PP-MCSA: Privacy Preserving Multi-channel Double Spectrum Auction

Zhili Chen[1(✉)], Sheng Chen[1], Hong Zhong[1], Lin Chen[2], and Miaomiao Tian[1]

[1] School of Computer Science and Technology, Anhui University, Hefei 230601, China
{zlchen,mtian}@ahu.edu.cn, shengchen9403@gmail.com

[2] Lab. Recherche Informatique (LRI-CNRS UMR 8623), Univ. Paris-Sud, 91405 Orsay, France
zhongh@mail.ustc.edu.cn, chen@lri.fr

Abstract. Auction is widely regarded as an effective way in dynamic spectrum redistribution. Recently, considerable research efforts have been devoted to designing privacy-preserving spectrum auctions in a variety of auction settings. However, none of existing work has addressed the privacy issue in the most generic scenario, double spectrum auctions where each seller sells multiple channels and each buyer buys multiple channels. To fill this gap, in this paper we propose PP-MCSA, a Privacy Preserving mechanism for Multi-Channel double Spectrum Auctions. Technically, by leveraging garbled circuits, we manage to protect the privacy of both sellers' requests and buyers' bids in multi-channel double spectrum auctions. As far as we know, PP-MCSA is the first privacy-preserving solution for multi-channel double spectrum auctions. We further theoretically demonstrate the privacy guarantee of PP-MCSA, and extensively evaluate its performance via experiments. Experimental results show that PP-MCSA incurs only moderate communication and computation overhead.

1 Introduction

Today, more and more emerging wireless technologies, such as Wifi, 4G, are penetrating into our daily work and life. At the same time, the traditional static and rigid spectrum allocation scheme renders the utilization of radio spectrum severely inefficient and unbalanced. According to the survey [1], many statically allocated spectrum channels are left idle by their current owners, exaggerating the gap between the ever-increasing spectrum demand of wireless services and the spectrum scarcity. Therefore, to improve and balance spectrum utilization, dynamic spectrum redistribution has been advocated to reallocate spectrum among primary and secondary users.

Spectrum auction is widely regarded as an effective way in dynamic spectrum redistribution. A large body of existing studies are focused on designing truthful spectrum auctions, where the auctioneer is assumed to be trusted, and bidders are stimulated to reveal their true valuations of spectrum channels. However, in many practical scenarios, the auctioneer is by nature self-interested and not

© Springer Nature Switzerland AG 2018
D. Naccache et al. (Eds.): ICICS 2018, LNCS 11149, pp. 248–267, 2018.
https://doi.org/10.1007/978-3-030-01950-1_15

trusted. It may disclose the true valuations of bidders, which may cause serious privacy vulnerabilities [2]. For example, a dishonest auctioneer may take advantage of learning the bidders' bids, and then tamper with the auction results so as to increase its own profit. Or the auctioneer may sell bidders' historical bids for profit. Therefore, privacy preservation is critical in spectrum auctions.

There has been significant research attention on privacy preserving auctions, such as [3–5]. These schemes do not consider spectrum reusability, and thus cannot be applied in spectrum auctions. Recently, a handful of propositions addressed privacy issues in spectrum auctions, such as [2,6,7], but most of them focus on protecting privacy for single-sided spectrum auctions. Only a few solutions such as [8] and [9], provide secure designs for double spectrum auctions. However, they assume that in the auction each seller sells only one spectrum channel and each buyer buys only one spectrum channel. Such *one-channel* assumption makes the problem much more tractable, but leaves open the most generic and practical version, the double spectrum auctions, involving multiple spectrum sellers selling multiple channels to multiple buyers [10].

To fill this gap, in this paper, we propose PP-MCSA, a Privacy-Preserving mechanism for Multi-Channel double Spectrum Auctions. Specifically, we manage to protect both sellers' request privacy and buyers' bid privacy for the double spectrum auction mechanism True-MCSA [10] that supports multi-channel auctions. To preserve privacy, we introduce in the auction framework of PP-MCSA a third party, namely an agent, who cooperates with the auctioneer to perform secure auction computations, as shown in Fig. 1. In such a framework, each seller m submits its request value s_m (i.e., the lowest per-channel selling price) and request number c_m (i.e., the number of selling channels) to the auctioneer. Similarly, each buyer n does the same thing with its bid value b_n (i.e., the highest per-channel buying price) and bid number (i.e., the number of buying channels). All submissions are appropriately encrypted such that all sensitive information (i.e. request values, bid values and bid numbers) are protected from either the auctioneer or the agent, but can be securely retrieved and computed with the cooperation between the two parties. Therefore, as long as the auctioneer and the agent do not collude with each other (Note that this assumption is essentially necessary, otherwise the privacy cannot be achieved.), PP-MCSA leaks nothing about the sensitive information to anyone except what can be revealed from the published auction outcome.

We list our main contributions as follows:

- We propose the first privacy-preserving and practical multi-channel double spectrum auction mechanism by combing public-key encryptions and garbled circuits, filling the research gap that there is no privacy consideration in multi-channel double spectrum auctions before.
- We design and optimize data-oblivious algorithms for multi-channel double spectrum auction mechanism True-MCSA, which is rather complex in auction logic, and address both the privacy and efficiency challenges.
- We fully implement PP-MCSA, and conduct extensive experiments to evaluate its computation and communication overhead.

The reminder of this paper is structured as follows. Section 2 briefly reviews related work. In Sect. 3, the underlying mechanism is introduced and the privacy goal is given. We describe the design challenges and rationale in Sect. 4, and present the detailed design of PP-MCSA and prove its privacy in Sect. 5. In Sect. 6, we implement PP-MCSA, and evaluate its performance in terms of computation and communication overheads. Finally, the paper is concluded in Sect. 7.

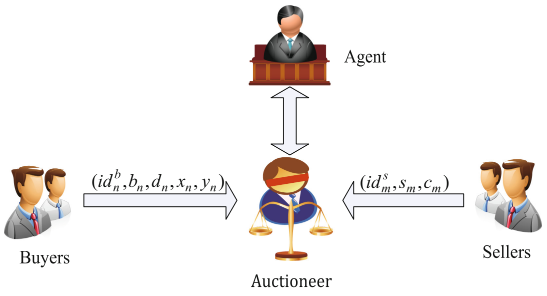


Fig. 1. Privacy-preserving auction framework for PP-MCSA

2 Related Work

In this section, we briefly review the existing works on privacy-preserving auction design, and distinguish our work from the existing ones.

2.1 Spectrum Auction

Spectrum auctions are widely used to redistribute spectrum. In the past few years, many researches have focused on designing truthful spectrum auctions. For example, Zhou et al. put forward TRUST [11], the first truthful double spectrum auction framework exploiting spectrum reusability. Chen et al. proposed the first truthful single-sided auction mechanism TAMES [12] for heterogeneous spectrum auctions, which allows buyers to freely bid their different preferences to heterogenous spectrum channels. Later, Feng et al. presented the first double auction mechanism for heterogeneous spectrum transaction [13]. Chen et al. proposed the first double multi-channel spectrum auction scheme, True-MCSA [10]. However, all the above studies did not address the privacy preservation issues.

2.2 Privacy-Preserving Spectrum Auction

In the past decade, there have been a great number of schemes for privacy-preserving auctions [3–5]. These schemes were originally designed for traditional goods (e.g., painting, stamps), where each commodity can only be allocated to one bidder. Unfortunately, when directly applied to spectrum auctions, they suffer severe under-utilization due to the lack of spectrum reusability consideration.

In recent years, quite a few research efforts have been made for the studies on privacy-preserving spectrum auctions [2,7–9,14,15]. Most, if not all, of them have focused on privacy preservation for single-sided spectrum auctions [2,7,14,15]. Different from these works, our work addresses the generic case of double spectrum auctions. There have been a few schemes for privacy-preserving double spectrum auctions [8,9]. But these schemes only addressed privacy issues for one-channel double spectrum auctions. As far as we know, we are the first to consider privacy preservation for multi-channel double spectrum auctions.

3 Underlying Mechanism and Privacy Goal

In this section, we introduce the underlying mechanism of the double multi-channel spectrum auction, and define the cryptographical protocol privacy.

3.1 TRUE-MCSA Auction Mechanism

Consider a single-round double multi-channel spectrum auction where there is a coordinator as the auctioneer, M primary spectrum users as the sellers, and N secondary spectrum users as the buyers. Consider the general case where each seller sells multiple channels, and each buyer requests multiple channels. The auction is sealed-bid and private, and each bidder (seller or buyer) submits its request or bid to the auctioneer by itself, without knowing any information about other bidders' submissions.

More specifically, in the spectrum auction, a seller m's request is denoted by (s_m, c_m) $(s_m > 0, c_m \geqslant 1)$, meaning that the seller m requires the minimum per-channel payment s_m to sell c_m channels; a buyer n's bid is denoted by (b_n, d_n) $(b_n > 0, d_n \geqslant 1)$, representing that the buyer n is willing to pay the maximum price b_n for each channel, and wants to buy at most d_n channels. We call s_m and c_m the seller m's request value and request number; and call b_n and d_n the buyer n's bid value and bid number.

An existing solution to the above-mentioned double multi-channel spectrum auction problem is True-MCSA auction mechanism [10]. We will use True-MCSA as our underlying double multi-channel spectrum auction mechanism. A brief review of True-MCSA auction can be found in Appendix A.

3.2 Cryptographical Protocol Privacy

Implicitly, True-MCSA assumes that the auctioneer is trusted. However, if this is not the case, True-MCSA simply leaks all requests and bids to the untrusted auctioneer, and thus no privacy is guaranteed.

To protect the privacy of bidders in the case of an untrusted auctioneer, we introduce an agent to cooperatively perform the auction with the auctioneer. Intuitively, our privacy goal is that as long as the auctioneer and the agent do not collude with each other (one of them may be semi-honest), nothing about the sensitive inputs (i.e., bid values, bid numbers, and request values) of bidders is leaked to them through the auction, except what is revealed from the auction outcome. We formally present this privacy definition as follows.

Definition 1 (Privacy against semi-honest adversaries). *Let $f(x, y)$ be a two-party deterministic auction functionality with inputs x and y from the auctioneer and the agent, respectively, and a common auction outcome $f(x, y)$ for both parties. Suppose that protocol Π computes functionality $f(x, y)$ between the auctioneer and the agent. Let $V_A^{\Pi}(x, y)$ (resp. $V_B^{\Pi}(x, y)$) represent the auctioneer's (resp. the agent's) view during an execution of Π on (x, y). In other words, if $(x, \boldsymbol{r}_A^{\Pi})$ (resp. $(y, \boldsymbol{r}_B^{\Pi})$) denotes the auctioneer's (resp. the agent's) input and randomness, then*

$$V_A^{\Pi}(x, y) = (x, \boldsymbol{r}_A^{\Pi}, m_1, m_2, ..., m_t), \text{ and}$$
$$V_B^{\Pi}(x, y) = (y, \boldsymbol{r}_B^{\Pi}, m_1, m_2, ..., m_t)$$

*where $\{m_i\}_{i=1}^{t}$ denote the messages passed between the two parties. Let $O^{\Pi}(x, y)$ denote the auction outcome after an execution of Π on (x, y). Then we have $O^{\Pi}(x, y) = f(x, y)$ for **correctness**, and say that protocol Π **protects privacy** against semi-honest adversaries if there exist probabilistic polynomial time (PPT) simulators S_1 and S_2 such that*

$$S_1(x, f(x, y)) \overset{c}{\equiv} V_A^{\Pi}(x, y) \tag{1}$$

$$S_2(y, f(x, y)) \overset{c}{\equiv} V_B^{\Pi}(x, y) \tag{2}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability.

4 PP-MCSA: Design Challenges and Rationale

In this section, we summarize the main challenges in our design, followed by our design rationale to tackle them.

4.1 Design Challenges

Recently, some secure mechanisms for double spectrum auctions, such as PS-TRUST or SDSA [8,9], have been proposed. However, they all assumed that in the spectrum auction a seller sells one channel, and a buyer buys one channel, and none of them addressed the privacy preservation issue in double multi-channel spectrum auctions. To protect privacy in double multi-channel spectrum auctions like TRUE-MCSA, we face two challenges indicated as follows.

The first one is the *privacy challenge.* As described in Appendix A, TRUE-MCSA involves complex operations in both "VBG splitting and bidding" and "winner determination" steps. How to perform such operations securely by protecting the sensitive inputs is our first challenge.

The second one is the *efficiency challenge.* Straightforwardly securing the auction in our context may result in heavy overhead and thus may degrade the overall performance. Thus, how to achieve practical efficiency in terms of performance with privacy guarantee consists of our second challenge.

4.2 Design Rationale

In order to tackle these two challenges above, we leverage garbled circuits [16,17] to carefully design the boolean circuits corresponding to the auction mechanism. Specifically, to achieve privacy, we designate binary flags to indicate various conditions, and implement the auction functionality based on these flags in a data-oblivious way; to achieve efficiency, we carefully cache some intermediate values, so that unnecessary repeated circuits are avoided.

5 PP-MCSA: Design Details and Proofs

In this section, we elaborate our privacy preserving spectrum auction protocol, namely PP-MCSA, and prove that it is secure against semi-honest adversaries.

5.1 Protocol Framework

In this subsection, we present the protocol framework of PP-MCSA. Generally speaking, PP-MCSA is a secure protocol for double multi-channel spectrum auctions executed between the auctioneer and the agent. We distinguish two types of inputs, *insensitive* and *sensitive* ones, among which the sensitive input needs to be protected in the spectrum auction. We combine public-key encryption with garbled circuits to protect the sensitive input throughout the auction. As shown in Fig. 2, our protocol consists of three phases, namely, *submission*, *group formation*, and *garbled auction computation*, as specified as follows.

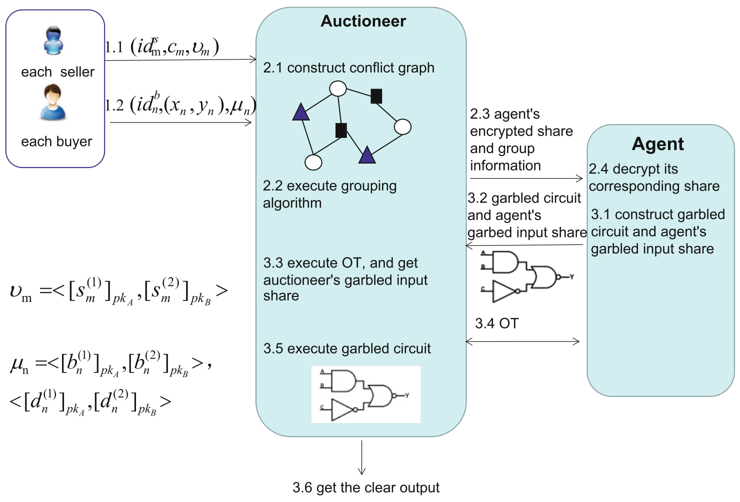
Phase I: Submission
In this phase, sellers and buyers encrypt their respective sensitive inputs, and then send all the necessary inputs to auctioneer. Sensitive inputs include all sellers' request values, all buyers' bid values and bid numbers, while the insensitive inputs include all sellers' IDs and request numbers, and all buyers' IDs and geographic locations. For sensitive inputs, we split all of them into two parts, and then encrypt them respectively with the auctioneer's public key pk_A and the agent's public key pk_B. For insensitive inputs, we directly send them to the auctioneer. The tuples that are submitted by sellers and buyers are presented as follows.

Seller m: $(id^s_m, \langle [s^{(1)}_m]_{pk_A}, [s^{(2)}_m]_{pk_B} \rangle, c_m)$ for $m = 1, 2, ..., M$
Buyer n: $(id^b_n, (x_n, y_n), \langle [b^{(1)}_n]_{pk_A}, [b^{(2)}_n]_{pk_B} \rangle, \langle [d^{(1)}_n]_{pk_A}, [d^{(2)}_n]_{pk_B} \rangle)$ for $n = 1, 2, ..., N$

where $[\cdot]_{pk_A}$ and $[\cdot]_{pk_B}$ denote encryptions with pk_A and pk_B, respectively, and $x^{(1)} + x^{(2)} = x \pmod{2^B}$ for any value x, where B is the bit length used.

Additionally, we assume that all communication channels are authenticated and secure, and no one can eavesdrop the data transmitted on the channels.

Phase II: Group Formation
Upon receiving the inputs from sellers and buyers, the auctioneer firstly constructs a conflict graph using all buyers' geographic locations. Then, according



each seller
each buyer
1.1 $(id_m^s, c_m, \upsilon_m)$
1.2 $(id_n^b, (x_n, y_n), \mu_n)$
Auctioneer
2.1 construct conflict graph
2.2 execute grouping algorithm
3.3 execute OT, and get auctioneer's garbled input share
3.5 execute garbled circuit
2.3 agent's encrypted share and group information
3.2 garbled circuit and agent's garbled input share
3.4 OT
Agent
2.4 decrypt its corresponding share
3.1 construct garbled circuit and agent's garbled input share
3.6 get the clear output
$\upsilon_m = <[s_m^{(1)}]_{pk_A}, [s_m^{(2)}]_{pk_B}>$
$\mu_n = <[b_n^{(1)}]_{pk_A}, [b_n^{(2)}]_{pk_B}>,$
$<[d_n^{(1)}]_{pk_A}, [d_n^{(2)}]_{pk_B}>$

Fig. 2. Protocol framework: First, each buyer or seller submits its input with sensitive parts properly split and encrypted; Next, the auctioneer constructs a conflict graph of buyers, executes buyer grouping algorithm and forwards encrypted input shares to the agent; Then, the agent obtains its corresponding input shares by decrypting the encrypted ones, constructs a garbled circuit based on the auction circuit, garbles its input shares, and sends the garbled circuit and garbled input shares to the auctioneer; Finally, the auctioneer obtains its garbled input shares through running an oblivious transfer with the agent, and executes the garbled circuit and outputs the clear result.

to the conflict graph, the auctioneer executes a bid-independent grouping algorithm to divide buyers into different groups, such that any two members of the same group do not conflict with each other. After group formation, the auctioneer gets group set $G = \{G_1, G_2, \ldots, G_T\}$, where the size of group G_t is denoted by N_t. An example of group formation is illustrated in step 2.1 in Fig. 2, where nodes represent buyers, edges represent conflict relations between buyers, nodes with the same shape represent members in the same group, and thus three groups are formed. At the end of this phase, the auctioneer sends the agent's encrypted shares of sensitive inputs, and the grouping information to the agent. Then, both the auctioneer and the agent can obtain their respective shares of sensitive inputs by decrypting the corresponding encrypted shares with their public keys.

Phase III: Garbled Auction Computation

In this phase, the agent constructs a garbled circuit based on the auction circuit which we will design in the next subsection, garbles its shares of sensitive inputs, and generates the output decoder which can decode the garbled output. The garbled circuit, the agent's garbled input shares, and the output decoder are then sent to the auctioneer. Upon receiving these data, the auctioneer executes oblivious transfers (OTs) with the agent to get its garbled shares of sensitive inputs. Finally, with both garbled shares of sensitive inputs and the insensitive inputs in hand, the auctioneer computes the garbled circuit to get a garbled auction result, and obtains the clear auction result by decoding the garbled one with the output decoder.

The crux of this phase is to design a boolean circuit for our underlying spectrum auction, True-MCSA. A boolean circuit is in essence the binary representation of a data-oblivious algorithm, whose execution path does not depend on its input. In our case, we only need to design auction algorithms which are data-oblivious for sensitive inputs. In the next subsection, we detail our design of such data-oblivious algorithms.

5.2 Data-Oblivious Auction Algorithms

In our context, we only need to protect the sensitive inputs from both sellers and buyers. Thus, we only need to perform sensitive input related operations in the garbled circuits. From here on, we represent a garbled x by $[\![x]\!]$, meaning that x needs to be protected and should remain in the garbled form throughout the computations. Our data-oblivious spectrum auction is further composed of four steps as follows.

(1) ***Initialization.*** In our algorithms, we use arrays of tuples to represent both sellers' and buyers' information. Specifically, we use an array of seller tuples $\mathbb{S}$ to represent all sellers, an array of buyer group tuples $\mathbb{G}$ to represent all buyer groups, an array of buyer tuples $\mathbb{G}_t$ to represent all buyers in the group t ($t = 1, \cdots, T$), and an array of virtual buyer group (VBG) tuples $\mathbb{G}_t^v$ to represent all VBGs derived from the group t. The four types of tuples are designed as follows.

Seller tuple: $(id_j^s, s_j, c_j, w_j^s)$, $j \in [1..M]$
Group tuple: (id_t^g, b_t^g, N_t), $t \in [1..T]$
Buyer tuple: $(id_{t,q}^b, b_{t,q}, d_{t,q}, w_{t,q}^b)$, $q \in [1..N_t]$, $t \in [1..T]$
VBG tuple: $(id_t^g, \pi_{t,k}, n_{t,k}, w_{t,k}^v)$, $k \in [1..D]$, $t \in [1..T]$

In a seller tuple, id_j^s, s_j and c_j are the ID, the request value, and the request number of seller j, respectively, while w_j^s is a binary flag indicating whether the seller is a winner (1) or not (0). In a group tuple, id_t^g, b_t^g and N_t are the ID, the minimum buyer bid, and the size of group t. In a buyer tuple, $id_{t,q}^b$, $b_{t,q}$, $d_{t,q}$ are the ID, the bid value, and the bid number of buyer q in the group t; $w_{t,q}^b$ describes whether a buyer is a winner. In a VBG tuple, $\pi_{t,k}$ and $n_{t,k}$ are the bid, and the size of VBG k derived from group t. $w_{t,k}^v$ is a binary flag indicating whether the VBG k is a winning VBG (1) or not (0). Additionally, D is the maximum bid number of all buyers, which is set as a parameter at the beginning of the auction.

We initialize the arrays $\mathbb{S}$, $\mathbb{G}$, $\mathbb{G}_t$ and $\mathbb{G}_t^v$ as follows, where the "null" symbol $\perp$ is a placeholder.

$$\mathbb{S} = \begin{pmatrix} j: & 1 & \cdots & M \\ id_j^s: & id_1^s & \cdots & id_M^s \\ s_j: & [\![s_1]\!] & \cdots & [\![s_M]\!] \\ c_j: & c_1 & \cdots & c_M \\ w_j^s: & 0 & \cdots & 0 \end{pmatrix}, \quad \mathbb{G} = \begin{pmatrix} t: & 1 & \cdots & T \\ id_t^g: & id_1^g & \cdots & id_T^g \\ b_t^g: & \perp & \cdots & \perp \\ N_t: & N_1 & \cdots & N_T \end{pmatrix}$$

$$\mathbb{G}_t = \begin{pmatrix} q: & 1 & \cdots & N_t \\ id_{t,q}^b: & id_{t,1}^b & \cdots & id_{t,N_t}^b \\ b_{t,q}: & [\![b_{t,1}]\!] & \cdots & [\![b_{t,N_t}]\!] \\ d_{t,q}: & [\![d_{t,1}]\!] & \cdots & [\![d_{t,N_t}]\!] \\ w_{t,q}^b: & 0 & \cdots & 0 \end{pmatrix}, \quad \mathbb{G}_t^v = \begin{pmatrix} k: & 1 & \cdots & D \\ id_t^g: & id_t^g & \cdots & id_t^g \\ \pi_{t,k}: & \perp & \cdots & \perp \\ n_{t,k}: & \perp & \cdots & \perp \\ w_{t,k}^v: & 0 & \cdots & 0 \end{pmatrix}$$

(2) ***VBG splitting and bidding.*** In this step, a data-oblivious algorithm should be designed for VBG splitting and bidding. The challenge is that this process depends on both buyers' bid values and their bid numbers, which are sensitive inputs and should be protected.

To design the data-oblivious algorithm, one difficulty is that we do not know the buyers' bid numbers since they are protected in garbled form, and thus we do not know how many VBGs should be derived from each buyer group. To overcome this difficulty, we assume that the maximum bid number $D = \max_t D_t$ is known, and hence derive exactly D VBGs from each buyer group. To protect both bid values and bid numbers, we keep them and their related computation results in garbled form, while use appropriate logic circuit to implement all required operations. The resulted algorithm is shown in Algorithm 1. Note that we only implement MMIN as the VBG bidding method, while GMAX can be similarly implemented.

Some explanations about Algorithm 1 are as follows.

First, for each group t, the algorithm compares every pair of neighboring buyer tuples (i.e., tuples j and $j+1$ in $\mathbb{G}_t$ for $j = 1$ to $N_t - 1$) in terms of their bid values, and swaps the two tuples if the former is smaller than the later, such that finally the tuple with the minimum bid value is placed at the last position of $\mathbb{G}_t$ (Line 2 to 5). Note that in Line 4, function $swap(\mathbb{G}_t, [\![\lambda]\!], j, j+1)$ swaps the two tuples j and $j+1$ of $\mathbb{G}_t$ if $\lambda = 1$. For each field x of the tuples, the swapping function can be implemented using the following circuit [18]:

$$x'_j \leftarrow ((x_j \oplus x_{j+1}) \cdot \lambda) \oplus x_j$$
$$x'_{j+1} \leftarrow x'_j \oplus (x_j \oplus x_{j+1})$$

where x'_j and x'_{j+1} represent the resulted field values. This circuit is very efficient for garbled circuits, since it needs only one non-XOR gate for swapping each pair of bits. Using the free XOR technique, garbled circuits can execute all XOR gates nearly for free, and thus their performances are determined by the number of non-XOR gates executed.

Algorithm 1. Data-oblivious VBG spplitting and bidding

```
Require: Tuple arrays $\mathbb{G}$ and $\{\mathbb{G}_t\}_{t=1}^{T}$
Ensure: The tuple array $\mathbb{G}_t^v$
 1: for $t = 1 \rightarrow T$ do
 2:     for $j = 1 \rightarrow N_t - 1$ do
 3:         $[\![\lambda]\!] \leftarrow ([\![b_{t,j}]\!] < [\![b_{t,j+1}]\!])$;
 4:         $swap(\mathbb{G}_t, [\![\lambda]\!], j, j+1)$;
 5:     end for
 6:     for $k = 1 \rightarrow D$ do
 7:         $[\![n_{t,k}]\!] \leftarrow 0$;
 8:         for $j = 1 \rightarrow N_t - 1$ do
 9:             $[\![\gamma]\!] \leftarrow ([\![d_{t,j}]\!] \geq k)$;
10:             $[\![n_{t,k}]\!] \leftarrow [\![n_{t,k}]\!] + [\![\gamma]\!]$;
11:         end for
12:         $[\![\pi_{t,k}]\!] \leftarrow [\![b_{t,N_t}]\!] \cdot [\![n_{t,k}]\!]$;
13:     end for
14: end for
        return $\mathbb{G}_t^v$
```

Second, Lines 6 to 13 compute the D VBGs for each group t. To compute the kth VBG, the bid number of each group member except the last one (who has the minimum bid value) is compared with k (Line 9), and if it is not smaller than k, the group member is added to the VBG (Line 10). Finally, the bid value of the kth VBG is computed (Line 12).

Note that in the computations, the sensitive inputs, i.e. $b_{t,j}$'s and $d_{t,j}$'s, and their related computation results, i.e., λ's, γ's, $n_{t,k}$'s and $\pi_{t,k}$'s, are all kept in garbled form, such that the sensitive inputs can be well protected.

(3) ***Winner determination.*** This step applies a variant of McAfee framework to determine winners as shown in Sect. 3. Since this process contains numerous operations, such as comparisons and selections, depending on requests or bids, designing its data-oblivious version is challenging. In order to address this challenge, our main idea is to introduce some appropriate binary flags to indicate different conditions, and construct suitable circuits based on them to data-obliviously achieve the required functions. We describe the data-oblivious winner determination in Algorithm 2.

The details of Algorithm 2 are described as follows.

First, both seller tuples and VBG tuples are appropriately sorted as required in McAfee framework (Lines 1 to 4). In Line 1, the total number of selling channels L is computed in the clear, since initially all request numbers c_j's are not protected. In Line 3, all VBG tuples from different groups are merged into a uniform VBG tuple array $\mathbb{G}^v = \{id_k^v, \pi_k, n_k, w_k^v\}_{k=1}^{T}$, where $id_k^v \in \{id_t^g\}_{t=1}^{T}$, and then in Line 4 $\mathbb{G}^v$ is sorted in term of π_k's. Note that once sorted (Lines 2 & 4), all fields of $\mathbb{S}$ and $\mathbb{G}^v$ become garbled, otherwise the ranking information of s_i's and π_k's would be leaked.

Algorithm 2. Data-oblivious winner determination

```
Require: Tuple arrays $\mathbb{S}$ and $\{\mathbb{G}_t^v\}_{t=1}^T$
Ensure: The winning seller tuple array $\mathbb{W}^s$, the winning VBG tuple array $\mathbb{W}^v$, and
    the critical request value $\varphi$
 1: Compute $L \leftarrow \sum_{i=1}^M c_i$;
 2: Sort $\mathbb{S}$ in no-descending order of $s_i$'s, s.t.
        $[\![s_1]\!] \leq [\![s_2]\!] \leq ... \leq [\![s_M]\!]$
 3: Merge $\mathbb{G}^v \leftarrow \bigcup_{t=1}^T \mathbb{G}_t^v$;
 4: Sort $\mathbb{G}^v$ in no-increasing order of $\pi_k$'s, s.t.
        $[\![\pi_1]\!] \geq [\![\pi_2]\!] \geq ... \geq [\![\pi_K]\!]$
 5: $Q \leftarrow \min\{L, K\}$; $[\![\varphi]\!] \leftarrow 0$; $[\![W]\!] \leftarrow 0$;
 6: for $i = 1 \rightarrow Q$ do
 7:     $[\![\lambda_M]\!] \leftarrow 0$; $[\![\delta_M]\!] \leftarrow 0$;
 8:     $[\![j_i]\!] \leftarrow 0$; $[\![\varphi_i]\!] \leftarrow 0$;
 9:     $[\![W_i]\!] \leftarrow 0$;
10:     for $j = M - 1 \rightarrow 1$ do
11:         $[\![\lambda_j]\!] \leftarrow [\sum_{l=1}^j [\![c_l]\!] < i]$;
12:         $[\![\delta_j]\!] \leftarrow [\![\lambda_j]\!] \oplus [\![\lambda_{j+1}]\!]$;
13:         $[\![j_i]\!] \leftarrow [\![j_i]\!] + [\![\delta_j]\!] \cdot j$;
14:         $[\![\varphi_i]\!] \leftarrow [\![\varphi_i]\!] + [\![\delta_j]\!] \cdot [\![s_{j+1}]\!]$;
15:         $[\![W_i]\!] \leftarrow [\![W_i]\!] + [\![\delta_j]\!] \cdot \sum_{l=1}^j [\![c_l]\!]$;
16:     end for
17:     $[\![\omega_i]\!] \leftarrow [(\sum_{l=1}^i [\![\pi_l]\!]) \geq i \cdot [\![\varphi_i]\!]]$;
18:     if $i > 1$ then
19:         $[\![\varphi]\!] \leftarrow [\![\varphi]\!] + [\![\varphi_{i-1}]\!] \cdot ([\![\omega_{i-1}]\!] \oplus [\![\omega_i]\!])$;
20:         $[\![W]\!] \leftarrow [\![W]\!] + [\![W_{i-1}]\!] \cdot ([\![\omega_{i-1}]\!] \oplus [\![\omega_i]\!])$;
21:     end if
22: end for
23: $[\![\varphi]\!] \leftarrow [\![\varphi]\!] + [\![\varphi_Q]\!] \cdot [\![\omega_Q]\!]$;$[\![W]\!] \leftarrow [\![W]\!] + [\![W_Q]\!] \cdot [\![\omega_Q]\!]$;
24: Reveal $[\![W]\!]$, and $\mathbb{W}^v \leftarrow$ the first $W$ tuples of $\mathbb{G}^v$;
25: Reveal $[\![j_{W+1}]\!]$, and $\mathbb{W}^s \leftarrow$ the first $j_{W+1}$ tuples of $\mathbb{S}$;
26: Reveal $[\![\varphi]\!]$ as the critical request value;
27: Resort $\mathbb{W}^v$ in increasing order of $id_t^g$'s, and then in no-increasing order of $\pi_k$'s;
28: Resort $\mathbb{W}^s$ increasing order of $id_j^s$'s;
        return $\mathbb{W}^s, \mathbb{W}^v, \varphi$;
```

Second, winners are determined with two nested for loops (Lines 5 to 22). Specifically, the outer loop iterates over each possible trade i, and computes $[\![\omega_i]\!]$ indicating whether trade i is profitable (Line 17), the critical request value $[\![\varphi]\!]$ (Line 19), and the number of winning VBGs $[\![W]\!]$ (Line 20). While the inner loop computes the index j_i of the last winning seller (Lines 8 & 13), the critical request value φ_i (Lines 8 & 14), the number of winning VBGs W_i (Lines 9 & 15), given trade i is the last profitable trade. Note all these computations are performed in the garbled form.

More specifically, to find the index j_i of the last profitable seller provided the last profitable trade i, we introduce two vectors of flags, i.e., λ_j's and δ_j's, where

λ_j: indicates whether $j \leq j_i$, i.e., $\sum_{l=1}^{j} c_l < i$ ($\lambda_j = 1$) or not ($\lambda_j = 0$) (Lines 7 & 11).

δ_j: indicates whether $j = j_i$ ($\delta_j = 1$) or not ($\delta_j = 0$) (Lines 7 & 12).

According to the auction logic, the two flag vectors should take values as the following pattern:

$$\begin{pmatrix} j : 1 \cdots j_i - 1 \; j_i \; j_i + 1 \cdots M \\ \lambda_j : 1 \cdots \quad 1 \quad 1 \quad 0 \quad \cdots 0 \\ \delta_j : 0 \cdots \quad 0 \quad 1 \quad 0 \quad \cdots 0 \end{pmatrix}$$

Thus, $\delta_j = \lambda_j \oplus \lambda_{j+1}$ holds (Line 12).

With similar idea, we compute the profitable flags $[\![\omega_i]\!]$'s, and the critical request value $[\![\varphi]\!]$ (which is the request value of the critical seller) and the number of winning VBGs $[\![W]\!]$ (Lines 17 to 20, and Line 23).

It is worth noting that in Line 14, we use s_{j+1} instead of s_j, since the critical seller is next to the last winning seller. Additionally, in Lines 11, 15 & 17, for simplicity, we repeatedly use the sum equations of c_l's or π_l's. However, in real implementation, it is not necessary to repeatedly compute the sums. Optimally, we can compute each sum just once, and cache them for later use.

Finally, some garbled results are appropriately revealed. Specifically, the number of winning VBGs $[\![W]\!]$ is revealed, and the first W tuples of $\mathbb{G}^v$ form the winning VBG tuple array $\mathbb{W}^v$ (Line 24). Then, $[\![j_{W+1}]\!]$ is revealed as the number of winning sellers, and the first j_{W+1} seller tuples of $\mathbb{S}$ form the winning seller tuple array $\mathbb{W}^s$ (Line 25). Next, $[\![\varphi]\!]$ is revealed as the critical request value(Line 26). At the same time, $\mathbb{W}^v$ and $\mathbb{W}^s$ are appropriately resorted, such that the bid order of winning VBGs from different groups and the request order of winning sellers will not be revealed when decoded in the later (Lines 27 & 28). At last, $\mathbb{W}^v$, $\mathbb{W}^s$, and φ are returned.

(4) ***Pricing.*** In this step, we compute the selling prices for winning sellers and the buying prices for winning buyers, as described in Algorithm 3. Specifically, each winning seller m sells all its c_m channels, and is paid by its selling price $p_m^s = c_m \cdot \varphi$ (Lines 1 to 14). Each winning VBG k is charged by its bid π_k, which is evenly shared by the winning buyers in the VBG. Thus, each winning buyer $n \in G_t$ is charged by its buying price $p_n^b = \min(d_n, D_t) \cdot b_t^g$, where $D_t = \sum_{V \in \mathbb{W}^v} [V.id_k^v = id_t^g]$ is the total number of winning channels for group G_t, and $b_t^g = \pi_{t,k}/(n_{t,k} - 1)$ is the minimum bid value of group G_t. Note that Lines 5 to 9 compute the set of winning buyer groups G^w, and Lines 10 to 18 compute the winning buyers in all winning groups and their prices.

Algorithm 3. Pricing

```
Require: The winning seller tuple array $\mathbb{W}^s$, the winning VBG tuple array $\mathbb{W}^v$, and
    the critical request value $\varphi$;
Ensure: Winners and their clearing prices;
 1: for $E \in \mathbb{W}^s$ do
 2:     Reveal $E.id_m^s$;
 3:     Seller $m$ sells $c_m$ channels, and is paid with $p_m^s \leftarrow c_m \cdot \varphi$;
 4: end for
 5: $G^w \leftarrow \phi$;
 6: for $V \in \mathbb{W}^v$ do
 7:     Reveal $V.id_k^v$ as $id_t^g$;
 8:     $G^w \leftarrow G^w \cup \{t\}$;
 9: end for
10: for $t \in G^w$ do
11:     $D_t = \sum_{V \in \mathbb{W}^v} [V.id_k^v = id_t^g]$;
12:     for $q = 1 \rightarrow N_t - 1$ do
13:         Reveal $id_{t,q}^b$ as $id_n^b$;
14:         $[\![h_t]\!] \leftarrow \min([\![d_n]\!], D_t)$;
15:         Reveal $[\![h_t]\!]$ and $[\![b_t^g]\!]$;
16:         Buyer $n$ buys $h_t$ channels, and pays $p_n^b \leftarrow h_t \cdot b_t^g$;
17:     end for
18: end for
        return All winners and their prices;
```

5.3 Security Analysis

In this section, we prove that our protocol preserves privacy in the sense of cryptography.

Theorem 1. *As long as the auctioneer and the agent do not collude with each other, PP-MCSA preserves privacy against semi-honest adversaries.*

Proof: The proof of privacy for both Phases I and II is trivial. The reasons are as follows. In Phase I no secure computations are involved, sensitive inputs are secretly shared between the auctioneer and the agent, and hence the view of adversary can be easily simulated. While in Phase II, group formation is completely dependent on sensitive inputs, and no privacy issues need to be considered. Therefore, we mainly prove the privacy of Phase III.

To prove the privacy of garbled auction computation phase, we actually need to prove the privacy of Algorithms 1, 2 and 3 separately, and then by applying the sequential composition theory [19] we can conclude that the phase III preserve privacy, and thus the whole protocol also preserve privacy.

We now examine the design of Algorithms 1, 2 and 3. We can see that for every sensitive input related operation, the algorithms compute it in a garbled circuit, and they also store every sensitive input related value by garbled values. At the same time, all garbled values that are revealed in the algorithms carry no more information than what can be inferred from the auction outcome. That is, these algorithms do not revealed any information about the sensitive information except what can be revealed from the auction outcome. By the privacy definition in Sect. 3.2, when one party (the auctioneer or the agent) is corrupted, the view of the adversary can be easily simulated (e.g., an encrypted or garbled value can be simulated by a random number of the same bit length). As a result, we can conclude that our algorithms achieve the privacy of garbled circuits, whose privacy is formally proved in [16].

Therefore, as long as the auctioneer and the agent do not collude with each other, PP-MCSA preserves privacy. □

6 Performance Evaluation

6.1 Experimental Setting

We implement our protocol in two cases: *original* implementation and *improved* implementation. In the original implementation, we implement our algorithms literally, while in the improved implementation, we implement them with cache optimization, where we compute the repeatedly used sums (i.e., Lines 11, 15 & 17 in Algorithm 2) just once and cache them for later use. Our experiments are carried out on top of FastGC [20], a java-based framework for the garbled circuit computations. We simulate the auctioneer and the agent with two processes on the same computer. Experimental settings are as follows: buyers are randomly distributed in a 2000m $\times$ 2000m area, and the interference radius is 400m. The request values of sellers and the bid values of buyers are generated randomly in the intervals [1..150] and [1..50], respectively. The both request numbers and bid numbers are generated randomly in the interval [1..10], and thus D is set to 10 which is the maximum bid number. Throughout our experiments, we use bit length 16, unless otherwise stated, and each point represents the average of 10 times simulation runs.

In the simulation, we run our protocol on a 64-bit Windows 7 Desktop with Intel(R) Core(TM) i5 CPU @3.3 GHz and 8 GB of memory. We focus on the following two performance metrics:

- *Computation overhead:* total running time for executing our protocol by the auctioneer and the agent.
- *Communication overhead:* total communication cost (data size of all messages that are sent between the auctioneer and the agent).

6.2 Result Analysis

We conduct experiments to compare the performance of the original and improved implementations in two cases: (1) when the number of sellers varies; (2) when the number of buyers varies. We further trace the performance of the improved implementation (3) when the bit length of request values and bid values varies; and (4) when bigger numbers of sellers and buyers vary.

(1) Number of sellers varies. Figure. 3 illustrates the comparisons of both computation and communication overheads between the original and improved implementations, when the number of sellers M increases from 50 to 100, and the number of buyers is fixed at $N = 500$, and $N = 600$. We can see that both running time and communication cost of the original implementation increase much faster than those of the improved implementation. The reason is that the cache optimization in the improved implementation (Lines 11, 15 and 17 in Algorithm 2) avoids repeating the addition computations in the nested loops, and thus greatly reduces the computation and communication overheads.

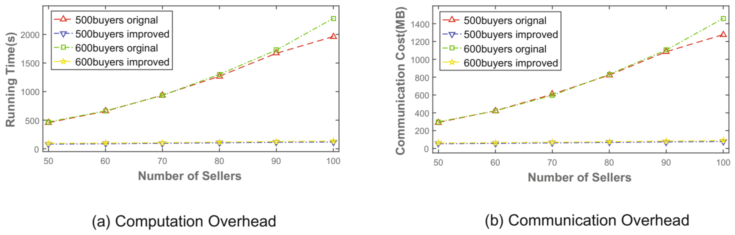


Fig. 3. Comparisons of computation and communication overheads between the original and improved implementations as the number of sellers M varies.

(2) Number of buyers varies. Figure 4 shows the performance comparisons between the original and improved implementations, when the number of sellers is fixed to $M = 100$ and $M = 110$, and the number of buyers N increases from 200 to 600. Similar to Figs. 3 and 4 demonstrates that the improved implementation is much more efficient than the original one in term of computation and communication overheads. In the same way, the cache optimization is the source of this performance improvement.

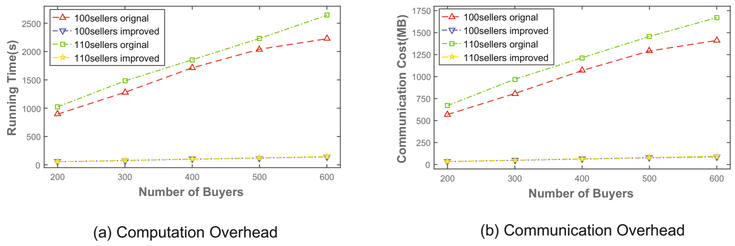
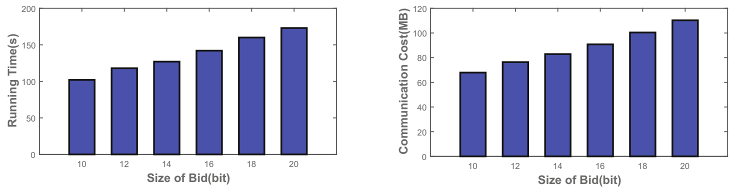


Fig. 4. Comparisons of computation and communication overheads between the original and improved implementations as the number of buyers N varies.

(3) Bit length varies. Figure 5 traces the impact on performance when changing the bit length of bid values and request values in the improved implementation. We vary the bit length from 10 to 20, while fix the number of sellers at

$M = 80$, and the number of buyers at $N = 500$. We can observe that both computation and communication overheads grow linearly as the bit length increases. This is natural, since most of the elemental boolean circuits (e.g., addition, comparison) grow linearly in size when the bit length of its input values increases.

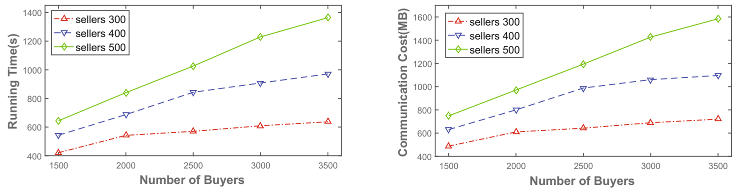


(a) Computation Overhead

(b) Communication Overhead

Fig. 5. Comparisons of computation and communication overheads between the original and improved implementations as the bit length varies.

(4) Then bigger numbers of sellers and buyers vary. Figure 6 traces the performance of the improved implementation when the number of buyers varies from 1500 to 3500, for the number of sellers $M = 300$, 400 and 500, respectively. This figure shows that our improved implementation is rather efficient in both computation and communication performance for bigger numbers. For example, all running times are within 23 min, and all communication costs are within 1600 MB. Meanwhile, both computation and communication overheads scale gracefully as the numbers of sellers and buyers increase.



(a) Computation Overhead

(b) Communication Overhead

Fig. 6. Computation and communication overheads of the improved implementation as the big numbers of sellers and buyers vary.

7 Conclusion

In this paper, we have proposed PP-MCSA, the first privacy-preserving mechanism for multi-channel double spectrum auctions to our knowledge. To address the challenges imposed by the multi-channel double spectrum auction scenario, we have leveraged garbled circuits in our protocol design. Specifically, we design data-oblivious algorithms whose execution path does not depend on their sensitive inputs and then turn these algorithms into garbled circuits to address the privacy challenge. Then, we use cache optimization, which caches some intermediate values to avoid repeated circuits, to improve the garbled circuits and hence address the efficiency challenge. Finally, we have theoretically proved the privacy of PP-MCSA, and experimentally shown that it incurs limited computation and communication overheads.

Acknowledgment. The work is supported by the Natural Science Foundation of China under Grant No. 61572031 & 61502443. We thank the anonymous reviewers for their valuable comments that helped improve the final version of this paper.

A True-MCSA Auction

True-MCSA is a truthful double spectrum auction mechanism that allows multi-channel requests from both buyers and sellers, while ensures spectrum reusability. The symbols of the auction can be described in Table 1. Specifically, TRUE-MCSA is composed of the following four steps:

(1) **Bid-independent Buyer Group Formation:** In this step, the conflict graph of buyers is constructed in term of their geographic locations, and buyers that do not interfere with each other are grouped into the same group. In this way, buyers in the same group can use the same channels without interference. Note that the group formation algorithm should be bid-independent, otherwise bid manipulation is allowed, and hence the auction becomes untruthful.

Table 1. Key symbols for TRUE-MCSA

M, N	Numbers of sellers and buyers
T	Numbers of buyer groups
s_m, c_m	Seller m's request value and request number
b_n,d_n	Buyer n's bid value and bid number
(x_n, y_n)	Location of buyer n
D_t	Maximal number of channel of group t
π	Bid vector of virtual buyer group
S	Request vector of sellers
$j(i)$	The seller in the i^{th} trade
k_l	The last profitable trade
L	Sum of sellers channel number
G	$G = \{G_t\}_{t=1}^{T}$, Global bid set of groups
G_t	The tuple of group t
G_t^v	The tuple of virtual buyer group t

(2) **Virtual Buyer Group (VBG) Splitting and Bidding:** To address the multi-channel requests from buyers, this step splits a buyer group G_t into $D_t = \max_{i \in G_t} d_i$ virtual buyer groups (VBGs), where each VBG only requests for one channel.
After splitting a buyer group into VBGs, we come up with the VBG bidding. Paper [10] proposed two VBG bidding algorithms, member-minimized (MMIN) and group-maximized (GMAX). We only review MMIN as follows. To bid for each VBG derived from a buyer group, the group member with the minimum bid is chosen as the critical buyer, which is removed from all the derived VBGs. Then, each VBG bids with the minimum bid (i.e., the critical buyer's bid) multiplying its size after removing.

(3) **Winner Determination:** At this point, suppose after VBG splitting and bidding we get totally K VBGs with bid values $\{\pi_k\}_{k=1}^{K}$. Recall that we have M sellers with request values $\{s_m\}_{m=1}^{M}$. Then, this step applies McAfee's framework to winner determination as follows.

First of all, the sellers' request values s_m's are sorted in non-decreasing order, and the VBGs' bid values π_k's are sorted in non-increasing order:

$$O' : s_1 \leq s_2 \leq \ldots \leq s_M$$

$$O'' : \pi_1 \geq \pi_2 \geq \ldots \geq \pi_K$$

Then, each seller's request value s_m is rewritten as many times as the number c_m of channels he bid, resulting in the bid mapping between sellers and VBGs as follows:

$$\begin{aligned} O' &: \overbrace{s_1 \leq \ldots \leq s_1}^{c_1} \leq \overbrace{s_2 \leq \quad \ldots \quad \leq s_2}^{c_2} \leq \ldots \leq \overbrace{s_M \leq \quad \ldots \quad \leq s_M}^{c_M} \\ O'' &: \pi_1 \geq \ldots \geq \pi_{c_1} \geq \pi_{c_1+1} \geq \ldots \geq \pi_{c_1+c_2} \geq \ldots \geq \pi_{1+\sum_{t=1}^{M-1} c_t} \geq \ldots \geq \pi_K \end{aligned}$$

Finally, find the last profitable trade k_l as:
$k_l = \arg\max_{i \leq \min\{L,K\}} \{\sum_{t=1}^{i} \pi_t \geq i \cdot s_{j(i)}\}$
Here, L represents the total number of channels provided by sellers, namely, $L = \sum_{j=1}^{M} c_j$, and $j(i)$ computes the seller index j when the trade index is i, namely,

$$j(i) = 1 + \arg\max_{0 \leq h \leq M-1} \{\sum_{t=1}^{h} c_t < i\}.$$

As a result, the last profitable seller is $j(k_l)$. In order to achieve truthfulness, the last profitable seller, as well as all trades involving the seller, should be sacrificed to price the winners. Then the auction winners are the first $j(k_l) - 1$ sellers in O' and the first $k = \sum_{t=1}^{j(k_l)-1} c_t \leq k_l - 1$ VBGs in O''.

(4) **Pricing**: Each buyer in the same winning VBG pays an even share of the VBG bid, and each winning channel is paid by the price $s_{j(k_l)}$. As a result, each winning buyer pays the sum of what it pays in all the winning VBGs it belongs to, and each winning seller is paid with the product of multiplying its request number and the price $s_{j(k_l)}$.

References

1. Valenta, V., Maršálek, R., Baudoin, G., Villegas, M., Suarez, M., Robert, F.: Survey on spectrum utilization in Europe: measurements, analyses and observations. In: Proceedings of CROWNCOM (2010)
2. Pan, M., Sun, J., Fang, Y.: Purging the back-room dealing: secure spectrum auction leveraging paillier cryptosystem. IEEE JSAC **29**(4), 866–876 (2011)
3. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of EC (1999)
4. Yokoo, M., Suzuki, K.: Secure generalized Vickrey auction without third-party servers. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 132–146. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27809-2_17
5. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: Robust, privacy protecting and publicly verifiable sealed-bid auction. In: Proceedings of ICICS (2002)
6. Huang, H., Li, X., Sun, Y., Xu, H.: PPS: privacy-preserving strategyproof social-efficient spectrum auction mechanisms. IEEE Trans. Parallel Distrib. Syst. **26**(5), 1393–1404 (2014)
7. Huang, Q., Tao, Y., Wu, F.: SPRING: a strategy-proof and privacy preserving spectrum auction mechanism. In: Proceedings of INFOCOM (2013)
8. Chen, Z., et al.: PS-TRUST: provably secure solution for truthful double spectrum auctions. In: Proceedings of INFOCOM (2014)
9. Chen, Z., Wei, X., Zhong, H., Cui, J., Xu, Y., Zhang, S.: Secure, efficient and practical double spectrum auction. In: Proceedings of IEEE/ACM IWQoS, pp. 1–6 (2017)
10. Chen, Z., Huang, H., Sun, Y., Huang, L.: True-MCSA: a framework for truthful double multi-channel spectrum auctions. IEEE Trans. Wirel. Commun. **12**(8), 3838–3850 (2013)
11. Zhou, X., Zheng, H.: TRUST: a general framework for truthful double spectrum auctions. In: Proceedings of INFOCOM, pp. 999–1007 (2009)
12. Chen, Y., Zhang, J., Wu, K., Zhang, Q.: TAMES: a truthful double auction for multi-demand heterogeneous spectrums. IEEE Trans. Parallel Distrib. Syst. **25**(11), 3012–3024 (2014)
13. Feng, X., Chen, Y., Zhang, J., Zhang, Q., Li, B.: TAHES: a truthful double auction mechanism for heterogeneous spectrums. IEEE Trans. Wirel. Commun. **11**(11), 4038–4047 (2012)
14. Huang, Q., Gui, Y., Wu, F., Chen, G.: A general privacy-preserving auction mechanism for secondary spectrum markets. IEEE/ACM Trans. Netw. **24**(3), 1881–1893 (2016)
15. Sun, Y., et al.: Privacy-preserving strategyproof auction mechanisms for resource allocation in wireless communications. In: Procedings of BigCom (2016)
16. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: FOCS (1982)

17. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. J. Cryptol. **22**, 161–188 (2009)
18. Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., Boneh, D.: Privacy-preserving matrix factorization. In: Proceedings of ACM CCS (2013)
19. Goldreich, O.: Foundations of Cryptography: Volume 2-Basic Applications. Cambridge University Press, Cambridge (2004)
20. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium, vol. 201, no. 1 (2011)