Practically Efficient Secure Single-Commodity Multi-Market Auctions

Abdelrahaman Aly and Mathieu Van Vyve

Université catholique de Louvain, CORE
Voie du Roman Pays 34, B-1348 Louvain-la-Neuve (Belgium)

Abstract. We study the problem of securely building single-commodity multi-markets auction mechanisms. We introduce a novel greedy algorithm and its corresponding privacy preserving implementation using secure multiparty computation. More specifically, we determine the quantity of supply and demand bids maximizing welfare. Each bid is attached to a specific market, but exchanges between different markets are allowed up to some upper limit. The general goal is for the players to bid their intended valuations without concerns about what the other players can learn. This problem is inspired by day-ahead electricity markets where there are substantial transmission capacity between the different markets, but applies to other commodity markets like gas. Furthermore, we provide computational results with a specific C++ implementation of our algorithm and the necessary MPC primitives. We can solve problems of 1945 bids and 4 markets in 1280 seconds when online/offline phases are considered. Finally, we report on possible set-ups, workload distributions and possible trade-offs for real-life applications of our results based on this experimentation and prototyping.

1 Introduction

Auctions have been proved to be economically efficient under many settings [1]. In recent years, with the advent of larger scale markets e.g. online commerce and commodities, factors like secrecy, integrity and fairness have become more important. Parties need adequate incentives to bid truthfully, without the risk of loosing competitive advantages in future interactions. Our aim is to solve the problem where a commodity is to be transported between the different markets up to a given capacity limit. In our setting, buyers, sellers, markets and control agencies may have to interact in a competitive environment where information about prices and volume can reveal much more than what any party is willing to disclose. In the day-ahead electricity markets in Europe, this reluctance delayed by several years the integration of the national markets, until it was actually imposed by the European authorities (Directives 2005/89 and 2003/54 and Regulation 1228/2003).

Traditional solutions include a neutral third party in charge of all computations and responsible to exert secrecy, integrity an fairness in his own processes.

However, such a third party is in general hard to find, and would concentrate all attacks making it a single vulnerable failure point.

We report on a mechanism where this third party can be replaced. Indeed, our *virtual third party* uses Secure Multiparty Computation (MPC) and can be composed of any subset of players. MPC is a secure mechanism that allows several players to compute a function in a distributed environment. From Yao's original result in 1982 [2], to the current state of the art, secure multiparty computation has evolved from a theoretical object of study, to a field that is used in real life applications. MPC offers a variety of techniques, primitives and applications that provide security under diverse models, and in a distributed environment.

1.1 Our Contribution

We introduce a novel greedy algorithm and its secure formulation, for auctions with several geographical markets where exchange between them is possible. We analyze and introduce variations and trade-offs of these building blocks to obtain efficient running times, addressing the privacy-preserving protocol implementation and its security and performance constraints. Additionally, we report on computational experimentation using historical data from electricity markets. To the best of our knowledge, this is the first time the problem of secure single commodities multi-market auctions with transmission constraints has been addressed in detail. We focus our attention on the following aspects:

Algorithm Design. Although this is a standard problem, we describe a novel greedy algorithm to compute its solution. This algorithm is better suited for its adaptation to secure multiparty computation. We give proofs of its correctness and that its MPC version is secure. Also since in practice the number of markets is limited (e.g. a few) but the number of bids can be large (e.g. a few thousands) we have aimed at keeping the complexity of the algorithm (close to) linear in the number of bids.

Complexity and Efficiency. As similar works in the field, we use communicational rounds (exchange of messages between parties involved in the computation) as the complexity measurement unit in our secure protocol. This is in line with our interest in practical use. Moreover, our general aim of minimizing the use of comparisons because of the constants associated to their computation. To facilitate reading, we abstract from our complexity analysis the cost associated to message exchange. Indeed, as in related works, this allow us to decouple our algorithm analysis from the sharing mechanism and the different implications linked to changes in the number of computational players.

Implementation. We have implemented our algorithm in C++, building from scratch our own modular MPC framework. Indeed, we could not find an open and efficient implementation suited to our need. We use NTL (Number Theory Library) [3] and GMP (GNU Multiple Precision Library) as external libraries. This implementation enables us to show that the algorithm we propose is capable of treating close to 2000 bids from 4 markets in a time that is practically relevant for our motivating application (20 minutes).

1.2 Related Work

Secure Auctions have been studied from different perspectives, both in terms of security, computational and economic efficiency. In all cases questions on topics like performance, fairness and integrity have been raised. In this section we cover some of the works with similar characteristics and explore their differences with our contributions.

Auctions with Secure Multiparty Computation. Bogetoft et al. [4] consider the problem of a real-life auction with secure multiparty computation. In their setting, Danisco, the only sugar beet processor of the danish market, and several thousand farmers settled clearance market prices in a secret and distributed fashion using MPC. They provide a secure MPC protocol for this single market application. In this paper, we explore a different setting, where there are several markets and each pair can exchange the commodity up to a given capacity. This setting is realistic for other types of commodities e.g. power and gas markets. Additionally, they built their protocols using VIFF [5], which proved to be reliable for the size of their problem. However, previous results for similar problems [6] suggest that this does not scale up very well. We describe the behavior of a dedicated implementation, using the flexibility of C++ and OOP, that uses a compact set of secure MPC primitives to provide security and efficiency.

Secure Auction Mechanisms with Secret Sharing. Several authors have studied the properties of secure auctions with secret sharing e.g. [7,8,9,10]. These works explore several different auction mechanisms in various environments. Recently, Nojoumian and Stinson [11] introduced algorithms for second-price and combinatorial auctions. Their protocols offer security against active and passive adversaries, using amongst others, Shamir secret sharing [12] and a verifiable secret sharing schemes (VSS). They model their auction problems as graph problems, and devise theoretically efficient algorithms, but no computational experimentation is reported.

Second Price Auctions. Some authors have considered cryptographic alternatives to guarantee security in second price auctions. Catane and Herzerg [13] propose trusting a supervising entity to perform the computations and using randomization. Their goal is to keep the bids secret from other players. Our privacy-preserving protocol provides security and fairness without relying on any third party. Also, their approach does not take into account the transmission exchanges that are essential for our model. Similar to [4], this solution would work for one market but needs to be adapted for a multi-market scenario.

1.3 Overview of the paper

The paper is organized as follows: Section 2 introduces the problem and some necessary concepts. Section 3 provides its network flow formulation and describes a novel polynomial-time algorithm for it that can be easily adapted to provide properties like data obliviousness. In Section 4, we describe the security model, building blocks and technical tools for later use in our secure protocol in the context of our secure algorithm. Section 5 describes our main protocol to

solve the problem. We analyze complexity, security and correctness. Experimentation and prototyping are described in Section 6.

2 Problem Overview

2.1 Auction Mechanism

The process we consider here is a reverse auction with several sellers or bidders. Markets or auctioneers adjudicate orders to supply and demand bids that maximize social welfare, while respecting the capacities of the transmission network. A control agency may be part of the process, to supervise and guarantee the integrity of the result. The security follows from the use of secure multiparty computation. Individual interests and involvement level are the following:

Markets and Transmission Network: The set of markets and the capacity of the transmission network are assumed to be public. The transmission network is represented by a capacitated network flow, i.e. pairs of markets are binded by bidirectional transmission lines. Each transmission line has an upper limit (i.e. capacity). Notice that in this case, markets are geographically separated.
Sellers/Buyers or Bidders: The set of players interested in acquiring or selling the commodity submit bids. Each bid is attached to a specific market. Each bidder can submit more than one bid, and to different markets. Bids are composed by a certain quantity Q (positive for buying and negative for selling) and a limit price P. All bids are enclosed and final i.e. no re-bidding is allowed. The bid placed by the player can be partially or totally adjudicated to the bidder depending on what maximizes social welfare. One of their interests is the secrecy of the information contained on each bid towards any other player e.g. other bidders and markets, for as long as the auction takes place. Their concerns are also correctness (the result of the auction is correct) and fairness (all players receive the same information at the same time, and are treated equally).
Automated Auctioneer: Is the proxy entity in charge of managing the auction. Our work proposes that the role of the auctioneer is to be taken by the computational parties representing markets, bidders and control agencies, in a distributed and secure fashion. This creates a virtual ideal functionality capable of determining the set of accepted supply and demand bids, guaranteeing correctness, without disclosing sensitive data.
Control Agency: Is a regulatory entity or any institution trusted by the Markets operators and Bidders. By the parties choosing, or environmental enforcement, it participates to add confidence to the process. Because of the nature of MPC, our secure protocol allows active participation of the Control Agency as a computational party, so that their presence would be necessary for the correct and secure operation of the protocols in conjunction with the model. The presence of a Control Agency remains optional.

On Computational parties. It is possible to have as many computational parties as considered necessary by the algorithm designer to guaranty security and bring confidence to the process. Although many of the building blocks require a minimum of three parties, the algorithm itself can be adapted to be used with

two-party computation. As stated many auctions require the presence of an external supervisor. A basic configuration would include a computational party representing the bidders, another the markets, and a third one for the supervisor or control agency. Another logical set-up would is to have one party for each geographical market. A larger number of computational parties can increase security and trust, but will negatively influence the performance.

2.2 Problem Definition

Formally, participants in the auction submit bids of the form (p_i, q_i, m_i) where p_i is the limit price, q_i is the quantity (positive for demand bid, negative for supply bid) and m_i is the market where the bid is submitted. Bids can be adjudicated partially, completely or not at all. The network operator also provides a capacity matrix C, where entry $C_{i,j}$ is the maximum amount that can be shipped from market i to market j. Note that this is similar to [14,15]. The goal is to adjudicate bids so that **(i)** social welfare is maximized, **(ii)** the exchanges implied can be executed on the network, **(iii)** the information contained in all bids is to be kept secret from other players until the end of the auction process. The network (its topology and the capacities) is assumed to be public. Another practical requirement is that the computations should not take more than, for instance, 30 minutes. Note that we do not associate costs to the transmission network.

Input Data. Data is provided as integer values over a finite field $\mathbb{Z}_q$ where input values are much smaller than q such that no overflow occurs. Its size is tied to the application in hand. Note that when they are secretly shared we can not differentiate between a demand bid and a supply bid.

2.3 Problem Formulation

Let us denote by $N = \{1, \ldots, n\}$ the set of all bids, and $K \cup D = N$ the partition into supply and demand bids respectively, $M = \{1, \ldots, m\}$ the set of markets, $K_j \in K$ and $D_j \in D$ the set of supply and demand bids respectively at market j. We define the nonnegative decision variable $\bar{x}_i \ \forall i \in N$ as the accepted quantity of bid i and variables $f_{i,j}$ as the flow on the line $(i, j) \in L = M \times M$ with capacity $C_{i,j}$. The problem can then be formulated as the following linear optimization problem:

$$\max \sum_{i \in D} p_i \bar{x}_i - \sum_{i \in K} p_i \bar{x}_i \tag{1}$$

$$\text{s.t.} \sum_{i \in K_j} \bar{x}_i + \sum_{i:(i,j) \in L} f_{i,j} = \sum_{i \in D_m} \bar{x}_i + \sum_{i:(j,i) \in L} f_{j,i} \quad \forall j \in M \tag{2}$$

$$0 \leq f_{i,j} \leq C_{i,j} \quad \forall (i,j) \in L \tag{3}$$

$$0 \leq \bar{x}_i \leq |q_i| \quad \forall i \in B \tag{4}$$

Note that by complementing demand bids ($x_i = q_i - \bar{x}_i$ for $i \in D$) and keeping supply bids as is ($x_i = \bar{x}_i$ for $i \in K$), one obtains an equivalent formulation

involving supply bids only (dropping the constant in the objective):

$$\min \sum_{i \in N} p_i x_i \tag{5}$$

$$\text{s.t.} \sum_{i \in K_j \cup D_j} x_i + \sum_{i:(i,j) \in L} f_{i,j} - \sum_{i:(j,i) \in L} f_{j,i} = \sum_{i \in D_j} q_i \quad \forall j \in M \tag{6}$$

$$0 \leq f_{i,j} \leq C_{i,j} \quad \forall (i,j) \in L \tag{7}$$

$$0 \leq x_i \leq |q_i| \quad \forall i \in B \tag{8}$$

Note that in this version, there is an external demand of $T_j = \sum_{i \in D_j} q_i$ to be met at each market j. The goal is to find the cheapest set of supply bids to satisfy these demands. Having supply bids only makes the description of the algorithm simpler. The is therefore the form that we will use in the rest of the text.

3 Network Flow Formulation

The problem (5) - (8) can actually be seen as a minimum cost capacitated network flow problem (MCF) on the graph $G = (V, A)$ as shown at Figure 3.

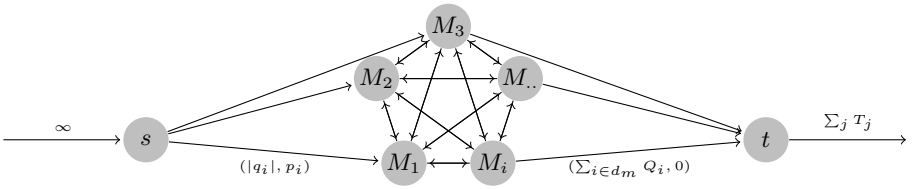

Fig. 1. The auction problem as a Minimum Cost Network Flow problem

Formally, the set of vertices is $V = M \cup \{s, t\}$ where s and t are artificial source and sink vertices. For each bid (supply and demand) $i \in K \cup D$, there is an arc (s, m_i) where m_i is the market of the bid i, with capacity $|q_i|$ and cost p_i. For each pair of markets $(i, j) \in L$ there is an edge between the respective vertices with capacity $C_{i,j}$ and no cost. For each market j, there is an edge (j, t) with capacity $T_j = \sum_{i \in d_j} q_i$ and zero cost. Finally there is (given) external flow arriving at vertex s and a given external flow leaving vertex t, both of magnitude $\sum_j T_j$. The associated minimum cost flow problem is obviously equivalent to the linear program (5) - (8).

Secure protocols to solve the MCF problem have been described by Aly and Van Vyve [6]. They provide secure polynomial-time algorithms. Although the protocol is theoretically efficient, in practice, its applicability seems to be limited by the high degree ($|V|^{10}$) of the polynomial in the complexity bound. Their computational experiments, using an implementation over VIFF [5], indicates that it would take around a year to solve the problem with perfect security in a 10 vertex complete graph. But to recast our problem in their context, we would need to introduce one vertex for every bid, and with ≈ 2000 bids in the instances we aim at solving, making that approach grossly impractical.

3.1 Greedy Algorithm

We describe now a more efficient greedy algorithm than just solving the problem as a general minimum cost flow problem. This greedy algorithm makes use of the special structure of the MCF we want to solve and can be easily generalized into an MPC environment. Intuitively the algorithm proceeds as follows. It considers each order in turn, starting with the cheapest (i.e. best from the objective function point of view) one. At each iteration, a max-flow problem is solved to try to use as much as possible of the quantity offered by the bid. The incremental value obtained is the quantity adjudicated to that bid. The following is a formal description of this greedy procedure:

```
1. ν ← 0
2. B ← sort-price:B
3. x_i ← 0   ∀i ∈ S
4. for all:   i ∈ B :
5.       x_i ← |q_i|
6.       ν' ← maxflow: G(V,A)
7.       x_i ← ν' − ν
8.       ν = ν'
9. End
```

Algorithm: 1: Iterative Greedy Algorithm for Multi-Market Auctions

First, we sort the set of all bids B in function of their price and set the capacities of edges in S to 0. Second, we restore the capacity of the edge associated to bid i to its original value $|q_i|$ and calculate then max-flow on G. We then set the capacity of such edge to the flow variation with respect to the max-flow calculated in the previous iteration. We repeat this process for all bids in the order of prices. Once this process is completed, the volume provided by demand bids is then automatically rejected and accepted for the supply bids.

3.2 Correctness

We now prove that the greedy algorithm described above is correct. To do this let us disaggregate each bid as a collection of bids of capacity 1, each with the same price as the orginal bid. This obviously does not modify the problem. So from now on in this subsection, we can safely assume that all bids have quantity 1, and that all bids will be completely accepted or rejected. For a given set of bids I, let $r(I)$ be the maximum amount of demand that can be satisfied using the bids of I only. This can be seen as a max-flow problem on the graph G.

Proposition 1. *The set function $r : 2^S \to \mathbb{R}^+$ is the rank function of a matroid.*

Proof We use a characterization of Whitney [16] for a function to be the rank function of a matroid:

(a) $r(\emptyset) = 0$.

(b) $r(I) \leq r(I + i) \leq r(I) + 1$ for $I \in S$ and $i \in S \setminus I$,

(c) for all $I \subseteq S, i, j \in S \backslash I$, if $r(I+i) = r(I+j) = r(I)$, then $r(I+i+j) = r(I)$.

The set of arcs associated to the bids themselves is a cut separating the source from the sink in the associated max-flow problem so $r(J) \leq |J|$ for any J, proving (a). Moreover (b) comes from the fact that adding one bid i to I amounts to increase the capacity of one arc by one unit in the associated max-flow problem. Therefore the capacity of any cut increases by at most 1, and the size of the minimum cut will certainly increase, but by one unit at most.

We now prove (c). Let S^I denote the set of vertices containing the source s defining a minimum cut associated with the max-flow problem of computing $r(I)$. In other words, $r(I) = c(\delta^+(I))$.

Note first that since $r(I+i) = r(I)$, there exists S^{I+i} such that the associated cut does not contain (s, i) the arc associated to the bid i. Similarly there exists S^{I+j} such that the associated cut does not contain (s, j) the arc associated to the bid j. This implies also that $\delta^+(S^{I+i} \cup S^{I+j})$ does not contain the arcs (s, i) and (s, j).

By submodularity of cut functions in directed graphs, we obtain that $r(I+i)+ r(I+j) = c(\delta^+(S^{I+i}))+c(\delta^+(S^{I+j})) \geq c(\delta^+(S^{I+i} \cup S^{I+j}))+c(\delta^+(S^{I+i} \cap S^{I+j}))$.

Since $\delta^+(S^{I+i} \cup S^{I+j})$ is an $s-t$ cut that does not contain (s, i) and (s, j), if $c(\delta^+(S^{I+i} \cup S^{I+j})) \leq r(I)$, statement (c) holds (the strict inequality case is ruled out by (b)). If $c(\delta^+(S^{I+i} \cup S^{I+j})) > r(I)$ then $c(\delta^+(S^{I+i} \cap S^{I+j})) < r(I)$. But this would contradict the minimality of S^I since $\delta^+(S^{I+i} \cap S^{I+j})$ is an $s-t$ cut. □

By classical properties of matroid structures, the last proposition directly implies that we can solve the auction problem greedily: it suffices to use the cheapest supply bids first, as long as the transmission network allows the use the bid to satisfy some demand. This is exactly what Algorithm 3.1 does.

4 Cryptographic Preliminaries

4.1 Security Model

Ben-Or et al. [17] showed, amongst other things, how (with Shamir's secrete sharing for passive adversaries or Verifiable Secret Sharing (VSS) [18] for active adversaries) every functionality can be computed under the *information theoretic* model. However, that does not necessarily imply efficiency in terms of performance. In our secure algorithms, variations can be included, to accelerate some functionality, at the price of providing statistical security and/or some leakage. Moreover, changes in the communicational or adversarial models would yield different security levels as well. Our protocols follow the same line of thought. Our privacy-preserving protocols can achieve the same level of security, than the underlying primitives (our algorithms have no leakage). A careful sub-routine selection can yield statistical security, with a significantly improvement in terms of performance. We study both aspects of the implementation of our secure protocols.

4.2 Basic Building Blocks

On Secret Sharing and other Primitives. Our algorithm is compatible with secret sharing methods and homomorphic encryption mechanisms that support MPC e.g. (Shamir Secret Sharing, Paillier Encryption). Our secure prototype uses secret-sharing to allow n parties to share information amongst each other, to later be reconstructed by a subset of the players. This is also true for more elaborated primitives like multiplications, that in the case of [17,19] can be executed with a single communicational round guaranteeing perfect security. For an extended review on sharing mechanism we refer the reader to [20].
Comparisons. Which are an essential part of our algorithms. There have been several methods for secure comparisons proposed during the last decade that provide perfect security e.g. ([21,22]). Here we use the constant rounds method of Catrina and Hoogh [23]. It is built upon a secure modulo operation. As for the equality test, we use the protocol of Limpaa and Toft [24] based on the hamming distance, that provides sub-linear complexity for the on-line phase. Although, these methods achieve constant complexity bounds, in practice, due to the high constants, they are typically much slower than multiplications.

4.3 Complex building Blocks

Our privacy-preserving protocol requires to solve a series of more complex problems, combinatorial in nature. The methods used to solve these problems have to guarantee correctness and security while at the same time minimize their impact over the performance. This includes a practically efficient vector shuffling protocol, sorting and max-flow mechanisms. We succinctly review them in the context of the needs of the application at hand.

Vector Permutation Mechanism. Our protocols require to securely permute a vector. This implies that for any vector of size n, the resulting configuration is uniformly distributed in the space of all permutations $n!$. Indeed, the state of the art describes several mechanisms for vector permutation that are compatible with our algorithms. We could mention, for instance, the work of Leur et al. [25] or Keller and Scholl [26] who introduced several permutation mechanisms that work with secret sharing e.g. (permutation matrix multiplication with $\mathcal{O}(n^2)$ and perfect security). They also offer other alternatives, e.g. (sorting methods) to improve complexity with $\mathcal{O}(n \times log(n))$ and further. Additionally, Czumaj et al. [27] have shown how to build a permutation network using exchange gates with $\frac{1}{2}$ probability. The result is a permutation with (almost) uniform probability in the space of all possible permutations. Note that we could also build such networks using AKS or the randomized shell sort network introduced by [28] among others sorting networks to achieve better complexity times e.g. $\mathcal{O}(n \times log(n))$. This is also compatible with our protocols. A more realistic approach, uses sorting networks and being subject to the distribution it provides for its solutions e.g. Batcher's odd-even merge. Note that, in the same spirit, we could uniformly choose a random permutation amongst a sub-set of all possible permutations using the network generated by the Merge step of such algorithms. Indeed, these last 2 are later used for experimentation.
Sorting Mechanisms. Our scheme needs to sort the bids in ascending order or price. Since the Sorting protocols are necessary building blocks of various complex solutions. Efficient secure sorting algorithms have been studied for several years,

yielding interesting results e.g. ([28,29,30]). More interestingly for us, is the approach proposed by Hamada et al. [31]. Their idea is to first randomly and securely shuffle the vector to be sorted. Once this is done, any traditional sorting algorithm can be executed, revealing the results of the comparison, while keeping secret the values to be sorted.

Max Flow Mechanisms. Max-Flow flow problems with perfect security have been recently studied by [32,33] amongst others. For the max-flow problem, Blanton et al. [33] introduced a mechanism to solve the problem using as building block the Bread First-Search algorithm with a complexity of $\mathcal{O}(n^5 log(n))$. The work by Aly et al. [32] provides 2 different data-oblivious protocols with perfect security as well. The most efficient method is $\mathcal{O}(n^4)$ and is based on the push-relabel algorithm. It also suggests the use of stopping conditions (with some leakage) to accelerate performance. This is what we have implemented here.

5 Secure Auction Mechanism

We extend the results of section 3 and introduce a secure variant of algorithm 3.1. We assume the configuration of the transmission network to be public, and all inputs to be integer.

5.1 Notation

Our protocol uses the traditional square brackets notation employed by several secure applications in distributed environments e.g. [21,34,32]. For instance, a secure assignment and secure addition are denoted by the use of the infix notation and the corresponding square brackets e.g. $[z] \leftarrow [x] + [y]$. The same treatment is extended to any other operation. Vectors are denoted by capital letters e.g. E where $|E|$ denotes the number of elements in E and E_i is the i-th element. To represent negative numbers we use the typical approach of using the lower half of the field for positive values and the upper half of the field for negative values. It has to be noticed that in shared form a negative value is indistinguishable from a positive one. And that in our approach all information related to the bids is kept secret including whether or not it is a supply or demand bid.
Each bid is represented by a tuple $([b_i], [m_i], [p_i], [q_i])$ where b_i is the bid identifier, m_i is the market where the bid is made, p_i its limit price and q_i its quantity. The transmission network is represented by the capacity matrix N. We will make repeated use of the following two subroutines.

- conditional assignment : This functionality serves as a replacement of a flow control instruction for branching. Although branching on encrypted data is not possible, the functionality can be emulated for assignments. Following [6] we represent the operator by : $[z] \leftarrow_{[c]} [x] : [y]$. Where much like in previous works e.g. [6,32,35,34] $[z]$ would take the value $[x]$ if $[c]$ is 1 and $[y]$ otherwise. This can be achieved simply by doing the following $[z] \leftarrow ([x] - [y]) \times [c] + [y]$.

- market identification : Part of the data that composes a bid is the identification of the market it belongs to. Users are required to input a single identification tag. During our algorithm, we transform this to a unary expansion defined as

$Z_{i,m} = 1$ if $m = m_i$ and 0 otherwise. This enables us to reduce the number of equality tests when performing the market identification for a bid. This transformation can be achieved following protocol:

Protocol 1: unary expansion for market identification

Input: vector of all markets M, bid $[i] \in B$.
Output: zero-one matrix $[Z]$ of size $n \times m$
for $i \leftarrow 1$ **to** m **do**
  $[Z]_{i,j} \leftarrow j == [m]_i$;
end
return $[Z]$;

5.2 Secure Auction with Transmission Constraints

The protocol is defined as follows:

Prerequisites. The number of bids or at least an upper bound on the size of the vector is assumed to be public. We assume the topology and capacities of the transmission network to be public.
1. Bids are sorted in ascending order of price.
2. The structure of the graph $G = (V, A)$ is public. The capacity of each edge is initialized with the following value. The capacities between market vertices are set to the capacity matrix C. The capacity of each edge (s, j) is set to 0. The capacity of each edge (j, t) is set to the sum of the quantities of all demand bids submitted to market j. This is simply done by exploring all the bids and using the market identification protocol 1.
3. Evaluate the viability of each of the bids from the recently sorted vector $[B]$ in ascending order. For a given bid, we do this by increasing the capacity of edge (s, j) where j is the market of the bid by its quantity $|q_{b_i}|$. We then compute the maximum (s, t)-flow in the graph G to determine whether the bid can improve the solution. The increment of the maxflow compared to the previous iteration is the amount adjudicated to the bid. Finally, the capacity of edge (s, j) is increased by the same increment. Protocol 2 shows a detailed description of this procedure.
4. Finally the bids then are permuted randomly, to hide their order. This is necessary to avoid leaking the result of the initial sorting from step 1.

On the prerequisites, several parties constantly submit bids in shared form, we believe it is safe to assume this will not occur simultaneously. Precomputed permutation matrices can be generated. A simple vector multiplication of the corresponding row of the matrix would suffice in this case to place the incoming data in their corresponding permuted position in the vector. Once all data is received, the existing vectors can be easily combined, the result is a single permuted vector. In case this approach is not feasible, the algorithm designer could make use of one of the suggested permutation mechanisms instead. Permuted bids would allow us to make use of Hamada et al [36] technique of `shuffling before sorting`. This improves considerably the performance of sorting protocols and allows them to achieve $\mathcal{O}(n \times \log n)$ complexity.
Furthermore, we introduce step *3.* to serve as an evaluation and allocation mechanism. It can be seen as some heuristic tool that allows us to identify the impact

of the bid on the result. Protocol 2 let us explore the inner works of Step *3.* in detail. *Line 2* allows us to explore all previously sorted bids in order. *Lines 3 to 5* augment the corresponding edge capacity from the source to the corresponding market with the volume of the bid. *On Line 6*, ν stores the maximum amount of flow that can be allocated with the new volume. On the final section of the protocol (*Lines 7 to 13*) the difference between previous and present flow gap is calculated. Moreover, the flow added to the graph at the beginning of the iteration is replaced by the gap variation. This value has to be stored as well as the amount of capacity assigned to the bid and the value of the maximum flow for future iterations.
Moreover, at the last and *4.* step, data can be edited at will by the algorithm designer. What information is taken to later be presented depends solely on the application's nature. The permutation, although capable to hide the sorting should be ignored in case the final answer also contemplates to open the prices of the bids as well. This is because any party could later sort the bids accordingly. Please note that our protocol complexity grows linearly with respect of the number of bids n and polynomially by the number of markets m.

Protocol 2: Implementation of secure auction.

Input: Capacity matrix $[C]_{ij}$. Vector of n bid tuples $([b], [m], [p], [q])$. Matrix of market identification $[Z]_{ij}$ $\forall i \in N$ and $\forall j \in M$
Output: Flow Matrix F, the list of bids and their accepted quantities $[x]$

1 $[\nu] \leftarrow [0]$
2 **for** $i \leftarrow 1$ **to** n **do**
3 **for** $j \leftarrow 1$ **to** m **do**
4 $[C]_{sj} \leftarrow_{[Z]_{ij}} [C]_{sj} + |[q_{b_i}]| : [C]_{sj}$;
5 **end**
6 $[\nu'] \leftarrow$ `maxflow`$([G])$;
7 $[\phi] \leftarrow ([\nu'] - [\nu])$;
8 $[x_{b_i}] \leftarrow [\phi]$;
9 **for** $j \leftarrow 1$ **to** m **do**
10 $[C_{sj} \leftarrow_{[Z]_{ij}} [C]_{sj} - |[q_{b_i}]| + [\phi] : [C]_{sj}$;
11 **end**
12 $[\nu] \leftarrow [\nu']$;
13 **end**

Finally, note that at the end of the protocol, the adjudicated quantities x_i together with the bid identifier will be disclosed.

Complexity. Oblivious shuffling can be achieved (theoretically) in $\mathcal{O}(n \log n)$, and sorting the bids is then $\mathcal{O}(n \log n)$. At step 3., a max flow problem on $m+2$ vertices has to be executed n times. Using the $\mathcal{O}(m^4)$ max-flow algorithm by Aly et al. [32], this gives an overall bound of $\mathcal{O}(nm^4)$ for Step 3. So in total this yields an overall complexity (communication rounds) of $\mathcal{O}(n(m^4 + \log n))$.
Since the number of markets is usually small (e.g. 1 to 5) for the application at hand, the quartic exponent is not too much of an issue in practice. The fact that the complexity is close to linear in the number of bids is on the other hand vital.

5.3 Security and Correctness

In principle, all steps of the algorithm could be implemented with perfect (i.e. information theoretic) security against passive and active adversaries when implemented with no leakage. This follows from the fact that there exists such protocols for sorting [29,31,28], max-flow [32,33] and oblivious shuffling [25,26,27] that provide this level of security under the information theoretic model. This is also true for multiplications and comparisons. Moreover, the data-obliviousness nature of the protocols implies that for the protocol simulation, the corresponding simulators of all other protocols and their atomic operations e.g. (secure additions multiplications) could be invoked in a predefined execution order, hence, modeling it as an arithmetic circuit.
However, to improve performance, we have decided to weaken the security requirement in the following sense. Firstly, the security of the sorting scheme we use directly depends on the security of the random shuffling implemented. Secondly, for the max-flow problems, instead of running the algorithm for the maximum theoretical number of iterations, we stop the algorithm as soon as optimality is reached. This leaks the number of iterations. Finally, correctness follows from the fact the the scheme is a secure implementation of algorithm 3.1.

6 Computational Experimentation

We have tested our protocol with our custom-made MPC Toolkit library implemented in C++. It implements all the primitives and building blocks described above, but also the underlying MPC crypto-primitives. It also provide our own communicational support and use NTL and GMP libraries for the underlying modulo arithmetic. We report below on the most relevant aspects of the implementation. A more thorough and detailed description can be found in [37].

6.1 Prototype Capabilities and Technical Characteristics

Table 1 give the implemented protocols for the usual basic operations. The more complex procedures described in the previous sections are built upon these.

Building Block	**Algorithm**
Sharing	Shamir Secret Sharing [12]
Multiplication	Gennaro et al [19]
Equality Test (Statistical)	Limpaa and Toft [24]
Inequality Test (Statistical)	Catrina and Hoogh [35]
Random Bit Gen.	Damgård et al [21]

Table 1. List of Primitives used by the Secure Auction protocol implementation

These are considered core functionalities. The architecture from the library is to provide a basic and decoupled processing unit similar to a small engine. This small engine implements the functionalities from table 1. Furthermore, it separates computational and cryptographic tasks from communicational tasks. Each engine runs these two sets of tasks in different threads that communicate with each other to coordinate. Basic requirements to obtain the best performance from the engine include 2 CPU threads and $\approx$ 500 KB in RAM for the

basic use of the primitives. Our configuration gives each computational party 2 similar CPU threads with unlimited access to a memory pool of up 42 GB with each player having a single engine's instance.

On Security. The library and prototype were built under the private channel model. Depending on the functionality used, the library provides statistical and perfect security against semi-honest adversaries with minority coalition. For instance, the inequality tests used in our tests brings statistical security meanwhile addition and multiplication perfect security. As mentioned before, statistical security for such method is given as a function of parameters k and the bit-size of the input by parameter l. The prototype was pre-configured to use $k = 29$ and $l = 32$. However, because of technical issues, shares themselves can only use up to 63 bits. This means in practice that under the scenario where only primitives with perfect security are used, the size of l could grow up to 63 bits.

6.2 Numerical Results

The computational experiments were done using historical data from the belgian day-ahead market. The data set is composed of a total of 1945 bids (demand and supply). The origin of the data is one hour of a day trade from the Belpex market (12 pm). We have created two data sets by randomly partitioning this set of bids into 2 and 4 markets. Additionally, we would like to note that in our experimentation we consider a 3 computational parties case.

We ran our instances on an Intel Xeon CPUs X5550 (2.67GHz) and 42GB of memory, running Mac OS X 10.10. All processes have the same computational power at their disposal (memory and CPU power). Table 2 shows number of communication rounds, comparisons and CPU time (10 executions average).

From these tests, $\approx 21 \times 10^6$ communicational rounds were dedicated to randomization processes for the comparison mechanisms e.g. random bit generations. The use of well studied results like PRSS [38] would limit the use of these communicational rounds and in general terms would allow us to achieve even better computational times. Furthermore, Catrina and Hoogh comparison method depends on the computation of l random bits for its calculations. An offline phase can be considered where this random numbers are pre-computed before the bids arrive to the server, and then distributed to the computational parties for its use. In this case the Secure Auction Mechanism would be executed in an online phase that no longer has to care about random number generation improving the performance for comparisons. Table 2 shows our numerical results and estimated the impact of the use of online/offline phases.

Markets and Perm. Method		Com. Rounds	Comparisons	CPU Time.	Online Phase
2 Markets	Batcher	$\approx 31.4 \cdot 10^6$	226021	2056 s.	613 s.
	Merge	$\approx 31.4 \cdot 10^6$	226021	2049 s.	606 s.
4 Markets	Batcher	$\approx 71.9 \cdot 10^6$	537627	4702 s.	1276 s.
	Merge	$\approx 71.9 \cdot 10^6$	537627	4694 s.	1268 s.

Table 2. Overall Results

When the number of markets increases from 2 to 4, we observe an increase of more than twice the number of rounds and comparisons. The same follows

on computational time, taking in average (10 execution rounds) around 4700 seconds to complete execution.
On memory, the application did not surpass the 2.5 MB per execution. During its life-cycle, some increment in memory consumption levels was registered during data generation phases. The phenomenon is especially evident in the data pre-processing phase. This is of course, because some data is generated and stored for later use. Less significantly changes are also present. This is explained by the continuous fragmentation of the memory throughout the repeated max-flow problems, where objects of different sizes are continuously created and then destroyed. Nonetheless, the increase of memory usage remains very modest. Figure 2 shows the typical memory usage of the application during its execution.

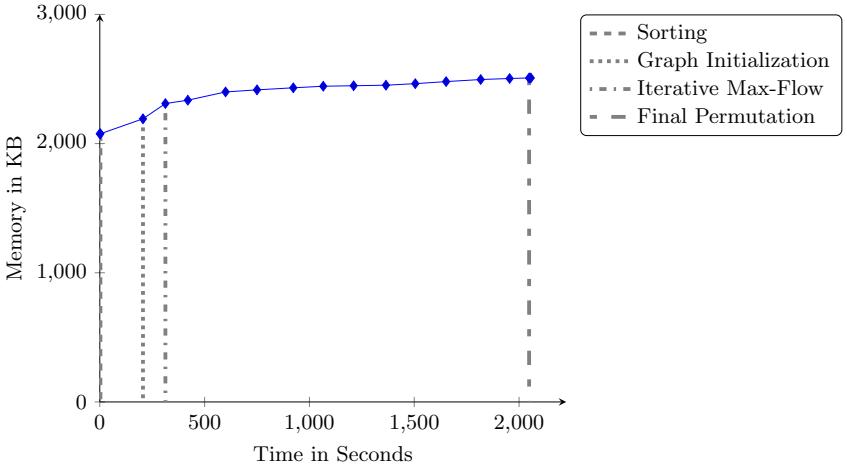

Fig. 2. Secure Auction Protocol Life Cycle

Finally, we found that the bottleneck of the application are the communications. Data transmission and related tasks are responsible of the 1705 seconds (83%) of the total computational time. Only a fraction of the time, 351 seconds (17%), was dedicated to other tasks e.g. share generation, basic arithmetic operations and other algorithmic tasks.

From these results we can conclude the following:

(i) Realistic computational times were indeed achieved for the data in question, with limited computational power. With the online/offline case, an hour of trade was solved in less than an hour of computations for both market configurations. Given that many of the processes of our protocol are sequential in nature, a computer with a better benchmarked CPU under the same basic configuration would yield better results. **(ii)** Memory is not a decisive factor in this case. Memory increases monotonically but modestly during the execution because of noise. **(iii)** The process can be further accelerated by pre-computing the random values that are needed by the protocol. In our case $\frac{3}{4}$ of communicational rounds that are used for comparisons are dedicated to randomization processes. Even with the use of PRSS, these operations would represent an important proportion of the workload. This is why an offline phase where these values are preprocessed could prove more

useful. For instance, to have dedicated servers calculating in shared form random bits and numbers and store them, such that they can just be fetched when any online process needs them. This would imply a reduction in the 2 markets case of ≈ 1450 seconds. This would allow us to solve the problem in ≈ 610 seconds, a little more than 10 minutes. When 4 markets are consider instead, the times are reduced to ≈ 1270 which is little more than 20 minutes. **(iv)** Moreover, even though we have put in place a light and dedicated communications setting, the performance of the prototype is largely dependent on the performance of the communications implemented. It was 4.8 times more expensive (in terms of running time) to transmit the data than to generate, reorganize and calculate it.

7 Conclusions

Our computational experiments show that secure auctions in realistic settings (≈ 2000 bids, 4 markets linked by capacitated transportation links, 30 minutes time limit) are indeed possible. This required the development of a specific algorithm to solve the problem (more amendable to MPC), a careful management of the trade-off between security and performance (perfect vs. statistical security, leakage of the number of iteration when solving the max-flow problems), and a dedicated low-level implementation of MPC primitives.

Since, in our current implementation, the bulk of the running time is in communications, we feel that this is where lies the best opportunities to improve performance. This could be achieved either in improving the communication efficiency, or by reducing the need for communication between the players. Moreover future practical implementations could make use of dishonest majority or active security protocols for usability.

Acknowledgements

This research was supported by the WIST Walloon Region project CAMUS and the Belgian IAP Program P7/36 initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors. The authors are grateful to Olivier Pereira, Ignacio Aravena and the anonymous reviewers for their feedback.

References

1. Klemperer, P.: What really matters in auction design, from auctions: Theory and practice. Princeton University Press (2004)
2. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, IEEE (1982) 160–164
3. Shoup, V.: Ntl: A library for doing number theory (2001)
4. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T. In Dingledine, R., Golle, P., eds.: Financial Cryptography and Data Security, Berlin, Heidelberg, Springer-Verlag (2009) 325–343

5. Geisler, M.: Cryptographic protocols: theory and implementation. PhD thesis, Aarhus University Denmark, Department of Computer Science (2010)
6. Aly, A., Van Vyve, M.: Securely solving classical network flow problems. In Lee, J., Kim, J., eds.: Information Security and Cryptology - ICISC 2014. Volume 8949 of Lecture Notes in Computer Science., Springer International Publishing (2015) 205–221
7. Franklin, M.K., Reiter, M.K.: The design and implementation of a secure auction service. In: Proceedings of the 1995 IEEE Symposium on Security and Privacy. SP '95, Washington, DC, USA, IEEE Computer Society (1995) 2–
8. Harkavy, M., Harkavy, M., Tygar, J.D., Tygar, J.D., Kikuchi, H., Kikuchi, H.: Electronic auctions with private bids (1998)
9. Kikuchi, H., Hotta, S., Abe, K., Nakanishi, S.: Distributed auction servers resolving winner and winning bid without revealing privacy of bids. In: Parallel and Distributed Systems: Workshops, Seventh International Conference on, 2000. (Oct 2000) 307–312
10. Peng, K., Boyd, C., Dawson, E.: Optimization of electronic first-bid sealed-bid auction based on homomorphic secret sharing. In Dawson, E., Vaudenay, S., eds.: Progress in Cryptology – Mycrypt 2005. Volume 3715 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 84–98
11. Nojoumian, M., Stinson, D.: Efficient sealed-bid auction protocols using verifiable secret sharing. In Huang, X., Zhou, J., eds.: Information Security Practice and Experience. Volume 8434 of Lecture Notes in Computer Science. Springer International Publishing (2014) 302–317
12. Shamir, A.: How to share a secret. Commun. ACM **22**(11) (1979) 612–613
13. Catane, B., Herzberg, A.: Secure second price auctions with a rational auctioneer. In: The 10-th SECRYPT International Conference on Security and Cryptography. (2013)
14. Madani, M., van Vyve, M.: A new formulation of the european day-ahead electricity market problem and its algorithmic consequences. CORE Discussion Papers 2013074, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE) (2013)
15. Madani, M., Van Vyve, M.: Computationally efficient {MIP} formulation and algorithms for european day-ahead electricity market auctions. European Journal of Operational Research (0) (2014) –
16. Whitney, H.: On the abstract properties of linear dependence. American Journal of Mathematics **57** (1935) 509–533
17. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, ACM (1988) 1–10
18. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: Foundations of Computer Science, 1985., 26th Annual Symposium on. (Oct 1985) 383–395
19. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In: Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing. PODC '98, New York, NY, USA, ACM (1998) 101–111
20. Beimel, A.: Secret-sharing schemes: A survey. In Chee, Y., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C., eds.: Coding and Cryptology. Volume 6639 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 11–46
21. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: TCC. (2006) 285–304

22. Nishide, T., Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In: Public Key Cryptography. (2007) 343–360
23. Catrina, O., de Hoogh, S.: Improved primitives for secure multiparty integer computation. In: SCN. (2010) 182–199
24. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: ICALP (2). (2013) 645–656
25. Laur, S., Willemson, J., Zhang, B.: Round-efficient oblivious database manipulation. In Lai, X., Zhou, J., Li, H., eds.: Information Security. Volume 7001 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 262–277
26. Keller, M., Scholl, P.: Efficient, oblivious data structures for mpc. IACR Cryptology ePrint Archive **2014** (2014) 137
27. Czumaj, A., Kanarek, P., Kutylowski, M., Lorys, K.: Delayed path coupling and generating random permutations via distributed stochastic processes. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '99, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1999) 271–280
28. Goodrich, M.T.: Randomized shellsort: A simple data-oblivious sorting algorithm. J. ACM **58**(6) (December 2011) 27:1–27:26
29. Jónsson, K.V., Kreitz, G., Uddin, M.: Secure multi-party sorting and applications. IACR Cryptology ePrint Archive **2011** (2011) 122
30. Goodrich, M.T.: Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in o(n log n) time. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing. STOC '14, New York, NY, USA, ACM (2014) 684–693
31. Hamada, K., Ikarashi, D., Chida, K., Takahashi, K.: Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. IACR Cryptology ePrint Archive **2014** (2014) 121
32. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Van Vyve, M.: Securely solving simple combinatorial graph problems. In: Financial Cryptography. (2013) 239–257
33. Blanton, M., Steele, A., Alisagari, M.: Data-oblivious graph algorithms for secure computation and outsourcing. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. ASIA CCS '13, New York, NY, USA, ACM (2013) 207–218
34. Toft, T.: Solving linear programs using multiparty computation. In: Financial Cryptography. Volume 5628 of LNCS., Springer (2009) 90–107
35. Catrina, O., de Hoogh, S.: Secure multiparty linear programming using fixed-point arithmetic. In: ESORICS. (2010) 134–150
36. Hamada, K., Kikuchi, R., Ikarashi, D., Chida, K., Takahashi, K.: Practically efficient multi-party sorting protocols from comparison sort algorithms. In: ICISC. (2012) 202–216
37. Aly, A.: Network Flow Problems with Secure Multiparty Computation. PhD thesis, Universté catholique de Louvain, IMMAQ (2015)
38. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In Kilian, J., ed.: Theory of Cryptography. Volume 3378 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 342–362