

# Privacy Preserving Learning in Negotiation

Sheng Zhang\* and Fillia Makedon  
Department of Computer Science, Dartmouth College  
Hanover, NH 03755, USA  
{clap, makedon}@cs.dartmouth.edu

## ABSTRACT

Machine learning techniques are widely used in negotiation systems. To get more accurate and satisfactory learning results, negotiation parties have the desire to employ learning techniques on the union of their past negotiation records. However, negotiation records are usually confidential and private, and owners may not want to reveal the details of these records. In this paper, we introduce a privacy preserving negotiation learning scheme that incorporate secure multiparty computation techniques into negotiation learning algorithms to allow negotiation parties to securely complete the learning process on a union of distributed data sets. As an example, a detailed solution for secure negotiation Q-learning is presented based on two secure multiparty computations: weighted mean and maximum. We also introduce a novel protocol for the secure maximum operation.

## Categories and Subject Descriptors

H.3 [Information Storage and Retrieval]: Information Search and Retrieval

## General Terms

Algorithms, Security

## Keywords

Negotiation, privacy, Q-learning, secure maximum

## 1. INTRODUCTION

Negotiation can be understood as the process toward a final agreement on one or more matters of common interest to different parties. One major obstacle in automated negotiation is incorporating intelligence into a computer system that carries out a negotiation [17], thus enabling the negotiation system to conduct automated negotiations effectively and intelligently on behalf of its clients.

\*Supported by NSF grant IDM-0308229

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA  
Copyright 2005 ACM 1-58113-964-0/05/0003 ...\$5.00.

Machine learning is widely adopted in negotiation learning mechanisms. Negotiation parties use various learning techniques to speculate and update their knowledge about environments and other parties, and these knowledge are good support to their later decisions. The past negotiation records play an important role in all these learning techniques because they compose the training data sets for learning algorithms. As the larger training set probably implies more accurate learning results and more profits, negotiation parties have the desire to obtain more related negotiation records. If several negotiation parties have the same negotiation learning objective and meanwhile they do not have any direct profit conflicts (*e.g.*, a group of buyers all conduct bilateral negotiations with a third party supplier), sharing their negotiation records will bring all of them benefits.

However, negotiation records are usually confidential and private, owners may not want the details of these records revealed to the other negotiation parties. At that time, we wish that there is a secure way to allow all parties to proceed learning algorithms on the union of all negotiation records in a way that each party can still keep his own data secure. We use the word secure to refer that only the final learning results will be known to parties. In this paper, we use *secure multiparty computation* method to solve this problem. Generally speaking, a secure multiparty computation method deals with computing any probabilistic function on any input, in a distributed environment where each party holds one of the inputs, ensuring independence of the inputs, correctness of the computation, and that no more information is revealed to a party in the computation than can be inferred from that party's input and output [8].

The first contribution of this present work is to introduce a privacy preserving negotiation learning scheme to the e-commerce community and give a detailed example of secure Q-learning in negotiation. The second contribution is a novel secure multiparty computation protocol for Maximum operation to allow  $k$  parties ( $k \geq 3$ , each has a value) to find the maximum value without revealing any other information. The organization of this paper is as follows. Section 2 presents our privacy preserving negotiation learning scheme. Section 3 introduces two secure multiparty computations: weighted mean and maximum. Section 4 gives the solution for privacy preserving Q-learning based on these two computations. Section 5 describes the related work of privacy preserving in learning techniques. And finally Section 6 concludes the paper.

## 2. PRIVACY PRESERVING NEGOTIATION LEARNING SCHEME

Learning techniques help negotiation parties obtain more accurate knowledge about other parties and negotiation environments. Zeng and Sycara [22] present an experiment system for updating negotiation offers between two intelligent negotiation agents in bilateral negotiations. They model negotiation as a sequential decision, and use Bayesian probability as the underlying learning mechanism. Another learning approach is to use genetic algorithms and genetic programming [14, 15]. Each negotiation strategy formed by rules is regarded as a gene, and new strategies are generated by means of genetic operations: reproduction, crossover, and activation. Besides supervised learning techniques, reinforcement learning has also been introduced in business markets [16, 4]. In this approach, a business process is considered as a Markov decision process, and players are trying to get a maximum reward.

To allow different negotiation parties to securely conduct the same learning technique on the union of their data sets, the corresponding learning algorithm must satisfy the following two conditions: 1) it is a distributed algorithm that can be conducted by multiple parties on distributed data sets; 2) all communications do not reveal any party's unnecessary information.

Our scheme is to design such an algorithm in three steps.

1. Find a distributed learning algorithm for the specific technique.
2. Divide this algorithm into two kinds of computations: local computations and multiparty computations. Local computations are those computations that can be done by each party without communicating with others, while multiparty computations involve communications among parties.
3. Find a secure multiparty computation protocol for each multiparty computation.

Compared with the first two steps, the last step is more important. As communications are involved in multiparty computations only, the security of the algorithm is decided by the security of all used multiparty computation protocols in it.

There are two important assumptions in this work. First, we are discussing the application in a distributed computing scenario, so each negotiation party keeps his negotiation records locally and there is no central server to gather all records and run learning algorithms. Each party holds a database of different negotiation records and the whole data set is the union of these database (it is denoted as *horizontally partitioned*). Second, we assume that the security model is a semi-honest model [6], which means each party correctly follows the privacy preserving protocols, and he does not use active ways to maliciously attack protocols. However, the party attempts to learn additional information by analyzing the message received during the protocol execution. This model is reasonable here because all parties do not have direct profit conflicts and instead they all have the same learning objective.

In the remaining of this paper, we will give a detailed example on how to conduct privacy preserving Q-learning in negotiation. Basically, we decompose the Q-learning algorithm into two secure multiparty computations: weighted

mean and maximum, and design secure protocols for these two computations.

## 3. SECURE MULTIPARTY COMPUTATIONS

This section will present two secure multiparty computation protocols for weighted mean and maximum computation. As you will find, a basic protocol for secure sum is used in both proposed protocols. Therefore, we first briefly present the secure sum protocol. The protocol details and complete proof can be found in [11].

Assume there are  $k$  ( $k \geq 3$ ) parties and each party  $i$  has an integer value  $v_i$ , they want to compute  $\sum_{i=1}^k v_i$ , and they already know that the sum is less than or equal to  $n$ . In the first step, party 1 generates a random number  $R$  in the range  $[0 \dots n - 1]$ , and sends the result  $R + v_1 \bmod n$  to party 2. Starting from party 2, each party  $i$  receives the result  $R + \sum_{j=1}^{i-1} v_j \bmod n$  from his previous party (party  $i - 1$ ), adds his own value  $v_i$  to this result and passes  $R + \sum_{j=1}^i v_j \bmod n$  to party  $(i + 1 \bmod k)$ . Finally, Party 1 receives  $R + \sum_{i=1}^k v_i \bmod n$  from party  $k$ . He computes  $\sum_{i=1}^k v_i$  by subtracting  $R$  from the received result, and broadcast it to the others.

In this protocol, each party  $i$  receives  $R + \sum_{j=1}^{i-1} v_j \bmod n$ , which is a value uniformly distributed across  $[0 \dots n - 1]$ , thus the party learns nothing more than guess. The protocol assumes that no two parties collude. If party  $i - 1$  and  $i + 1$  reveal the values they send and receive to each other, then they can compute the value party  $i$  has. One way to extend the protocol to prevent this kind of collusion is that each party  $i$  divides his local value  $v_i$  into shares [11]. The sum for each share is computed individually by using a different sequence of parties to ensure that no party has the same neighbor twice. The communication cost for this protocol is  $O(\log n)$  bits for each party and  $O(\log n \cdot k)$  bits for all parties.

### 3.1 Secure Weighted Mean

Now assume each party  $i$  has a value  $v_i$  and a support value  $w_i$ . The weighted mean is to compute  $\sum_{i=1}^k w_i \cdot v_i / \sum_{i=1}^k w_i$ . A simple way to compute this weighted mean is to respectively calculate the numerator and denominator using the secure sum protocol, then do the division. However, this process reveals the sum result of numerator and denominator, which means that each party can know the exact weight of his value in the final result (*i.e.*,  $w_i / \sum_{j=1}^k w_j$ ). The following protocol using approximate  $\ln(x_1 + x_2)$  protocol reveals nothing except the final result. Kantarcioglu and Vaidya [11] use the similar idea for secure probability computation.

Assume  $s_1 = \sum_{i=1}^k w_i \cdot v_i$  and  $s_2 = \sum_{i=1}^k w_i$ , we can evaluate the division by  $\ln$  operation:

$$s_1/s_2 = \exp(\ln s_1 - \ln s_2).$$

Instead of letting party 1 get the result  $s_1 + r_1 \bmod n$  and  $s_2 + r_2 \bmod n$  from party  $k$  (assume that  $r_1$  and  $r_2$  are two random numbers generated by party 1 in executing the secure sum protocol for computing  $s_1$  and  $s_2$ ), we use the secure approximate  $\ln(x_1 + x_2)$  protocol in [12] to generate two pair of random numbers  $\{t_1, t_k\}$  and  $\{u_1, u_k\}$  ( $t_1$  and  $u_1$  for party 1,  $t_k$  and  $u_k$  for party  $k$ ) such that  $t_1 + t_k = C \cdot \ln(s_1 + r_1 - r_1 \bmod n) \bmod n$ , and  $u_1 + u_k = C \cdot \ln(s_2 + r_2 - r_2 \bmod n) \bmod n$ . Here,  $C$  is a public constant used to

make all elements integer, which is given in [12]. Therefore, party  $k$  can send  $t_k - u_k \bmod n$  to party 1 and party 1 can calculate  $s_1/s_2 = \exp((t_1 - u_1 + t_k - u_k \bmod n)/C)$ .

To prove the security of this protocol, we can divide all communications involved in the protocol into two categories. The first category of communications are involved in the secure sum protocol. Based on its security, parties  $2, \dots, k-1$  can not infer anything except the final weighted mean result. The second category of communications are involved in the secure approximate logarithm protocol. The security of that protocol [12] ensures that  $t_1, t_k, u_1, u_k$  are also uniformly distributed, therefore party 1 and  $k$  can not infer  $s_1, s_2$  either.

The communication cost of the secure logarithm is  $O(\log(n) \cdot t)$  bits ( $t$  is the security parameter in [12]), so the total cost of this protocol is  $O(\log(n) \cdot (t+k))$  bits.

## 3.2 Secure Maximum

Let us first talk about the secure comparison problem. Assume that party A has integer  $x$  and party B has integer  $y$ , secure comparison is to find whether  $x \geq y$  (Yao's millionaire problem [20]) without revealing  $x$  to party B and  $y$  to party A. Ioannidis and Grama [9] give an efficient protocol that is particularly suitable for implementation, and the communication cost is  $O((\log n)^2)$  ( $n$  is the size of the field for values). The details of this protocol will not be presented here, but we will use this result for our later protocol in secure maximum computation.

Now there are  $k$  parties and each party  $i$  has a pairwise different integer value  $x_i$  ( $x_i \neq x_j$  if  $i \neq j$ ). They want to find the maximum. A simple method is to use secure comparison method to let party 1 and party 2 compare  $x_1$  and  $x_2$ , and the winner (the owner of the larger value) then conducts the comparison with party 3. After  $k-1$  comparisons, the maximum will be known. The problem of this method is that although each party's value (except the maximum) is not revealed to other parties, each party obtains some information about the relation between his value with another party's value, moreover, the winner party is finally revealed.

The following protocol ensures that no other information will be revealed except the maximum value. This protocol can be extended to the case in which two or more parties have equal values by taking into account both parties' values and their unique identifications.

Let each party generate a vector with  $2k$  elements. Denote  $V^i$  as the vector that party  $i$  generates, and  $V_j^i$  ( $1 \leq j \leq 2k$ ) as the  $j$ th element in  $V^i$ . Set  $V_j^i = x_i$  ( $j = 1, 3, \dots, 2k-1$ ) and  $V_j^i = -x_i$  ( $j = 2, 4, \dots, 2k$ ).

**Step 1:** party 1 generates a random vector  $R$  ( $2k$  elements) and a random permutation  $\pi$ . The permutation  $\pi$  works on the vector of  $2k$  elements by deciding for any  $1 \leq j \leq k$  whether the  $(2j-1)$ th element and the  $(2j)$ th element will exchange in the vector after permutation. Hence, there are  $2^k$  possible permutations. By using the homomorphic public key system [2] (in which we have  $E(x) \cdot E(y) = E(x+y)$  and  $E$  is the public key), each party  $i$  (except party 1) encrypts his own vector  $V^i$  using his public key (denoted as  $E_i$ ). The encrypted vector is a vector in which each element is the encrypted, so  $E_i(V^i) = \{E_i(V_1^i), E_i(V_2^i), \dots, E_i(V_{2k}^i)\}$ . Party  $i$  sends  $E_i(V^i)$  and  $E_i$  to party 1, and party 1 computes  $E_i(R)$ , and  $E_i(V^i) \cdot E_i(R) = E_i(V^i + R)$ . Then party 1 uses permutation  $\pi$  to get  $\pi(E_i(V^i + R))$ , and sends it back to party  $i$ . By using his own private

key, party  $i$  can get  $\pi(V^i + R)$  without knowing  $\pi$  and  $R$ . Meanwhile, party 1 computes  $\pi(V^1 + R)$ . From now on, we use  $V^i$  to represent the vector  $\pi(V^i + R)$ .

**Step 2:** each party  $i$  (except party 1) starts a series of secure comparisons with all the other parties (including party 1), in which party  $i$  compares  $V_{2j-1}^i$  (and  $V_{2j}^i$ ) with party  $j$ 's  $V_{2j-1}^j$  (and  $V_{2j}^j$ ). Here, they use the asymmetric version of secure comparison, so only party  $i$  obtains the comparison result. Party  $i$  uses a vector  $T^i$  to record all comparison results. He sets  $T_{2j-1}^i = 1$  if he has a bigger value than party  $j$  does and otherwise  $T_{2j-1}^i = 0$  (similarly to  $T_{2j}^i$ ). For two exceptional elements, he sets  $T_{2i-1}^i = T_{2i}^i = 0$ .

**Step 3:** after each party  $i$  (except party 1) computes  $T^i$ , each of them works with party 1 to find whether he is the owner of the maximum in the following way. Party 1 generates a vector  $\hat{T}^i$  in which  $\hat{T}_{2j-1}^i = 1$  and  $\hat{T}_{2j}^i = 0$  (for  $j \neq i$ ), and  $\hat{T}_{2i-1}^i = \hat{T}_{2i}^i = 0$ . Then party 1 uses permutation  $\pi$  on  $\hat{T}^i$  to get  $\pi(\hat{T}^i)$ . Now party 1 and party  $i$  use the commutative encryption scheme (in which we have  $E_1(E_2(u)) = E_2(E_1(u'))$  iff  $u = u'$ ) to tell whether  $\pi(\hat{T}^i) = \pi(T^i)$  (all elements in  $\hat{T}^i$  or  $T^i$  are combined to be a single value for simplicity). Here they also use the asymmetric version, so only party  $i$  knows the comparison result. If party  $i$  finds  $\pi(\hat{T}^i) = \pi(T^i)$ , then he knows that he is the owner of the maximum. After party 1 completes  $k-1$  such comparisons with all the other parties, either there is a party  $i$  ( $i \neq 1$ ) who knows that he is the winner or none of them is the winner (which implies that party 1 is the winner, but remember at this time party 1 has not known it yet).

**Step 4:** now the winner needs to broadcast his maximum. This is accomplished by letting all parties conduct the secure sum computation protocol twice. While in the second time, the winner does not add his value and just forwards the result he receives to the next party. By the end of the second secure sum computation, if party 1 finds that the sum is equal to the result obtained from the first sum computation, he knows that he is the winner and he sends the result (subtracting his own value from the sum) to the other parties. Otherwise, if the sum is less than the previous sum (which implies the winner already skips adding his value), party 1 just forwards this sum to the other parties. So all parties get the maximum by subtracting the second sum result from the first sum result.

**THEOREM 1.** *The four step protocol presented above securely computes the maximum in the semi-honest model.*

**PROOF.** We prove the security from each party's view of the protocol. Party 1 receives encrypted vectors and public keys from the other parties in step 1, but it is computationally hard for him to compute original values. In step 2 and step 3, secure comparison protocols are asymmetric, so party 1 does not know either relation information between his value and other values or who is the winner. In step 4, the secure sum computation protocol ensures that he does not know who skips adding the value if that happens.

To parties  $2, \dots, k$ , in step 1, the homomorphic public key system ensures that they all get  $\pi(E_i(V^i + R))$  without knowing  $\pi$  and  $R$ . In step 2, when a party  $i$  ( $i \neq 1$ ) compares his  $2j-1$ th and  $2j$ th elements with another party  $j$ , he always gets the result in which one value is larger and one value is smaller because either  $x_i > x_j$  (which implies  $x_i + r > x_j + r$  and  $-x_i + r < -x_j + r$ ) or  $x_i < x_j$  (which implies  $x_i + r < x_j + r$  and  $-x_i + r > -x_j + r$ ). However, party

$i$  can not tell whether  $x_i > x_j$  as the  $(2j - 1)$ th and  $(2j)$ th elements in the original vector  $V^i$  may exchange (with 1/2 possibility) in the vector  $V^{i'}$ . In step 3,  $\hat{T}^i$  generated by party 1 is the result vector  $T^i$  if party  $i$  is the winner and  $V_i$  is not permuted, thus party  $i$  is ensured to know that he is the winner by finding that  $\pi(\hat{T}^i) = \pi(T^i)$ . In step 4, the secure sum algorithm ensures that even if the party  $i$  skips adding his value in the second time's sum computation, the next party who receives the result from party  $i$  can not notice it. If party 1 is the winner, as the second time's final sum result is known to party 1 only, the other parties can not tell whether the sum they receive has the share of party 1 or not.  $\square$

Table 1 shows the communication cost of this protocol in each step. The total cost is  $O((\log n \cdot k)^2)$  bits for all parties and is dominated by the cost of step 2. In comparison, the communication cost of the simplest protocol that needs  $k - 1$  secure comparisons and reveals relations of values is  $O((\log n)^2 \cdot k)$ .

**Table 1: Communication Cost of the Secure Maximum Protocol (unit is the number of bits)**

Step	Party1	Other Party	Total
Step 1	$O(\log n \cdot k^2)$	$O(\log n \cdot k)$	$O(\log n \cdot k^2)$
Step 2	$O((\log n)^2 \cdot k)$	$O((\log n)^2 \cdot k)$	$O((\log n \cdot k)^2)$
Step 3	$O(\log n \cdot k)$	$O(\log n)$	$O(\log n \cdot k)$
Step 4	$O(\log n)$	$O(\log n)$	$O(\log n \cdot k)$

## 4. PRIVACY PRESERVING IN LEARNING

### 4.1 Q-Learning

Q-learning, as a species of reinforcement learning, is a popular learning technique in negotiation. The standard Q-learning formula [4] is

$$Q(a, s) = R(s, a) + \gamma \max_{a'}(Q(a', s')). \quad (1)$$

Here,  $R(s, a)$  is the immediate reward at state  $s$  with action  $a$ ,  $\gamma$  is the discount factor for future rewards, and  $s'$  is the state resulting from taking action  $a$  in state  $s$ . The notion of state represents a state of world (negotiation offers or primitives proposed by two parties from the start of negotiation, the index of the current round), and the possible action is to offer, counteroffer, accept and decline.

There are two kinds of values needed for iteratively computing Q values, one is Q values of final states and one is immediate rewards. In negotiations, Q values of final states in which the negotiation ends successfully or fails are easy to define or compute. Immediate rewards can be computed by some rules like comparing offers in two neighbor states. In some applications, where immediate rewards are not meaningful, we can set them as 0 and set  $\gamma$  to 1, which do not discount future rewards.

Oren Etzioni *et al.* [4] present an idea to define an equivalent class over states, which enables the learning to train on a limited set of observations of the class using a variant of the averaging step described in [13]. If we denote that  $s$  and  $s^*$  are in the same equivalence class by  $s \sim s^*$ , the revised Q-learning formula is:

$$Q(a, s) = Average_{s \sim s^*}(R(s^*, a) + \gamma \max_{a'}(Q(a', s'))). \quad (2)$$

To find a solution for secure Q-learning, we first make a distributed algorithm for it. Denote  $Q_i(a, s)$  as the local Q-values of party  $i$ , and  $Q(a, s)$  as the global Q-values. The formula for computing local Q-values according to (2) is:

$$Q_i(a, s) = Average_{s \sim s^*}(R_i(s^*, a) + \gamma \max_{a'}(Q(a', s'))). \quad (3)$$

Here,  $R_i(s^*, a)$  are obtained from party  $i$ 's negotiation records containing  $(s^*, a)$  (assume that there are  $n_i(s^*, a)$  such records).

Now, the formula for computing global values becomes:

$$Q(a, s) = \begin{cases} \max\{Q_i(a, s)\} & \text{if } s \text{ is the ending state} \\ \frac{\sum_{i=1}^k Q_i(a, s) \cdot n_i(s^*, a)}{\sum_{i=1}^k n_i(s^*, a)} & \text{otherwise} \end{cases} \quad (4)$$

The computation of local Q-values (3) can be conducted locally by each party on his own records. The computation of global Q-values (4) involves two multiparty computations: multiparty maximum for ending states and weighted mean for the other states. Based on two secure protocols presented in section 3, we have the protocol (see algorithm 1) to conduct secure Q-learning.

---

#### Algorithm 1 Secure Q-learning

---

**Require:**  $k$  ( $k \geq 3$ ) parties

- 1: **for** each state equivalent class  $s$  and action  $a$  **do**
  - 2:   **if**  $s$  is the ending state **then**
  - 3:     All parties use the secure maximum protocol to get  $Q(a, s)$ .
  - 4:   **else**
  - 5:     Each party computes  $Q_i(a, s)$  by (3) based on his local records and previously computed global Q values  $Q(a', s')$ .
  - 6:     All parties use the secure weighted mean protocol to compute  $Q(a, s)$ .
  - 7:   **end if**
  - 8: **end for**
- 

**THEOREM 2.** *Algorithm 1 is a secure Q-learning protocol in the semi-honest model.*

**PROOF.** All communications in algorithm 1 are involved either in secure maximum computations or in secure weighted mean computations. The protocols used in these two multiparty computations are proved to be secure in the semi-honest model in section 3. These two protocols do not reveal any other information except the results they generate. Actually these results are global Q values that all parties try to compute, so the whole protocol does not reveal any unnecessary information.  $\square$

If the number of ending states is  $|s_1|$ , and the number of other states is  $|s_2|$ . The communication complexity for one iteration of Q-learning is  $O((\log n \cdot k)^2 \cdot |s_1| + \log n \cdot (t+k) \cdot |s_2|)$ .

## 5. RELATED WORK

Agrawal and Srikant propose a privacy preserving data mining scheme using *Random Data Perturbation* [1]. The idea is adding *noise* data to the original data set before the learning process, however this kind of *noise* data is drawn from some distribution, which can mitigate the noise impact from learning results. Another approach is using *Randomized Responses* techniques that were developed in the

statistic community for the purpose of protecting surveyee's privacy. Evfimievski *et al.* propose an approach to conduct privacy preserving association rules mining based on this technique [5]. The problem of the randomized responses techniques is that the parameter used in randomized response techniques affects the accuracy of learning results.

The last popular approach to achieve privacy preserving learning is to use *Secure Multiparty Computation* techniques. Yao first suggests a general secure two-party function evaluation technique [21]. Goldreich *et al.* extend this to any multiparty function [7, 6]. This generic method is based on representing each function as a boolean circuit, and then the parties run a protocol for every gate in the circuit. However, as Goldreich points in [6], the communication complexity of this generic method depends on the size of the circuit that expresses the function to be computed, and using the solutions derived by this generic method for special cases of multiparty computation can be impractical. Therefore, recent research work is focused on developing efficient techniques for special cases. These include efficient privacy preserving techniques in secure cooperative statistical analysis [3], naive bayes classifier [11], association rules data mining [10, 18], decision tree [12] and clustering [19].

## 6. CONCLUSION

In this work, we show a scheme to achieve privacy preserving negotiation learning on the union of distributed negotiation history through secure multiparty computations. As an example, the detailed solution for secure Q-learning is introduced based on secure weighted mean and secure maximum. This kind of privacy preserving learning in negotiation is important and useful because parties can get more reasonable and accurate knowledge in various negotiation environments by borrowing experience from other parties (even if environments are strange to them).

Our future work includes designing secure protocols (with lower communication complexity) for other learning techniques that are used in negotiation systems, and extending the current work to resist some degree of collusion and malicious attacks.

## 7. REFERENCES

- [1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2000.
- [2] W. Du and M. J. Atallah. Protocols for secure remote database access with approximate matching. In *the First Workshop on Security and Privacy in E-Commerce, the seventh ACM Conference on Computer and Communications Security*, 2000.
- [3] W. Du and M. J. Atallah. Privacy-preserving statistical analysis. In *Proceedings of the 17th Computer Security Applications Conference*, 2001.
- [4] O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price. In *Proceedings of the Ninth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2003.
- [5] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.
- [6] O. Goldreich. Secure multi-party computation. Working Draft, 2000.
- [7] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th ACM conference on Theory of computing*, 1987.
- [8] S. Goldwasser. Multi party computations: past and present. In *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing*, 1997.
- [9] I. Ioannidis and A. Grama. An efficient protocol for yao's millionaires' problem. In *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003.
- [10] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 2002.
- [11] M. Kantarcioglu and J. Vaidya. Privacy preserving naive bayes classifier for horizontally partitioned data. In *the IEEE ICDM Workshop on Privacy Preserving Data Mining*, 2003.
- [12] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.
- [13] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1-3):159–195, 1996.
- [14] S. Matwin, T. Szapiro, and K. Haigh. Genetic algorithms approach to a negotiation support system. *IEEE transaction on System, Man, and Cybernetics*, 21(1):102–114, 1991.
- [15] J. Oliver. A machine learning approach to automated negotiation and prospects for electronic commerce. <http://opim.wharton.upenn.edu/oliver27/papers/jmis.ps> December, 1997.
- [16] C. Raju, Y. Narahari, and K. Ravikumar. Reinforcement learning applications in dynamic pricing of retail markets. In *Proceedings of the IEEE International Conference on E-Commerce*, pages 339–346, 2003.
- [17] S. Y. Su, C. Huang, J. Hammer, Y. Huang, H. Li, L. Wang, Y. Liu, C. Pluempitiwiriyawej, M. Lee, and H. Lam. An internet-based negotiation server for e-commerce. *VLDB*, 10(1):72–90, 2001.
- [18] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.
- [19] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the Ninth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2003.
- [20] A. Yao. Protocols for secure computation. In *Proceeding of 23rd IEEE Symposium on Foundations of Computer Science*, 1982.
- [21] A. Yao. How to generate and exchange secrets. In *Proceeding of 27th IEEE Symposium on Foundations of Computer Science*, 1986.
- [22] D. Zeng and K. Sycara. Bayesian learning in negotiation. *International Journal of Human Computer Systems*, 48(1):125–141, 1998.