_j + r_j))$. Let us denote it by e(V + R), where
$V = \sum_{j=1}^{n} v_j$ and $R = \sum_{j=1}^{n} r_j$.

**Step III:** Computing $e(-\sum_{j=1}^{n} r_j)$.

1. $P_n$ computes $e(-r_n)$, and sends it to $P_{n-1}$.
2. $P_{n-1}$ computes $e(-r_n)\times e(-r_{n-1}) = e(-r_n - r_{n-1})$, and sends it to $P_{n-2}$.
3. Repeat steps 1, 2 until $P_1$ obtains $e(-R) = e(-r_1 - r_2 - \cdots - r_n) = e(-\sum_{j=1}^{n} r_j)$.

**Step IV:** Computing $e(\sum_{j=1}^{n} v_j)$.

1. $P_1$ sends $e(-\sum_{j=1}^{n} r_j)$ to $P_{n-1}$.
2. $P_{n-1}$ computes $e(\sum_{j=1}^{n}(v_j + r_j))\times e(-\sum_{j=1}^{n} r_j) = e(\sum_{j=1}^{n} v_j)$, then sends it to $P_n$.
3. $P_n$ computes $d(e(\sum_{j=1}^{n} v_j)) = \sum_{j=1}^{n} v_j$.

In the next section, we show that the outputs of the protocols are correct, we argue that the data privacy is preserved, and we analyze the complexity for each protocol.

### 4.3 The Analysis of Correctness, Privacy and Complexity

*Correctness Analysis* Assuming all of the parties follow the protocol, to show $\sum_{i=1}^{n} v_i$ is correctly computed, we need to discuss it step by step. In Step II, what $P_{n-1}$ obtains is $e(v_1 + r_1)\times e(v_2 + r_2)\times\cdots\times e(v_n + r_n)$ which equals to $e(\sum_{j=1}^{n}(v_j + r_j))$ according to Eq.(8). In Step III, $P_1$ obtains $e(-r_1 - \cdots - r_n) = e(-\sum_{j=1}^{n} r_j)$ consistent with Eq.(8). In Step IV, $P_n$ finally gets

$d(e(\sum_{j=1}^{n} v_j + \sum_{j=1}^{n} r_j))\times e(-\sum_{j=1}^{n} r_j)) = d(e(\sum_{j=1}^{n} v_j + \sum_{j=1}^{n} r_j - \sum_{j=1}^{n} r_j)) = d(e(\sum_{j=1}^{n} v_j)) = \sum_{j=1}^{n} v_j$. This is the desired result that multiple parties want to obtain.

*Privacy Analysis* There are two levels of privacy protection. One is that the actual count of each party is hidden by a digital envelope, e.g, $r_i$; the other is the protection by semantically secure encryptions. Before any party sends anything related to their actual counts, the counts are concealed by this two-leveled protector. For example, prior to $P_1$ sending values related to $v_1$ to $P_2$, he computes $e(v_1 + r_1)$. Instead of

sending $v_2$ to $P_3$, $P_2$ sends $e(v_1+v_2+r_1+r_2)$, etc. Since $P_1$, $P_{n-1}$ and $P_n$ play more important role than others, e.g., Step IV only involves these three parties, we provide more analysis for these three parties. (1) In Step II, $P_{n-1}$ gets $e(\sum_{j=1}^{n}(v_j+r_j))$. Because each $v_j$ is protected by a digital envelope $r_j$, and the summation of each count with a digital envelope is encrypted by a semantic secure encryption, $P_{n-1}$ cannot learn anything about each $v_j$ for $j\in[1,2,\cdots,n-2,n]$. (2) In Step III, $P_1$ obtains $e(-\sum_{j=1}^{n}r_j)$. Since it is the summation of all the digital envelopes and is encrypted by e, she cannot know anything about each digital envelope $r_j$ for $j\in[2,n]$. (3) $P_n$ finally obtains $\sum_{j=1}^{n}v_j$ which is the desired output of the protocol. It will be shared by all the parties. From the above analysis, we can see that the protocol discloses nothing about each private count.

*Complexity Analysis* The communication cost of this protocol is $\alpha(3n-1)$ where $\alpha$ is the number of bits for each encrypted element, and n is the total number of parties. The computational costs are contributed by: (1) the generation of n digital envelopes; (2) n additions; (3) 2n-1 multiplications; (4) 2n encryptions; (5) 1 decryption. Thus, the total computational costs are 6n.

## 5 Conclusion and Future Work

In this paper, we consider the problem of collaboratively learning Support Vector Machines, by using linear, polynomial or sigmoid kernel functions, on private data. We develop a secure collaborative protocol using homomorphic encryption and digital envelope techniques. In our protocol, the parties do not need to send all their data to a central, trusted party. Instead, we use the homomorphic encryption and digital envelope techniques to conduct the computations across the parties without compromising their data privacy. Privacy analysis of our protocol is provided. Correctness of our protocol is shown and complexity of the
protocol is addressed as well. As future work, we will develop secure protocols for the cases where other kernel functions are applied.

## References

1. Aggarwal G., Mishra N., and Pinkas B. *Secure computation of the kth-rankerd element*. In EUROCRYPT, 40-55, 2005.

2. Agrawal R. and Srikant R. *Privacy-preserving data mining*. In Proceedings of the ACM SIGMOD Conference on Management of Data, 439-450, ACM Press, May 2000.
3. Gehrke J.E., Evfimievshi A., and Srikant R. *Limiting privacy breaches in privacy preserving data mining*. In Proceedings of the 22nd ACM SIGMOD Symposium on Principles of Database Systems, San Diego, CA, June 2003.
4. Burges C. *A tutorial on support vector machines for pattern recognition*. Data Mining and Knowledge Discovery, 2(2):121-167, 1998.
5. Cortes C, and Vapnik V. *Support vector networks*. Machine Learning, 20(3):273-297, 1995.
6. Shawe-Taylor J., and Cristianini N. *An introduction to support vector machines*. In Cambridge University Press.
7. Domingo-Ferrer J. *A provably secure additive and multiplicative privacy homomorphism*. In Information Security Conference, 471-483, 2002.
8. Du W., and Zhan Z. *Using randomized response techniques for privacy-preserving data mining*. In Proceedings of The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27, 2003.
9. Dwork C., and Nissim K. *Privacy-preserving data mining on vertically partitioned databases*. In CRYPTO 2004, 528-544.
10. Goethals B, Laur S., Lipmaa H., and Mielikainen T. *On secure scalar product computation for privacy-preserving data mining*. In Proceedings of The 7th Annual International Conference of Information Security and Cryptology, volume 3506 of Lecture Notes in Computer Science, 104-120, Seoul, Korea, December 2-3, 2004, Springer-Verlag.
11. Goldreich O. *Secure multi-party computation (working draft)*. http://www.wisdom,weizmann.ac.il/home/oded/public_html/foc.html, 1998.
12. Joachims T. *Text categorization with support vector machines: learning with many relevant features*. In Proceedings of 10th European Conference on Machine Learning, number 1398, pages 137-142, Chemnitz, DE, 1998.
13. Vaidya J., and Clifton C. *Privacy preserving association rule mining in vertically partitioned data*. In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada.
14. LeCun Y., Botou L., Jackel L., Drucker H., Cortes C., Denker J., Guyon I., Muller U., Sackinger E., Simard P., and Vapnik V. *Learning algorithms for classification: A comparison on handwritten digit recognition*, 1995.
15. Lindell Y., and Pinkas B. *Privacy preserving data mining*. In Advances in Cryptology – Crypto2000, Lecture Notes in Computer Science, volume 1880, 2000.

16. Freund R., Girosi F., and Osuna e. *Training support vector machines: An application to face detection*. In Proceedings of Computer Vision and Pattern Recognition, 130-136.
17. Paillier P. *Public-key cryptosystems based on composite degree residuosity classes*. In Advances in Cyrptography – EUROCRYPT99, 223-238, Prague, Czech Republic, May 1999.
18. Platt J. *Sequential minimal optimization: A fast algorithm for training support vector machines*. Technical Report, MST-TR-98-14, Microsoft Research, 1998.
19. Rizvi S., and Haritsa J. *Maintaining data privacy in association rule mining*. In Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.
20. Schlkopf B., Smola A., and Mller K. *Nonlinear component analysis as a kernel eigenvalue problem*. Neural Computation, 10(5):1299--1319, 1998.
21. Smola A., Schlkopf B., and Burge C. *Advances in kernel methods – support vector learning*. In MIT Press.
22. Sweeney L. *k-anonymity: a model for protecting privacy*. In International Journal on Uncertainty, fuzziness and Knowledge-based Systems, 10(5), 557-570, 2002.
23. Vapnik V. *The nature of statistical learning theory*. In Springer-Verlag, New York, 1995.
24. Vapnik V. *Estimation of dependences based on empirical data*. In Springer-Verlag, New York, 1982.
25. Wright R., and Yang Z. *Privacy-preserving Bayesian network structure computation on distributed heterogeneous data*. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.
26. Yao A. *Protocols for secure computations*. In Proceedings of the 23rd Annual IEEE Symposium on foundations of Computer Science, 1982.