Multi-Party Computation Auction Mechanisms for a P2P Electricity Market with Geographical Prioritization

Mariana Gama[1], Fairouz Zobiri[2], and Svetla Nikova[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium.
[2] ESAT-ELECTA & EnergyVille, KU Leuven, Leuven, Belgium.
mariana.botelhodagama@kuleuven.be
fairouz.zobiri@kuleuven.be,
svetla.nikova@kuleuven.be

Abstract. A peer-to-peer (P2P) electricity market allows prosumers, who have residential renewable generation units, to trade their excess generation with other consumers or prosumers within their community. This increases the financial well-being of both buyers and sellers while also incentivizing more users to become prosumers themselves. Unfortunately, if the data associated with the users' consumption pattern is public, it may facilitate consumer profiling and ultimately reveal further details about their private lives. The privacy concerns associated with P2P electricity trading deter users from entering the market. It is, therefore, necessary to resort to privacy-enhancing technologies to build a privacy-preserving P2P market that can be safely adopted. Additionally, we would like to pave the way for more energetically independent local energy communities. Thus, we propose to use auction mechanisms where priority is given to trades within the same geographical area, thus minimizing transmission losses.

Multi-party computation (MPC) is a cryptographic technique that allows several parties to perform computations over their inputs while keeping them private. Previous work has already demonstrated that MPC can be used for computing secure auctions. In particular, different auction mechanisms were implemented with MPC, with the goal of securing markets in the financial services sector.

In this work, we consider two of the auction algorithms that were previously considered for the dark pool use case and apply them in the context of an intraday electricity market. We extend these algorithms in order to prioritise users in the same geographical areas (i.e., closest neighbours). The algorithms will be implemented and tested using the SCALE-MAMBA MPC software system to ascertain their suitability for P2P electricity markets. We test them for different parameter sets such as the number of submitted orders, the total number of neighbourhoods and the number of priority groups. For the Volume Matching algorithm, we additionally explore the possibility of prioritizing users who submit small volumes. The results show that the Volume Matching algorithm is more efficient than the Continuous Double Auction, achieving a high order throughput even with the additional functionality of prioritizing small volume orders. The Continuous Double Auction has the advantage that prosumers can determine the price at which they wish to trade electricity, but due to its slower runtimes, it should only be used when we expect a low order submission frequency.

1 Introduction

The emergence of the Smart Grid, a next-generation electricity grid supporting the bidirectional flow of both data and electricity, facilitates the emergence of a peer-to-peer (P2P) electricity market, where citizens can trade the electricity generated by their own Renewable Energy Sources (RESs) between themselves. The P2P electricity market allows the owners of RESs to get appropriate compensation for any excess electricity generated by selling it directly to other users, instead of injecting it back into the grid for little to no remuneration [tar, ver]. This new paradigm will therefore incentivize more users to acquire their own RESs, while also benefiting the grid by promoting exchanges between users nearby and hence diminishing transmission losses [SHPB12]. However, this new market is not without risk, and there are several privacy concerns associated with sensitive personal data that could be revealed in the process. Knowing someone's energy consumption patterns allows consumer profiling and might reveal other personal information related to lifestyle and habits [MCA16].

To address these privacy issues, cryptographic techniques such as multi-party computation (MPC) and homomorphic encryption (HE) have been used to design privacy-preserving auction mechanisms. MPC and HE possibilitate computing over encrypted data, meaning that we can process orders and find matches without accessing the information they contain. In [SSAA21], the authors presented a privacy-preserving discrete-time double auction algorithm using HE. Even though the orders are anonymized, the information they contain is published on a public bulletin board. In [ZGND22], MPC is used to develop a privacy-preserving algorithm for the day-ahead flexibility market. However, the algorithm is complex and requires many expensive MPC operations, rendering it too slow for the intraday market. An MPC-based algorithm for an electricity auction is proposed in [AACM16], and the simulation results confirm its suitability for the electricity market. Nonetheless, geographical prioritization was not considered.

There are also several results concerning privacy-preserving auctions for securing dark pools [CSTA19, dGCP+21, CSTA22]. In these works, it was shown that it is indeed possible to use MPC for securing high-frequency trading platforms, with some of the algorithms having an appropriate performance for real-world use. However, these algorithms do not consider geographical distance, and all the information is publicly revealed as soon as the orders are matched. For the P2P electricity market, we would like to give priority to orders coming from close neighbours, and the amount of electricity to be traded by any given user should only be revealed to them.

In this work, we use MPC and adapt two of the algorithms previously proposed for the financial sector to the P2P intraday electricity market. We present a continuous double auction (CDA) algorithm, modified from [CSTA19] in order to prioritize close neighbours and reveal the final output to the corresponding users only. We also present a volume matching algorithm, modified from the one presented in [dGCP+21], where we additionally allow prioritizing orders of smaller volume. Even though the increased complexity makes these algorithms slower than their original versions, we obtain the functionality we desire while maintaining good enough performance for our use case.

The rest of the paper is organised as follows: Section 2 gives a short overview of the used cryptographic techniques, along with the requirements our algorithms should fulfil. Section 3 presents the algorithms for executing the auction mechanisms in a privacy-preserving manner. Section 4 gives and analyses the performance results. Finally, Section 5 summarises our work and discusses possible directions for future research.

2 Preliminaries

Multi-party computation: Using MPC, a set of distrustful parties can compute over their inputs without revealing anything but the final output or any other information that can be deduced from it. In this work, we use SCALE-MAMBA [ACC+21], a software that implements various MPC protocols based on secret sharing. In these protocols, secret values are divided into multiple shares, one for each computational party. A share by itself does not leak any information about the secret value, which we can only recover by putting together a certain number of shares. When a value x is secret shared we write it as $\langle x \rangle$. We will be using Shamir Secret Sharing based MPC, which is guaranteed to provide privacy as long as there is an honest majority among the parties participating in the computation. When there is no honest majority, the honest parties will abort the protocol with overwhelming probability. The corrupted parties are active, meaning that they may arbitrarily deviate from the protocol. SCALE works in the pre-processing model. In this model, we have an offline phase, where input independent data is generated before the inputs are received. This data is then consumed in the online phase and allows us to perform our computation more efficiently.

Additions can be performed locally by each party by adding the respective shares. Multiplication, however, requires one round of communication, and comparisons require seven rounds of communication. Because performing MPC with a large number of parties becomes costly in terms of communication, the computation will be delegated to a small number of computational parties. To avoid collusions between them, these parties should have conflicting interests. In a P2P electricity market, the computational parties might be, for example, a supplier, an aggregator and the TP.

Public key encryption: a public key encryption scheme consists of three algorithms. The first one, $\mathsf{KeyGen}(1^\lambda)$, generates a public key and a private key, $(\mathsf{pk}, \mathsf{sk})$ with respect to a security parameter λ. $\mathsf{Enc}_{\mathsf{pk}}(m, r)$ uses the public key pk to encrypt the message m under randomness r and outputs a ciphertext c. Finally, $\mathsf{Dec}_{\mathsf{sk}}(c)$ uses the secret key sk to decrypt c and outputs m. The scheme is correct if $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m, r)) = m$ for any randomness r.

Functional and Privacy Requirements: The market must receive the users' orders, match them and inform users of their traded volume and price. By the end of the auction, no one should know the information contained in each order except for the user who submitted it, and the transactions to be performed by each user should only be revealed to them. The auction mechanism should allow orders coming from nearby locations to be matched first, and only then look through all the other orders.

3 Auction algorithms

3.1 Volume matching

In a volume matching algorithm, orders are matched according to their volume only, with the price per volume being defined beforehand. We provide a high-level description of the volume matching from [dGCP+21], followed by the changes introduced in this work.

Volume matching for dark pools [dGCP+21]: Each order i has the form $\mathsf{ord}_i = (\langle \mathsf{id}_i \rangle, \langle v_{i,j} \rangle, \langle b_i \rangle, \langle s_i \rangle)$. id_i represents the identity of the user and $v_{i,j}$ is the volume of the order written as a sequence of l bits, $j = 0, ..., l-1$. The bits b_i and s_i indicate the direction of the order: if $b_i = 1$, ord_i is a buy order; if $s_i = 1$, ord_i is a sell order; if $b_i = s_i = 0$, ord_i is a dummy order, meaning that it is neither a buy nor a sell order and the volume is zero by default. Dummy orders will be useful later on since users can hide their trading activity by submitting dummies to auctions where they do not wish to trade. Note that we cannot have $b_i = s_i = 1$ and orders in this form are discarded during the input phase. The algorithm has three phases:

1. The input phase, where the orders are received and checked to ensure all the components have the correct format as described above. Incorrect orders are discarded, and we obtain a list of n correct orders. For each of these orders, we obtain their actual volume in each direction by calculating $\langle v_i^b \rangle = b_i \cdot \sum_{j=0}^{l-1} v_{i,j} \cdot 2^j$ and $\langle v_i^s \rangle = s_i \cdot \sum_{j=0}^{l-1} v_{i,j} \cdot 2^j$.

2. The clearing phase one, where we check which direction has largest total volume. Suppose there is more total buy volume (the case with more sell volume is symmetric). All the sell orders will for sure be matched, so we open the sell volume shares $\langle v_i^s \rangle$ of every order. Note that if $v_i^s > 0$, then we know that order i is a sell order, and since we know it will be matched, we also open the $\langle \_i \rangle$ shares and remove the order from the list. If $v_i^s = 0$, order i might be either a buy or a dummy order, and we calculate the cumulative buy volume of the first i orders, $\langle w_i \rangle = \sum_{h=1}^{i} \langle v_h^b \rangle$. The total sell volume $\sigma = \sum_{i=1}^{n} \langle v_i^s \rangle$ is revealed.
3. The clearing phase two, where we open the volume shares in the direction with largest total volume. As before, suppose there is more total buy volume. We want to open the buy volume shares $\langle v_i^b \rangle$ to find out which of the buy orders are matched. However, we do not want to reveal any unmatched volume. To do so, we run a binary search to find the first index u such that $\langle w_u \rangle \geq \sigma$. The first $u - 1$ orders are opened, as they will be completely matched. Since part of the volume in order u might remain unmatched, we do not open $\langle v_u^b \rangle$. We simply subtract the volume $\sigma - \langle w_{u-1} \rangle$ from it and then open $\langle \mathsf{id}_u \rangle$.

By the end of the auction, we know exactly who traded what volume. We will now show how to process the orders such that no information is revealed.

Volume matching for the P2P electricity market: In this version of the algorithm, we assume we know the user associated with each order from the beginning. Each user submits one order to every auction, and this order will be a dummy whenever the user does not wish to trade any volume. Thus, knowing who sent each order reveals absolutely no information since everyone always has the same behaviour in every auction. Each order i has the form $\mathsf{ord}_i = (\mathsf{id}_i, \langle v_{i,j} \rangle, \langle b_i \rangle, \langle s_i \rangle)$. The order components have the same meaning as in [dGCP+21], the only difference being that id_i is not a secret shared value anymore. We now summarise the changes introduced in each phase of the algorithm. A formal description can be found in Figure 1.

1. The input phase will proceed in the same manner as before. We check that orders have the correct format and accepted orders move on to the clearing phase.
2. In clearing phase one, we check which direction has largest total volume. Suppose once again there is more total buy volume (the case with more sell volume is symmetric). Now, we cannot open the sell volume shares $\langle v_i^s \rangle$, or the exact amount of electricity each user is selling will become public. Instead, we simply inform every user that all their sell volume was matched (for some users this volume will be zero, in which case they can just ignore it). As before, the total sell volume $\sigma = \sum_{i=1}^{n} \langle v_i^s \rangle$ is revealed and we calculate the cumulative buy volume of the first i orders, $\langle w_i \rangle = s_i \cdot \sum_{h=1}^{i} \langle v_h^b \rangle$, for every order i.
3. In clearing phase two, we cannot simply use a binary search to find the last order with some matched volume, since doing so will reveal that whoever submitted this order wants to buy electricity. We avoid that by calculating the following values for each order i: $\langle w_i' \rangle$, which is 1 if $\langle w_i \rangle \leq \omega$ and 0 otherwise; $\langle w_i'' \rangle$, which is 0 except for the first i such that $\langle w_i' \rangle = 0$. We can now obtain the amount of matched buy volume in each order by calculating

$$\langle match_i^b \rangle = \langle v_i^b \rangle \cdot \langle w_i' \rangle + (\sigma - \langle w_{u-1} \rangle) \cdot \langle w_i'' \rangle$$

   This value is then revealed to the user id_i, who submitted ord_i.

```
Volume Matching

Input phase: On input $\mathsf{ord}_i = [\mathsf{id}_i, \langle v_{i,j} \rangle, \langle b_i \rangle, \langle s_i \rangle]$, where $\mathsf{id}, v_{i,j}, b_i, s_i \in \mathbb{F}_p$:
  1. $\langle t_i \rangle \leftarrow \sum_{j=0}^{\ell-1} (\alpha_{i,j} \cdot (\langle v_{i,j} \rangle \cdot \langle v_{i,j} \rangle - \langle v_{i,j} \rangle)) + \beta_{i,1} \cdot (\langle b_i \rangle \cdot \langle b_i \rangle - \langle b_i \rangle) + \beta_{i,2} \cdot (((\langle s_i \rangle \cdot \langle s_i \rangle - \langle s_i \rangle)) + \gamma_i \cdot ((\langle b_i \rangle \cdot \langle s_i \rangle),$
     for random $\alpha_{i,j}, \beta_{i,1}, \beta_{i,2}, \gamma_i$.
  2. $t_i \leftarrow \mathsf{Open}(\langle t_i \rangle)$
  3. If $t_i = 0$ then execute $\langle v_i^b \rangle \leftarrow \langle b_i \rangle \cdot \sum_{j=0}^{\ell-1} \langle v_{i,j} \rangle \cdot 2^j$ and $\langle v_i^s \rangle \leftarrow \langle s_i \rangle \cdot \sum_{j=0}^{\ell-1} \langle v_{i,j} \rangle \cdot 2^j$.
  4. If $t_i = 1$ then discard $\mathsf{ord}_i$.
  5. If $\rho = 0$, add $\mathsf{ord}_i \leftarrow (\mathsf{id}_i, \langle v_i^b \rangle, \langle v_i^s \rangle)$ to list $L$, for $i = 1, ..., n$.
  6. If $\rho > 0$, for $k = 0, ..., \rho - 1$ execute:
       I. $\langle p_i^{c_k} \rangle \leftarrow (1 - \langle p_i^{c_0} \rangle) \cdot ... \cdot (1 - \langle p_i^{c_{k-1}} \rangle) \cdot (1 - \langle v_{i,\ell-1} \rangle) \cdot ... \cdot (1 - \langle v_{i,\mathsf{idx}(c_k)+1} \rangle)$.
      II. $\langle v_{i+kn}^b \rangle \leftarrow \langle v_i^b \rangle \cdot \langle p_i^{c_k} \rangle$ and $\langle v_{i+kn}^s \rangle \leftarrow \langle v_i^s \rangle \cdot \langle p_i^{c_k} \rangle$
     III. Add $\mathsf{ord}_{i+kn} \leftarrow (\mathsf{id}_i, \langle v_{i+kn}^b \rangle, \langle v_{i+kn}^s \rangle)$ to the list $L$, for $i = 1, ..., \rho - 1$.

Clearing phase one: On input $L = [\mathsf{ord}_1, \ldots, \mathsf{ord}_n]$ (or $L = [\mathsf{ord}_1, \ldots, \mathsf{ord}_{\rho n}]$, if $\rho > 0$):
  1. $\langle g \rangle \leftarrow \sum_{i=1}^{n} \langle v_i^b \rangle - \langle v_i^s \rangle$
  2. $\langle h \rangle \leftarrow (\langle g \rangle > 0)$
  3. $h \leftarrow \mathsf{Open}(\langle h \rangle)$
  4. If $h = 1$, announce that all sell orders were matched and calculate $\langle w_i \rangle = \sum_{h=1}^{i} \langle v_h^b \rangle$ and
     $\sigma \leftarrow \mathsf{Open}\left(\sum_{i=1}^{n} \langle v_i^s \rangle\right)$.
     If $h = 0$, announce that all buy orders were matched and calculate $\langle w_i \rangle = \sum_{h=1}^{i} \langle v_h^s \rangle$ and
     $\sigma \leftarrow \mathsf{Open}\left(\sum_{i=1}^{n} \langle v_i^b \rangle\right)$.

Clearing phase two: On input $L$, assuming $h = 1$ (if $h = 0$, substitute every $b$ for $s$), execute for every $i$:
  1. $\langle w_i' \rangle \leftarrow (\langle w_i \rangle \leq \sigma)$.
  2. $\langle w_i'' \rangle \leftarrow \langle w_{i-1}' \rangle \cdot (1 - \langle w_i' \rangle)$.
  3. $\langle match_i^b \rangle \leftarrow \langle v_i^b \rangle \cdot \langle w_i' \rangle + (\sigma - \langle w_{i-1} \rangle) \cdot \langle w_i'' \rangle$.
  4. If $\rho > 0$, $\langle match_i^b \rangle \leftarrow \sum_{k=0}^{\rho-1} \langle match_{i+kn}^b \rangle$.
```

Figure 1: Volume Matching

With these changes, the outcome of the auction is not public anymore, with each user only knowing the outcome of their own order. However, geographical prioritization is still missing. To introduce this feature, we simply divide the users into neighbourhoods (not necessarily the same as the ones already defined for many cities), with each neighbourhood running its own auction. At the end of each intra-neighbourhood auction, we know that for each neighbourhood the orders in the direction with the largest total volume still have some volume left to trade. In case there are neighbourhoods with leftover sell volume, as well as neighbourhoods with leftover buy volume, we can run an inter-neighbourhood auction, with orders from all the neighbourhoods. Note that we do not need to run the order format check again, since all the orders were already confirmed to have the correct format.

This new auction between neighbourhoods will proceed exactly the same way as before, except that the input is slightly different. Assume neighbourhood X has some leftover buy volume, while neighbourhood Y has some leftover sell volume. Then, we create a sell list with orders $\mathsf{ord}_i^s = (\mathsf{id}_i^s, \langle v_i^s \rangle)$, and a buy list with orders $\mathsf{ord}_i^b = (\mathsf{id}_i^b, \langle v_i^b \rangle)$. For the same index i, we have an id associated with the sell volume, and a different id associated with the buy volume, being that both these volumes might be positive. Users will receive the output of the auction only for their corresponding order direction.

To maximise social welfare, we can introduce further changes to the volume matching algorithm such that orders of low volume are prioritized. A possible way of achieving this is through secure sorting, where we would sort the orders by ascending volume before matching them. However, secure sorting algorithms are expensive. Therefore, we choose to divide orders into ρ size categories, with each order belonging to a single category. If we do not wish to prioritize smaller orders, we set $\rho = 0$. Each size category c_k, for $k = 0, ..., \rho - 1$, contains orders ord_i with volume $v_{c_{k-1}} < v_i \leq v_{c_k}$, except for the last category which contains all the remaining orders. The orders in the lower volume category will be matched first, and then we proceed to the next categories until we reach the category of highest volume orders. Within the same category, orders that arrived earlier are matched first.

Recall that the submitted volume v_i is written using l bits $v_{i,j}$. We will use this to place each order in the correct size category using secure multiplications instead of the more expensive secure comparison operation. To do so, we associate with each size category c_k an index $\mathsf{idx}(c_k)$ between 0 and $l-1$. Category c_k will then contain orders i where the volume's most significant bit is $v_{i,j}$ with $\mathsf{idx}(c_{k-1}) < j < \mathsf{idx}(c_k)$. For example, suppose $\mathsf{idx}(c_0) = 1$ and $\mathsf{idx}(c_1) = 3$. Then, category c_0 contains orders of volume up to 3, while category c_1 contains orders of volume from 4 up to 15. The highest volume category $c_{\rho-1}$ will have $\mathsf{idx}(c_{\rho-1}) = l - 1$. For each order i, we calculate $\langle p_i^{c_k} \rangle$, which is 1 if ord_i is in the size category c_k and 0 otherwise. Then, if we run the auction as if we had $\rho \cdot n$ orders, with the volume of order $j = i + kn$ being given by $\langle v_i \rangle \cdot \langle p_i^{c_k} \rangle$. The volume traded for order i in the n order auction is then calculated by adding the volume traded for the orders $j = i + kn$, with $k = 0, ..., \rho - 1$, in the $\rho \cdot n$ order auction.

One must be careful when using size categories with inter-neighbourhood auctions. If the number of size categories ρ and the volume limits v_{c_k} of each category are the same as in all of the intra-neighbourhood auctions, we do not need to run the input phase again. For the orders from neighbourhoods where the intra-neighbourhood auction had different ρ or v_{c_k}, we recalculate $\langle p_i^{c_k} \rangle$.

3.2 Continuous double auction

In a continuous double auction, every order contains both a price and a volume to trade. There is a limit order book (LOB), which consists of a list of buy orders sorted by price descending, and a list of sell orders sorted by price ascending. Note that on top of each list we will have the orders with the best prices : the highest price for buy orders, and the lowest price for sell orders. When a new order arrives, we process it immediately by checking if the price is compatible with the best price in the list of orders in the opposite direction. Matches are executed until the new order has no more volume to trade, or until the prices do not allow it. The new order is then placed in the corresponding list of the LOB.

We provide a high-level description of the continuous double auction from [CSTA19], followed by the changes introduced in this work. Throughout this section, we assume that the new arriving order is a buy order. The operation for a new sell order is symmetric.

CDA for dark pools [CSTA19]: Each order i has the form $\mathsf{ord}_i^{\mathsf{dir}} = \left(\langle \mathsf{id}_i^{\mathsf{dir}} \rangle, \mathsf{dir}, \langle v_i^{\mathsf{dir}} \rangle, \langle p_i^{\mathsf{dir}} \rangle\right)$. id_i represents the identity of the user, dir is a public value indicating whether the order is buy or sell, v_i is the volume of the order and p_i the associated price. In the LOB we have a list of M buy orders $B = \left(\langle \mathsf{id}_i^b \rangle, \mathsf{buy}, \langle v_i^b \rangle, \langle p_i^b \rangle\right)_{i=1}^{M}$ and a list of N sell orders $S = (\langle \mathsf{id}_i^s \rangle, \mathsf{sell}, \langle v_i^s \rangle, \langle p_i^s \rangle)_{i=1}^{N}$. The algorithm can be divided into three procedures:

1. Processing the sell list : suppose a new buy order $\left(\langle \mathsf{id}_0^b \rangle, \mathsf{buy}, \langle v_0^b \rangle, \langle p_0^b \rangle\right)$ is submitted. We securely check whether the new order has any volume left to trade and the price is high enough to match the top sell order ord_1^s. If both of these conditions hold, we calculate $\langle v_0^b \rangle \geq \langle v_1^s \rangle$ and use the output of the computation to obtain the volume to be traded and to know which of the orders was fully matched. We subtract the traded volume from each order and, if the sell order was fully matched, we run the procedure for reorganizing the sell list. The traded volume, the sell price and id_1^s are revealed. This process is repeated for each order in the sell list until the buy order has no more volume to trade, the prices are not compatible or the sell list is empty.
2. Inserting the new order into the buy list : the new order is inserted in the buy list even if all of its volume was already matched. In this case, we change the new order price to zero, such that it will be introduced at the end of the list. Then, we go through the buy list and compute $\langle p_0^b \rangle \leq \langle p_i^b \rangle$ for every buy order i already in the LOB. The result of these comparisons is then used to place the new order such that the buy list is still sorted by price descending.
3. Reorganizing the sell list : if a sell order is fully matched, we reorganize the sell list by simply moving every order one place up in the list.

CDA for the P2P electricity market: Each order i has the form $\mathsf{ord}_i^{\mathsf{dir}} = \left(\langle \mathsf{id}_i^{\mathsf{dir}} \rangle, \mathsf{dir}, \langle v_i^{\mathsf{dir}} \rangle, \langle p_i^{\mathsf{dir}} \rangle, \langle x_i^{\mathsf{dir}} \rangle, \langle y_i^{\mathsf{dir}} \rangle\right)$. The new order components $\langle x_i^{\mathsf{dir}} \rangle, \langle y_i^{\mathsf{dir}} \rangle$ are the coordinates of the user submitting the order. For each new arriving order we divide the orders in the opposite direction list into ω categories, with each order belonging to a single category. Each of these categories o_k, for $k = 0, ..., \omega - 1$, contains orders $\mathsf{ord}_i^{\mathsf{dir}}$ with distance $d_{o_{k-1}} < d_i \leq d_{o_k}$, except for the last category which contains all the remaining orders. Orders in the first distance category will be matched first before we proceed to the second category, and so on. Within the same category, orders with a better price are matched first. As before, the LOB contains a list of buy orders sorted by price descending and a list of sell orders sorted by price ascending. We now summarise the changes introduced in each part of the algorithm. A formal description can be found in Figure 2.

1. Processing the sell list : suppose a new buy order $\left(\langle \mathsf{id}_0^b \rangle, \mathsf{buy}, \langle v_0^b \rangle, \langle p_0^b \rangle, \langle x_0^b \rangle, \langle y_0^b \rangle\right)$ is submitted. We first calculate the square of the distance between the new order and every order i in the

```
Continuous Double Auction

Suppose in the LOB there is a list of buy orders $B = \big(\langle \mathsf{id}_i^b\rangle, \mathsf{buy}, \langle v_i^b\rangle, \langle p_i^b\rangle, \langle x_i^b\rangle, \langle y_i^b\rangle\big)_{i=1}^M$ and a list of sell orders
  $S = (\langle \mathsf{id}_i^s\rangle, \mathsf{sell}, \langle v_i^s\rangle, \langle p_i^s\rangle, \langle x_i^s\rangle, \langle y_i^s\rangle)_{i=1}^N$.

Process sell list: On input $\big(\langle \mathsf{id}_0^b\rangle, \mathsf{buy}, \langle v_0^b\rangle, \langle p_0^b\rangle, \langle x_0^b\rangle, \langle y_0^b\rangle\big)$:
  1. $\langle d_i\rangle \leftarrow (\langle x_i^s\rangle - \langle x_i^b\rangle)^2 + (\langle y_i^s\rangle - \langle y_i^b\rangle)^2$, for $i = 1, ..., N$.
  2. For $i = 1, ..., N$, do:
       I. For $k = 0, ..., \omega - 2$, $\langle q_i^{o_k}\rangle \leftarrow (1 - \langle q_i^{o_0}\rangle) \cdot ... \cdot (1 - \langle q_i^{o_{k-1}}\rangle) \cdot (\langle d_i\rangle < \langle d_{o_k}\rangle)$.
       II. $\langle q_i^{o_{\omega-1}}\rangle \leftarrow (1 - \langle q_i^{o_0}\rangle) \cdot ... \cdot (1 - \langle q_i^{o_{\omega-2}}\rangle)$.
  3. For $i = 1, ..., N$ and $k = 0, ..., \omega - 1$, do:
       I. $\langle isRich\rangle \leftarrow \langle p_0^b\rangle \geq \langle p_i^s\rangle$.
       II. $\langle hasVol\rangle \leftarrow \langle v_0^b\rangle > 0$.
       III. $\langle isTrade\rangle \leftarrow \langle isRich\rangle \cdot \langle hasVol\rangle \cdot \langle q_i^{o_k}\rangle$.
       IV. $\langle clear_s\rangle \leftarrow (\langle v_0^b\rangle \geq \langle v_i^s\rangle)$.
       V. $\langle \alpha_i\rangle \leftarrow \langle \alpha_i\rangle + \langle clear_s\rangle$.
       VI. $\langle t\rangle \leftarrow \langle clear_s\rangle \cdot (\langle v_i^s\rangle - \langle v_0^b\rangle) + \langle v_0^b\rangle$.
       VII. $\langle trade_i\rangle \leftarrow (\langle \mathsf{id}_0^b\rangle, \langle \mathsf{id}_i^s\rangle, \langle t\rangle, \langle p_0^b\rangle \cdot \langle t\rangle)$.
  4. Open $\langle trade_i\rangle$ and $\langle \alpha_i\rangle$ for $i = 1, ..., N$.
  5. $\langle e\rangle \leftarrow (\langle v_0^b\rangle = 0)$.
  6. $\langle p_0^b\rangle \leftarrow \langle p_0^b\rangle \cdot (1 - \langle e\rangle)$.

Insert into buy list: Identical to the algorithm in [CSTA19, Figure 2].

Reorganize sell list: For $i = 1, ..., N$ do:
  1. If $\alpha_i = 0$, put sell order $i$ in the next position of the new sell list. If $\alpha_i = 1$, do nothing.
```

Figure 2: Continuous Double Auction

sell list: iterate through the distance categories, and for each of them, check if order i is in the current category, if the new order has any volume left to trade and if $\langle p_0^b\rangle \geq \langle p_i^s\rangle$. If all of these conditions hold, there will be a trade. However, we do not reveal this. Instead, we proceed as if there is always a trade, by registering the traded volume in a transaction list, updating the volumes of the orders and keeping track of the indexes of the totally cleared sell orders. Whenever the conditions do not hold, the traded volume will be zero and the volumes of the orders remain the same.

2. Inserting the new order into the buy list : exactly as in [CSTA19].
3. Reorganizing the sell list : as before, we know how many orders were completely matched. However, because of the priority given to orders from closer users, these orders might not be the ones at the top of the sell list. To remove the correct orders, we use the list of indexes of the cleared sell orders generated during the sell list processing. This means that instead of removing sell orders as soon as they are matched, we wait until we finish processing the matches for the new buy order. This way, even if we know the sell list position of all the cleared orders, we do not know which of them were cleared first.

Note that the algorithm in Figure 2 still includes publicly revealing the output of the auction in Step 4. This can be avoided by replacing the user id shares $\langle \mathsf{id}_i\rangle$ with shares of the public key of each user, $\langle \mathsf{pk}_i\rangle$. In Step 4, pk_i is revealed, and each computational party uses it to encrypt their own shares of the corresponding transaction information. The encrypted transaction shares are then broadcast, and only the user with the associated secret key sk_i will be able to reconstruct the information. Users should generate a new key pair $(\mathsf{pk}, \mathsf{sk})$ for each submitted order, and should not announce their public key or use it elsewhere. This way, orders will be anonymous and it will be impossible to link orders submitted by the same person.

This algorithm leaks whether an order is buy or sell, as well as which of the orders in the LOB are completely cleared by each new order. However, we do not know who submitted any of these orders.

4 Runtimes

The runtimes for both algorithms are presented below. We assume that the offline phase is run when the market is closed and present runtimes for the online phase only.

4.1 Setting

We used SCALE-MAMBA with 3 parties using Shamir secret sharing. All the parties run identical machines with an Intel i-9900 CPU and 128GB of RAM. The ping time between the machines is 1.003 ms.

4.2 Online phase of volume matching

The runtime of the input phase depends on the length of the bit sequence representing the volume of each order. In [dGCP+21], only the runtime for 32-bit sequences was presented. However, for the intraday P2P electricity market, 16-bit or even 8-bit sequences could be used, either because we do not expect orders with very high volume in the intraday market, or because we want to enforce a cap on the volume that can be submitted. Additionally, when prioritizing smaller users there is an extra step in the input phase, corresponding to the generation of the $p_i^{c_k}$ values. Runtimes for the input phase with different bit lengths l and and numbers of size categories ρ are presented in Table 1.

Table 1: Runtimes in seconds for the input phase of the algorithm in Figure 1. l is the number of bits of the input volume and ρ is the number of size categories.

l	$\rho = 0$	$\rho = 3$	$\rho = 5$
8	0.00037	0.00067	0.00067
16	0.00047	0.00128	0.00207
32	0.00062	0.00249	0.00427

Runtimes for the clearing phases as well as total runtimes are presented in Table 2. While in [dGCP+21] the runtime of the clearing phases depends on the number of dummy orders and the number of matched orders, that is not the case here. The fact that we do not publicly reveal the auction output in our version of the algorithm results in more expensive clearing phases. In [dGCP+21], the input runtime was at least one order of magnitude above the runtime for the clearing phases, while in our algorithm the clearings do take more time than the input phase, especially when using size categories. Even with the increased runtime, this algorithm still has a good performance, with 100 thousand orders being matched in 1.5 minutes when using no size categories, or in just over 6 minutes when using five size categories. Since we know exactly how many orders to expect, we can choose when to stop accepting orders for a given consumption period accordingly.

Suppose we want to match orders coming from twenty neighbourhoods, each of them with 10 thousand users, using three size categories in the intra-neighbourhood auction and no size categories in the inter-neighbourhood auction. The orders' volumes are 8-bit sequences. The intra-neighbourhood auctions are run in parallel, taking 24 seconds. Suppose ten of the neighbourhoods are left with unmatched buy volume, while the other ten neighbourhoods are left with unmatched sell volume. Then, the inter- neighbourhood auction will be run as an auction on 100 thousand orders. Since we do not need to run the input phase again, it takes 171 seconds (2.85 minutes). The total runtime for the two auctions is 195 seconds (3.25 minutes).

Table 2: Runtimes in seconds for algorithm in Figure 1. Total runtimes correspond to the clearing phases runtimes plus the input phase runtimes for the respective number of orders and using 8-bit volume entries. n is the number of orders and ρ the number of size categories.

n	$\rho = 0$		$\rho = 3$		$\rho = 5$	
	Clearings	Total	Clearings	Total	Clearings	Total
10	0.0061	0.0098	0.018	0.025	0.030	0.039
100	0.055	0.092	0.17	0.24	0.28	0.38
1000	0.56	0.93	1.7	2.4	2.9	3.9
10000	5.4	9.0	17	24	28	38
100000	54	90	171	239	275	374

4.3 Online phase of CDA

Runtimes for each phase of the CDA algorithm with two distance categories are presented in Table 3. Runtimes for each phase of the CDA algorithm with three and five distance categories are presented in Table 4. The initial sharing of the inputs and the list reorganizing procedure are independent of the number of distance categories, and also very fast, barely affecting the total runtimes. In fact, 70% to 90% of the total runtime comes from processing the matches, which depends only on one of the lists. Therefore, we might have orders that take much longer to process than others whenever the buy and sell lists are unbalanced.

Note that even when using only two distance categories, the time for processing each new order becomes considerably high as the number of orders in the buy and sell lists grows. For example, after $M = N = 1000$, it will take over an hour to process the next 1000 orders, assuming that M and N stay approximately the same. Each new order is always inserted into the corresponding list, and so this list will grow. In the meantime, the other list might either decrease or stay the same, depending on whether some of the orders are totally cleared or not. Depending on the balance between the number of submitted buy and sell orders and their prices, it might be possible to maintain the lists relatively small even when receiving many orders. However, to ensure that the auction is completed on time, this algorithm should not be used in situations where we require a high throughput.

Table 3: Runtimes in seconds for the algorithm in Figure 2 with $\omega = 2$ distance categories. M and N represent the sizes of the buy and sell lists, respectively.

M, N	Share inputs	Reorganize list	Process matches	Insert into list	Total
10	0.003	0.00005	0.034	0.009	0.047
50	0.006	0.00028	0.164	0.040	0.210
100	0.011	0.00052	0.313	0.075	0.400
500	0.047	0.00272	1.640	0.379	2.069
1000	0.093	0.00541	3.282	0.757	4.136

Table 4: Runtimes in seconds for the algorithm in Figure 2. ω is the number of distance categories, and M and N represent the sizes of the buy and sell lists, respectively. he runtimes for sharing the inputs and reorganizing the list are presented in Table 4.

M, N	$\omega = 3$			$\omega = 5$		
	Process matches	Insert into list	Total	Process matches	Insert into list	Total
10	0.052	0.009	0.064	0.089	0.009	0.102
50	0.264	0.041	0.312	0.412	0.038	0.457
100	0.502	0.076	0.590	0.837	0.076	0.924
500	2.514	0.380	2.944	4.298	0.384	4.732
1000	5.156	0.775	6.029	8.500	0.755	9.352

5 Conclusion

In this paper, we propose two auction mechanisms for the intraday P2P electricity market. These algorithms use MPC to guarantee that the information contained in the users' orders and the auction results remains private, while also allowing the prioritization of close neighbours when matching the orders. Simpler algorithms will generally be faster, but by taking into account which operations are more expensive with MPC it is possible to obtain auction algorithms with the desired functionalities and maintain appropriate runtimes for our use case.

The volume match algorithm can quickly process large amounts of orders, even when prioritizing low volume orders. Since all the users always submit one order to every auction, we know exactly how many orders to expect and can adapt the sizes of the considered neighbourhoods as well as the window for submitting orders to each time period accordingly.

The CDA has the advantage of considering the exact location of each order instead of relying on predefined neighbourhoods for geographical prioritization. Additionally, it also takes into account the prices at which users want to trade, instead of using a common predefined price per volume unit. However, its performance deteriorates considerably as the limit order book grows. To ensure it remains usable, the number of orders submitted to this auction should be limited, either by implementing it only in areas with low population density or by closing the auction as soon as the LOB reaches a certain size.

To extend this work, it would be relevant to develop a privacy-preserving billing and settlements protocol adapted to the proposed auction mechanisms and test the efficiency of the combined system. This would ensure that the full process of intraday P2P electricity trading can take place in a privacy- preserving manner.

Acknowledgments

This work was supported in part by the Research Council KU-Leuven : C16/15/058, by the Research Foundation - Flanders through FWO SBO project SNIPPET S007619, and by CyberSecurity Research Flanders with reference number VR20192203. Additionally, the first author is supported by ERC Advanced Grant ERC-2015-AdG-IMPaCT.

References

AACM16. Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A. Mustafa. An mpc-based privacy-preserving protocol for a local electricity trading market. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security*, pages 615–625, Cham, 2016. Springer International Publishing.

ACC$^+$21. Abdelrahaman Aly, Kelong Cong, Daniele Cozzo, Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Oliver Scherer, Peter Scholl, Nigel P. Smart, Titouan Tanguy, and Tim Wood. SCALE-MAMBA v1.12: Documentation, 2021.

CSTA19. John Cartlidge, Nigel P. Smart, and Younes Talibi Alaoui. Mpc joins the dark side. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, pages 148–159, New York, NY, USA, 2019. Association for Computing Machinery.

CSTA22. John Cartlidge, Nigel P. Smart, and Younes Talibi Alaoui. Multi-party computation mechanism for anonymous equity block trading: A secure implementation of turquoise plato uncross. *Int. J. Intell. Syst. Account. Financ. Manage.*, 28(4):239–267, mar 2022.

dGCP$^+$21. Mariana Botelho da Gama, John Cartlidge, Antigoni Polychroniadou, Nigel P. Smart, and Younes Talibi Alaoui. Kicking-the-bucket: Fast privacy-preserving trading using buckets. *IACR Cryptol. ePrint Arch.*, page 1549, 2021.

MCA16. Mustafa A. Mustafa, Sara Cleemput, and Aysajan Abidin. A local electricity trading market: Security analysis. 10 2016.

SHPB12. Walid Saad, Zhu Han, H. Vincent Poor, and Tamer Basar. Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications. *IEEE Signal Processing Magazine*, 29(5):86–105, 2012.

SSAA21. Roozbeh Sarenche, Mahmoud Salmasizadeh, Mohammad Ameri, and Mohammad Aref. A secure and privacy-preserving protocol for holding double auctions in smart grid. *Information Sciences*, 557:108–129, 05 2021.

tar. Feed-in tariff (fit) rates. ttps://www.ofgem.gov.uk/environmental-programmes/fit/fit-tariff-rates. Accessed: Jan. 11, 2022.

ver. Vergoeding overtollige elektriciteit? http://www.vreg.be/nl/vergoeding-overtollige-elektriciteit. Accessed: Jan. 11, 2022.

ZGND22. Fairouz Zobiri, Mariana Gama, Svetla Nikova, and Geert Deconinck. A privacy-preserving three-step demand response market using multi-party computation. In *2022 IEEE PES Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, 2022.