Secure Combinatorial Auctions by Dynamic Programming with Polynomial Secret Sharing

Koutarou Suzuki[1] and Makoto Yokoo[2]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation
1-1 Hikari-no-oka, Yokosuka, Kanagawa, 239-0847 Japan
url: info.isl.ntt.co.jp/~koutarou/
e-mail: koutarou@isl.ntt.co.jp
[2] NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
url: www.kecl.ntt.co.jp/csl/ccrg/members/yokoo/
e-mail: yokoo@cslab.kecl.ntt.co.jp

Abstract. Combinatorial auctions have recently attracted the interests of many researchers due to their promising applications such as the spectrum auctions recently held by the FCC. In a combinatorial auction, multiple items with interdependent values are sold simultaneously and bidders are allowed to bid on any combination of items. This paper presents a method for implementing several secure combinatorial auction protocols based on our newly developed secure dynamic programming protocol. Dynamic programming is a very effective, widely used technique for tackling various combinatorial optimization problems, including several types of combinatorial auctions. Our secure dynamic programming protocol utilizes secret sharing techniques and can obtain the optimal solution of a combinatorial optimization problem, i.e., result of a combinatorial auction, without revealing the inputs of the problem, i.e., bidding prices. We discuss the application of the method to several combinatorial auctions, i.e., multiple-unit single-item auctions, linear-goods auctions, and general combinatorial auctions.
key words : dynamic programming, combinatorial auction, spectrum auction, secret sharing

1 Introduction

Internet auctions have become an especially popular part of Electronic Commerce. Some studies on Internet auctions have already been conducted [18, 34], and combinatorial auctions have recently attracted considerable attention [8, 13–15, 26, 27, 36]. In contrast with conventional auctions that sell a single item at a time, combinatorial auctions sell multiple items with interdependent values simultaneously and allow the bidders to bid on any combination of items.

In a combinatorial auction, a bidder can express complementary/substitutional preferences over multiple bids. For example, in the Federal Communications Commission (FCC) spectrum auction [17], a bidder may desire licenses covering adjoining regions simultaneously (i.e., these licenses are complementary), while

being indifferent as to which particular channel was awarded (channels are substitutional). By considering the complementary/substitutional preferences, we can increase the participants' utility and the revenue of the seller. To execute a combinatorial auction, we need to solve a combinatorial optimization problem called the winner-determination problem, i.e., we need to find the combination of bids with disjoint sets of goods, such that the sum of the bidding prices is maximized. The winner determination problem has been tackled using various optimization techniques recently [8, 22, 25, 26].

On the other hand, from the view point of security, to hide bidding prices is an important problem, and there are many researches on secure auction protocols [1, 2, 5–7, 10–12, 19, 20, 23, 24, 29, 30].

If we can trust the auctioneer, we simply gather all private information (i.e., bidding prices) at the auctioneer, who can then solve the problem using any available centralized optimization technique. However, we cannot take it for granted that there exists such a trusted auctioneer. For example, in a standard first-price sealed-bid auction [21], where the highest bidder wins and pays his/her own price, the auctioneer might collude with a particular participant and reveal information of incoming bids to that participant during the auction.

If we use a strategy-proof mechanism, such as a second-price sealed bid (Vickrey) auction [21], where the highest bidder wins and pays the second highest price, the information of other participants' bids becomes useless; thus we can discourage such collusion between the auctioneer and bidders. However, in a second-price sealed-bid auction, if the auctioneer can know the highest bid, he/she can increase his/her revenue by fabricating a fake bid whose price is very close to the highest bid.

We can utilize various cryptographic technologies to ensure that the auctioneer cannot learn bidding prices while accepting incoming bids. However, if the auctioneer can know bidding prices even after the auction ends, he/she can utilize the information of the bids for future auctions. For example, the auctioneer learns the behavior/preference of a certain participant from past auctions and conducts fraudulent activities based on this information, or the auctioneer might reveals/sells such private information to others. Varian [33] pointed out this problem as follows: *"Even if current information can be safeguarded, record of past behavior can be extremely valuable, since historical data can be used to estimate the willingness to pay. What should be the appropriated technological and social safeguards to deal with this problem?".*

This paper aims to provide a solution to this problem: in a combinatorial auction, multiple auction servers can cooperatively solve the winner determination problem, i.e., they can find the combination of bids that maximizes the sum of the bidding prices, while the information of bids that are not part of the optimal solution is kept secret even from the auction servers. More specifically, by utilizing secret sharing techniques [28], we develop a method for securely performing a dynamic programming algorithm [3] that is very effective and widely used against various combinatorial optimization problems. We show also how to

use our secure dynamic programming protocol in various types of combinatorial auctions.

The method proposed in this paper is based on the method proposed for $M+1$-st price auctions by Kikuchi [12]. His method represents the bidding price as the degree of a polynomial, and utilize the fact that the maximum of the degrees of two polynomials can be obtained from the degree of the sum of the two polynomials. In this paper, to implement a secure dynamic programming protocol, we additionally utilize the fact that the sum of the degrees of two polynomials can be obtained from the degree of the product of the two polynomials.

The rest of this paper is organized as follows. In Section 2, we briefly review dynamic programming techniques. In Section 3, we present our newly developed secure dynamic programming protocol. In Section 4, we describe how to apply the proposed method to various types of combinatorial auctions. In Section 5, we discuss its relation to existing techniques.

2 Dynamic Programming

Dynamic programming [3] was developed by R. Bellman during the late 1950's. Dynamic programming is a powerful method that can be applied to various combinatorial optimization problems.

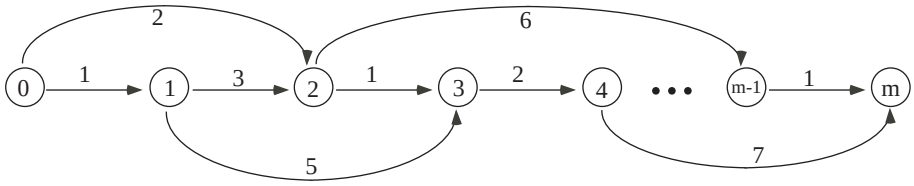

Fig. 1. Example of one-dimension directed graph

In the following, we use the problem of finding the longest path in the one-dimensional directed graph described in Figure 1 to illustrate the concept of dynamic programming. This graph consists of nodes $0, 1, 2, \ldots, m$ with directed links among them. A link is represented as (j, k), where $j < k$. For each link (j, k), the weight $w(j, k)$ of the link is given. The goal is to find the longest path from initial node 0 to terminal node m, i.e., to find a path from 0 to m such that the sum of the weights of the links is maximized. For simplicity, we assume that for each node j (where $0 \leq j < m$), there exists at least one link that starts from j, i.e., there is no dead-end node except m.

One notable characteristic of this problem is as follows. Assume L is the longest path from 0 to m. It follows that for any node j on L, the last half of L, i.e., the part of L from j to m, is also a longest path from j to m. This characteristic is called the *principle of optimality*. This feature enables us to

find the optimal solution of the original problem from the optimal solutions of sub-problems.

More specifically, we can obtain the length of the longest path from 0 to m by solving the following recurrence formula from node $m-1$ to 0. In this formula, $f(j)$ represents the length of the longest path from j to m. We call $f(j)$ the *evaluation value* of node j. For terminal node m, $f(m)$ is defined as 0. For initial node 0, $f(0)$ represents the optimal solution, i.e., the length of the longest path from 0 to m.

$$f(j) = \max_{(j,k)}\{w(j,k) + f(k)\}$$

When calculating this formula, for each node j, we record the link (j,k) that gives the evaluation value $f(j)$, i.e., the link that gives $\max_{(j,k)}\{w(j,k)+f(k)\}$. We can construct the longest path by following these recorded links from 0 to m.

We give below a generalization that is used in Section 4. There are $m+1$ stages $j = 0, 1, \ldots, m$ and states $\{(j,s)\}$ at each stage j. There can be directed links $((j,s),(k,t))$ between these states only if $j < k$, i.e., there are no links at the same stage nor from higher stage to lower stage. For each link, weight $w(((j,s),(k,t)))$ is given. Dynamic programming evaluates function f defined by the following recurrence relation

$$f((j,s)) = \max_{j<k,\ ((j,s),(k,t)):\text{link}}\{w(((j,s),(k,t))) + f((k,t))\}.$$

We can compute value $f((0,s))$, that is the optimal value of the original problem, by iterative application of the relation for $j = m, m-1, \ldots, 0$ with initial values $f((m,s)) = iv(s)$.

In the rest of this paper, we describe our secure dynamic programming protocol based on the longest path finding problem in a one-dimensional directed graph. As discussed in Section 4, the application of the proposed protocol to the general case is also straightforward.

3 Secure Dynamic Programming

The proposed secure dynamic programming protocol is presented below together with a discussion of its security and efficiency.

3.1 Requirements

The requirements for our secure dynamic programming protocol is as follows:

- Each weight publisher sends information of the weight of one link to evaluators.
- Evaluators cooperatively execute dynamic programming and find the optimal solution, while each weight is kept secret.

To realize this protocol, we have to answer the following question: how can we determine the maximum and sum of weights without revealing the weights themselves? The decision on how to represent and encrypt the weight is crucial to making these tasks feasible. Our approach is to represent a weight as the degree of a polynomial; thus the maximum/sum of the degree of two polynomials can be obtained by the degree of the sum/product of the two polynomials.

3.2 Preliminaries

We start by explaining the secret sharing used in our protocol. This secret sharing is based on Shamir's polynomial secret sharing [28], but differs from it at the point that we represent a secret by the degree of a polynomial while Shamir uses the constant term of a polynomial. This kind of polynomial secret sharing is also used for $M+1$-st price auctions by Kikuchi [12].

Weight publisher P who has a secret $s \in \mathbf{Z}_p$ performs secret sharing as follows. Weight publisher P randomly chooses n $(n > s)$ points $x_1, x_2, \ldots, x_n \in \mathbf{Z}_p$ and constant $c \in \mathbf{Z}_p$ and publishes them, and randomly chooses polynomial $A \in \mathbf{Z}_p[x]$ s.t. $deg(A) = s$ and $A(0) = c$ and holds it secret. Weight publisher P then sends l-th share $A(x_l)$ to l-th evaluator E_l through a secure channel.

In this situation, we can examine whether $deg(A) \leq d$ or not while polynomial A is kept secret. First, mask publisher T randomly chooses mask polynomial $M \in \mathbf{Z}_p[x]$ s.t. $deg(M) = d$ and $M(0) = 0$ and keeps it secret, and sends l-th share $M(x_l)$ to l-th evaluator E_l through a secure channel. Next, $d+1$ evaluators $E_1, E_2, \ldots, E_{d+1}$ publish masked shares $A(x_l)+M(x_l)(l = 1, 2, \ldots, d+1)$. Using these $d+1$ masked shares, we perform polynomial interpolation, i.e., determine polynomial $A+M$, recover $A(0) = A(0)+M(0)$, and check whether $A(0) = c$ or not. If $deg(A) \leq d$, we have $deg(A+M) = d$ and can recover the constant term $A(0) = c$ from $d+1$ shares. If $deg(A) > d$, we have $deg(A+M) > d$ and cannot recover the constant term $A(0) = c$ from $d+1$ shares. Hence, if $A(0) = c$ holds, we are convinced that $deg(A) \leq d$

Furthermore, the maximum/sum of the degree of two polynomials can be obtained by the degree of the sum/product of the two polynomials by the following formulae

$$max\{deg(A), deg(B)\} = deg(A+B),\ deg(A) + deg(B) = deg(A \cdot B),$$

respectively. Each evaluator E_l can compute, locally, its share of sum $A+B$ / product $A \cdot B$ of two polynomials A and B by taking sum $A(x_l)+B(x_l)$ / product $A(x_l) \cdot B(x_l)$ of two shares $A(x_l)$ and $B(x_l)$. This allows the maximum/sum of two secrets to be locally determined.

3.3 Secure Dynamic Programming

There is weight publisher $P_{(i,j)}$ for link (i, j), plural e evaluators $E_1, \ldots, E_e$ where e is greater than the length of the longest path, and $t_m + 1$ mask publishers $T_0, \ldots, T_{t_m}$ where t_m is a threshold parameter of mask publishers. In the

following Step 3 and 4, each mask publisher T_l publishes a part of mask M_l, and sum $M = M_0 + \ldots + M_{t_m}$ is used as the mask. Our protocol consists of the following steps:

- Step 1 : Weight publisher $P_{(i,j)}$ of link (i, j) sends to evaluator E_l l-th share of the weight of link (i, j).
- Step 2 : Each evaluator E_l computes, locally, the recurrence relation of dynamic programming, to obtain the l-th share of the optimal value.
- Step 3 : Evaluators obtain the optimal value cooperatively.
- Step 4 : Evaluators trace the links back to obtain the optimal path.

First, mask publisher T_0 randomly chooses e points $x_1, x_2, \ldots, x_e \in \mathbf{Z}_p$ and constant $c \in \mathbf{Z}_p$ and publishes them.

In Step 1, weight publisher $P_{(i,j)}$ decides weight $\tilde{w}(i, j)$, extends weight $w(i, j) = \tilde{w}(i, j) + t_w \times (j - i)$ where t_w is a threshold parameter of weight publishers. By this extension, the optimal solution, i.e., the longest path from node 0 to m will not change. We denote by $\tilde{f}(i)$ / $f(i)$ the evaluation value of node i using the original weights $\tilde{w}$ / the extended weights w. Then, $f(i) = \tilde{f}(i) + t_w \times (m - i)$ holds for each node i. Thus we can find the maximum and perform secure dynamic programming in the same manner as described in the preliminaries with slight modification. Weight publisher $P_{(i,j)}$ then randomly chooses polynomial $W_{(i,j)} \in \mathbf{Z}_p[x]$ s.t. $deg(W_{(i,j)}) = w(i, j)$ and $W_{(i,j)}(0) = c$ and holds it secret. Weight publisher $P_{(i,j)}$ then sends l-th share $W_{(i,j)}(x_l)$ to l-th evaluator E_l through a secure channel.

In Step 2, each evaluator E_l computes, locally, the following formula

$$F_j(x_l) = \sum_{(j,k)} W_{(j,k)}(x_l) \cdot F_k(x_l)$$

for $j = m - 1, m - 2, \ldots, 0$ with $F_m(x_l) = 1$. This formula corresponds to the recurrence relation of dynamic programming

$$f(j) = \max_{(j,k)} \{w(j, k) + f(k)\},$$

thus we have $deg(F_j) = f(j)$.

In Step 3, as described in the preliminaries, we can check whether $deg(F_0) \leq d$ or not. (Notice that we have to check whether $F_0(0)$ is equal to the appropriate constant determined by constant c and the one-dimensional graph or not. For example, if we set $c = 0$, $F_0(0)$ should be 0.) By using this check, we can perform binary search, find the optimal value $f(0) = deg(F_0)$, and publish it.

In Step 4, we obtain the optimal path that attains the optimal value $f(0)$ as follows. Consider that we know $f(j) = deg(F_j)$ and want to find node k s.t. $deg(F_j) = deg(W_{(j,k)} \cdot F_k)$. We examine whether $deg(W_{(j,k)} \cdot F_k) \leq deg(F_j) - 1$ or not for all nodes k linked to node j. Since the inequality holds for node k that does not attains $f(j)$, we are convinced that the node k attains $f(j)$ if the inequality does not hold for node k. After we find the node k that attains $f(j)$, we determine $f(k) = deg(F_k)$ as in Step 3, and publish it. Iterating these process recursively yields the optimal path.

3.4 Example

We give an example for the one-dimension graph shown in Figure 2.

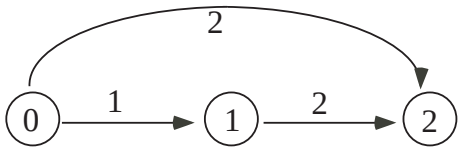

Fig. 2. One-dimension graph

There are three links $(0,1)$, $(1,2)$, $(0,2)$ with weights 1, 2, 2, respectively. Weight publishers $P_{(0,1)}$, $P_{(1,2)}$, $P_{(0,2)}$ for these links generate the following polynomials

$$W_{(0,1)} = x,\ W_{(1,2)} = x^2 - x,\ W_{(0,2)} = 2x^2 + 2x.$$

There are four evaluators E_1, E_2, E_3, E_4, with $x_1 = -2$, $x_2 = -1$, $x_3 = 1$, $x_4 = 2$. For simplicity, we set $t_w = 0$ and $t_m = 1$. We also assume $c = 0$.

Table 1. Shares

	$W_{(0,1)}$	$W_{(1,2)}$	$W_{(0,2)}$	F_1	F_0	M	$W_{(0,1)} \cdot F_1 + M$	$W_{(0,2)} \cdot F_2 + M$
E_1	-2	6	4	6	-8	8	-4	12
E_2	-1	2	0	2	-2	2	0	2
E_3	1	0	4	0	4	2	2	6
E_4	2	2	12	2	16	8	12	20

Table 1 shows the shares of $W_{(0,1)}$, $W_{(1,2)}$, $W_{(0,2)}$, and the shares of F_1 and F_0 as computed by each evaluator in Step 2. From shares $F_0(-2) = -8$, $F_0(-1) = -2$, $F_0(1) = 4$, $F_0(2) = 16$, we can recover $F_0(x) = x^3 + x^2 + 2x$ of degree 3 with constant term $F_0(0) = 0$. This convinces us that $f(0) = 3$.

Table 1 also shows masked shares with $M = 2x^2$ in Step 4. Since the constant term of the polynomial of degree 2 recovered by shares of $W_{(0,1)} \cdot F_1 + M$ is not equal to 0, we are convinced that link $(0,1)$ attains $f(0) = 3$.

3.5 Security

In this section, we discuss the security of our protocol against the passive adversary that can see public information and all information of collusive participants, and tries to find secret information; it cannot manipulate the collusive participants.

Notice that a value at a point of a polynomial that is uniformly randomly chosen is uniformly random. In the protocol, all published shares are masked with random polynomials generated by $t_m + 1$ mask publishers. Hence, if the number of collusive mask publishers is less than or equal to the threshold t_m, masked shares are uniformly random to the adversary and leak no information.

By the extension of weight, extended weight $w(i, j) = deg(W_{(i,j)})$ is more than or equal to t_w. Hence, if the number of collusive evaluators is less than or equal to threshold t_w, the adversary cannot recover polynomial $W_{(i,j)}$ and cannot obtain any information about the weight.

Thus, our protocol is unconditionally secure against the passive adversary.

Theorem 1. *The proposed protocol unconditionally hides all information except the optimal value $f(m)$ and optimal path against any passive adversary with t_m or less collusive mask publishers and t_w or less collusive evaluators.*

Finally, we mention that each weight publisher can publish dummy weights for all links at the lowest value to conceal for which link he publish his weight. In the auction scenario, this means that each bidder can cast dummy bids for all goods at the lowest price to conceal which goods he want to buy.

3.6 Efficiency

Table 2 shows communication pattern, round complexity, and volume per roud through secure channels of each step. We omit communications without secure channels, i.e., evaluators publish shares in step 3 and 4, which can be implemented by a bulletin board. Here, l is the number of links, n is the number of nodes, e is the number of evaluators (which is equal to or greater than possible maximal value), $t_m + 1$ is the number of mask publishers, and p is the order of the finite field $\mathbf{Z}_p$.

Table 2. Communication complexity

	pattern	round	volume
Step 1	$P_{(i,j)} \to E_k$	$l \times e$	$\log p$
Step 2	—	0	0
Step 3	$T_i \to E_k$	$(t_m + 1) \times e \times \log e$	$\log p$
Step 4	$T_i \to E_k$	$(t_m + 1) \times e \times (l + \log e \times n)$	$\log p$

The round complexity is proportional to the number of links or nodes, so our protocol can handle large scale directed graphs (large number of items in auction scenario). However, the round complexity is also proportional to the number of evaluators, so complexity becomes large for wide weight ranges (wide price ranges in auction scenario).

4 Application to Combinatorial Auctions

In this section, we discuss the application of our secure dynamic programming protocol to several types of combinatorial auctions, i.e., multi-unit auctions, linear-goods auctions, and general combinatorial auctions.

4.1 Multi-unit Auctions

In multi-unit auctions, m units of an identical item are auctioned. Each bidder B_k $(1 \leq k \leq N)$ declares his/her bidding price $b_k(j)$ for each quantity j, where $0 \leq j \leq m$. The goal is to find the allocation that maximizes the sum of the bidding prices.

The optimal allocation in a multi-unit auction can be obtained by solving the following recurrence formula [31, 32].

$$f((1,j)) = b_1(j), \quad f((k,s)) = \max_{0 \leq j \leq s} \{f((k-1, s-j)) + b_k(j)\}.$$

In this formula, $f((k,s))$ represents the value of the optimal solution of a subproblem, i.e., optimally allocating s units among k participants, i.e., bidders $B_1, B_2, \ldots, B_k$. The optimal solution is given by $f((N,m))$.

Applying our secure dynamic programming protocol to this problem is rather straightforward.

4.2 Linear-goods Auctions

In a linear-goods auction [31, 22], there exist m goods $G = \{1, 2, \ldots, m\}$. These goods are sequentially ordered. Each bidder $B_k (1 \leq k \leq N)$ bids his/her bidding price $b_k([l_k, u_k])$ for an interval $[l_k, u_k] \subseteq G$ of the goods, i.e., a bidder wants to obtain a continuous sequence of goods. We assume each bidder bids for a single interval (or equivalently bids for several intervals independently). The result of the auction is the allocation of the m goods that maximizes the sum of all bidders' bidding prices.

Auctions for linear-goods can be used for time scheduling (e.g., for the allocation of time slots in a conference room), or for the allocation of a one-dimensional space (e.g., for parts of a seashore), or spectrum right auctions in which each bidder (wireless carrier) wants a continuous frequencies to minimize interference between carriers.

This problem can be directly mapped into the problem of finding the longest path in a one-dimensional graph as follows [16]. We consider the graph with nodes $\{0, 1, 2, \ldots, m\}$, i.e., there exists an initial node 0 and nodes that correspond to goods. Between these nodes, we place links $(l_k - 1, u_k)(1 \leq k \leq N)$ corresponding to bids $b_k([l_k, u_k])(1 \leq k \leq N)$ and dummy links $(j, j+1)(0 \leq j \leq m-1)$ with weights

$$w((l_k - 1, u_k)) = b_k([l_k, u_k]) \ \ (1 \leq k \leq N), \quad w((j-1, j)) = 0 \ \ (1 \leq j \leq m).$$

The allocation of the m goods that maximizes the sum of all bidders' bidding prices corresponds to the longest path from node 0 to node m, and can be securely computed by our secure dynamic programming protocol.

The idea of linear-goods auctions can be generalized to route auctions, in which each bidder bids for a path in a general graph. The result of the auction is the path from the start node to the destination node that maximizes/minimizes the sum of all bidders' bidding prices.

One application of route auctions is transportation task assignment, in which the auctioneer wants to transport his/her cargo from a starting city to a destination city. There are several transportation companies. Each company bids his/her price to carry the cargo for a path (which can be only a part of the total journey), and the auctioneer chooses the combination of paths that minimizes the total cost. This problem can be formalized as finding the shortest path in a graph, and so can be solved using the secure dynamic programming protocol presented in this paper.

4.3 General Combinatorial Auctions

In a general combinatorial auction, there exist multiple different goods $G = \{1, 2, \ldots, m\}$. Each bidder $B_k (1 \leq k \leq N)$ bids his/her price $b_k(S)$ for each subset $S \subseteq G$. The goal is to find the allocation of goods that maximizes the total price.

As described in [22], this problem can be solved by dynamic programming as follows. For each subset $S \subseteq G$, we create a node S. Between these node S, we place N multiple links

$$\{(S, \phi)_k | S \subseteq G, 1 \leq k \leq N\}, \quad \{(S, C)_k | C \subset S \subseteq G, |C| \geq |S|/2, 1 \leq k \leq N\}$$

where ϕ represents the empty set and $|X|$ is the number of elements of X, with weights

$$w((S, \phi)_k) = b_k(S), \quad w((S, C)_k) = b_k(S \setminus C).$$

The optimal allocation is given by the longest path from initial node G to terminal node ϕ. It is clear that this problem can be solved using the secure dynamic programming protocol presented in this paper.

One obvious disadvantage of this approach is that the number of nodes becomes very large, i.e., 2^m. However, this seems somewhat inevitable if we are to solve this problem using dynamic programming, since the winner determination problem of a general combinatorial auction is NP-complete [22].

5 Discussions

There have been various works on secure auction protocols [1, 2, 5–7, 10–12, 19, 20, 23, 24, 29, 30]. However, as far as the authors' know, there has been no other research on secure dynamic programming nor on secure combinatorial auction based on dynamic programming techniques. Here, we discuss some related works.

Kikuchi [12] developed a secure $M+1$-st price auction protocol by using secret sharing. However, in his auction, multiple units of an identical goods are auctioned, but each participant is assumed to want only one unit. Our method can be applied to the cases where each participant wants to buy multiple units.

It is well known that any combinatorial circuit can be computed securely using general-purpose multi-party protocols [4, 9]. Therefore, if we can construct a combinatorial circuit that implements a dynamic programming algorithm, in principle, such an algorithm can be executed securely. However, to execute such a general-purpose multi-party protocol, for each computation of an AND gate in the circuit, evaluators must communicate with each other. Using such a general purpose multi-party protocol for large-scale dynamic programming applications is not practical due to the required communication costs.

Naor et al. [19] proposed a general method for executing any auction protocol, including combinatorial auction protocols, by using a technique called garbled circuit [35]. This method is efficient and does not require interactive communications among multiple evaluators. However, we need to construct a circuit that implements a dynamic programming algorithm, and after the construction we cannot change the algorithm, i.e., the directed graph of the dynamic programming or the number of bidders. In our protocol, by contrast, we can change dynamically these things during the auction.

6 Conclusion

In this paper, we presented a secure dynamic programming method that utilizes secret sharing techniques. By using this method, multiple servers cooperatively solve a combinatorial optimization problem, while the inputs are kept secret even from the servers. Furthermore, we discussed its application to several combinatorial auctions, i.e., multi-unit auctions, linear-goods auctions, and general combinatorial auctions.

One limitation of the proposed method is that, we need a large number of evaluators, since the number of evaluators must be proportional to the length of the longest path. It is an important and challenging problem to develop a method that requires a smaller number of evaluators [37].

Although dynamic programming is a very powerful, widely-used combinatorial optimization method, applying dynamic programming techniques to general combinatorial auctions is not feasible for large-scale problems, since it requires an exponential number of nodes. We are now investigating how to securely compute more average-case efficient algorithms [8, 26, 27].

References

1. Abe, M. and Suzuki, K.: M+1-st Price Auction using Homomorphic Encryption, *Proceedings of Public Key Cryptography 2002* (2002)
2. Baudron, O. and Stern, J.: Non-interactive Private Auctions, *Proceedings of Financial Cryptography 2001* (2001)

3. Bellman, R.: *Dynamic Programming*, Princeton University Press, Princeton, NJ (1957)
4. Ben-Or, M., Goldwasser, S., and Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, *Proceedings of 20th ACM Symposium on the Theory of Computing* (1988) 1–10
5. Cachin, C.: Efficient Private Bidding and Auctions with an Oblivious Third Party, *Proceedings of 6th ACM Conference on Computer and Communications Security* (1999) 120–127
6. Chida, K., Kobayashi, K., and Morita, H.: Efficient Sealed-bid Auctions for Massive Numbers of Bidders with Lump Comparison, *Proceedings of ISC 2001* (2001)
7. Franklin, M. K. and Reiter, M. K.: The Design and Implementation of a Secure Auction Server, *IEEE Transactions on Software Engineering*, Vol. 22, No. 5, (1986) 302–312
8. Fujishima, Y., Leyton-Brown, K., and Shoham, Y.: Taming the Computation Complexity of Combinatorial Auctions: Optimal and Approximate Approaches, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)* (1999) 548–553
9. Goldreich, O., Micli, S., and Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority, *Proceedings of 19th ACM Symposium on the Theory of Computing* (1987) 218–229
10. Harkavy, M., Tygar, J. D., and Kikuchi, H.: Electronic Auctions with Private Bids, *Proceedings of Third USENIX Workshop on Electronic Commerce* (1998) 61–74
11. Kikuchi, H., Harkavy, M., and Tygar, J. D.: Multi-round Anonymous Auction Protocols, *Proceedings of first IEEE Workshop on Dependable and Real-Time E-Commerce Systems* (1998) 62–69
12. Kikuchi, H.: (M+1)st-Price Auction Protocol, *Proceedings of Financial Cryptography 2001* (2001)
13. Klemperer, P.: Auction Theory: A Guide to the Literature, *Journal of Economics Surveys*, Vol. 13, No. 3, (1999) 227–286
14. Lehmann, D., O'Callaghan, L. I., and Shoham, Y.: Truth Revelation in Approximately Efficient Combinatorial Auction, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)* (1999) 96–102
15. Leyton-Brown, K., Pearson, M., and Shoham, Y.: Towards a Universal Test Suite for Combinatorial Auction Algorithms, *Proceedings of the Second ACM Conference on Electronic Commerce (EC-00)* (2000) 66–76
16. Matsui, T. and Watanabe, T.: Sealed bid multi-object auctions with necessary bundles and its application to spectrum auctions, *Proceedings of the 4th Pacific Rim International Workshop on Multi-agents (PRIMA-2001)*, Springer-Verlag (2001) 78–92, Lecture Notes in Artificial Intelligence 2132
17. McMillan, J.: Selling Spectrum Rights, *Journal of Economics Perspectives*, Vol. 8, No. 3, (1994) 145–162
18. Monderer, D. and Tennenholtz, M.: Optimal Auctions Revisited, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)* (1998) 32–37
19. Naor, M., Pinkas, B., and Sumner, R.: Privacy Preserving Auctions and Mechanism Design, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)* (1999)
20. Omote, K. and Myaji, A.: An Anonymous Auction Protocol with a Single Non-trusted Center Using Binary Trees, *Proceedings of ISW2000* (2000) 108–120
21. Rasmusen, E.: *Games and Information*, Blackwell (1994)
22. Rothkopf, M. H., Pekeč, A., and Harstad, R. M.: Computationally Manageable Combinatorial Auctions, *Management Science*, Vol. 44, No. 8, (1998) 1131–1147

23. Sako, K.: Universally verifiable auction protocol which hides losing bids, *Proceedings of Public Key Cryptography 2000* (2000) 35–39
24. Sakurai, K. and Miyazaki, S.: A bulletin-board based digital auction scheme with bidding down strategy, *Proceedings of 1999 International Workshop on Cryptographic Techniques and E-Commerce* (1999) 180–187
25. Sakurai, Y., Yokoo, M., and Kamei, K.: An Efficient Approximate Algorithm for Winner Determination in Combinatorial Auctions, *Proceedings of the Second ACM Conference on Electronic Commerce (EC-00)* (2000) 30–37
26. Sandholm, T.: An Algorithm for Optimal Winner Determination in Combinatorial Auction, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)* (1999) 542–547
27. Sandholm, T., Suri, S., Gilpin, A., and Levine, D.: CABOB: A Fast Combinatorial Algorithm for Optimal Combinatorial Auctions, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)* (2001) 1102–1108
28. Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, (1979) 612–613
29. Stubblebine, S. G. and Syverson, P. F.: Fair On-line Auctions Without Special Trusted Parties, *Proceedings of Financial Cryptography 1999* (1999)
30. Suzuki, K., Kobayashi, K., and Morita, H.: Efficient Sealed-Bid Auction using Hash Chain, *Proceedings of International Conference Information Security and Cryptology 2000 (LNCS 2015)* (2000) 183–191
31. Tennenholtz, M.: Some Tractable Combinatorial Auctions, *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)* (2000) 98–103
32. Hoesel, vanS. and Müller, R.: Optimization in electronic markets: examples in combinatorial auctions, *Netnomics*, Vol. 3, (2001) 23–33
33. Varian, H. R.: Economic Mechanism Design for Computerized Agents, *Proceedings of the First Usenix Workshop on Electronic Commerce* (1995)
34. Wurman, P. R., Wellman, M. P., and Walsh, W. E.: The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents, *Proceedings of the Second International Conference on Autonomous Agents (Agents-98)* (1998) 301–308
35. Yao, A. C.: How to generate and Exchange secrets, *Proceedings of IEEE Symposium on Foundations of Computer Science* (1986) 162–167
36. Yokoo, M., Sakurai, Y., and Matsubara, S.: Robust Combinatorial Auction Protocol against False-name Bids, *Artificial Intelligence*, Vol. 130, No. 2, (2001) 167–181
37. Yokoo, M. and Suzuki, K.: Secure Multi-agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions, *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)* (2002): (to appear)