An MPC-based Privacy-Preserving Protocol for a Local Electricity Trading Market

Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A. Mustafa

KU Leuven, ESAT-COSIC and iMinds,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{firstname.lastname}@esat.kuleuven.be

Abstract. This paper proposes a decentralised and privacy-preserving local electricity trading market. The proposed market employs a bidding protocol based upon secure multiparty computations and allows users to trade their excess electricity among themselves. The bid selection and calculation of the clearance price at which the electricity is traded are performed by the market in a decentralised and privacy-preserving manner. We implemented the market in C++ and tested its performance with realistic data sets. Our simulation results show that the market tasks can be performed for 2500 bids in less than five minutes in the online phase, showing its feasibility for a typical electricity trading period of, for example, 30 minutes.

Keywords: Secure Multiparty Computation, Local Electricity Trading Market, Smart Grid, Renewable Energy Source, Security and Privacy.

1 Introduction

The Smart Grid (SG) is an electricity grid supporting bidirectional communication between the main components in the grid. For instance, an important component is smart meters which allow real-time grid management [1]. Potential benefits of SG include improved grid management, efficiency and reliability, and seamless integration of various green energy sources such as, Renewable Energy Sources (RESs) (e.g., solar panels, wind turbines), into the distribution grid. When these RESs generate more electricity than needed by their owners, the excess electricity is fed back to the grid. Currently, households get some compensation from their suppliers for such excess electricity at a regulated, low price. However, households with such excess electricity may be interested in selling directly to other consumers at a competitive price for monetary gains. Enabling that would also incentivise more and more households to own RESs. To address this, a local electricity market that allows RES owners to trade their excess electricity with other households in their neighbourhood has been proposed in [2]. However, such a local electricity market has user privacy risks, since users' bids and offers reveal private information about their lifestyle [3].

There are various proposals for an electricity trading market that allows users to trade with each other or suppliers, using game-theoretic approaches (cf. Section 1.1). None of these, however, addresses the privacy concerns. The security and privacy concerns in such a local market have been analysed in [2], and initial ideas for designing one has been proposed in [4]. However, no concrete solution has been proposed. In this work, we not only propose a concrete secure and privacy-preserving solution for such a local market for trading electricity, but also implement and evaluate its performance using realistic data.

1.1 Related Work

Privacy in Local Electricity Markets. Preserving users' privacy in SG has already been recognised as an important issue by the research community [5, 6]. However, the majority of the privacy-preserving smart metering solutions focus on efficient grid management and billing. While some use anonymisation of the metering data [7, 8], others utilise homomorphic cryptographic schemes to aggregate the consumption data [9–11]. There are various local electricity trading market models that have been proposed in the literature [12, 13]. Mustafa et al. are the first to perform a comprehensive security analysis of such a market and raise the privacy concerns associated with it in [2]. However, a concrete solution has not been proposed. Recently, Abidin et al. [4] proposed to use secure multiparty computation to address these privacy concerns, however, they have also not provided a concrete solution, not to mention an evaluation of their ideas.

Electricity Markets using MPC. Ever since Yao's seminal work [14], Multiparty Computation (MPC) has grown from being a theoretical result to being a mechanism used in practical applications [15]. Various frameworks such as, VIFF, Tasty, Sharemind [16–18] have been developed and significant improvements in terms of efficiency and security are made to MPC protocols [19, 20]. Contributions to problems such as, secure comparisons, secure sorting, and network flows [21, 22] opened the door for the design of auction mechanisms based on day-ahead electricity markets [23], allowing electricity suppliers and generators to interact among themselves in a secure and privacy-preserving manner.

Contributions. Our specific contributions include:

– A novel application of MPC to identify the selected bids, calculate the clearance price, and compute the total amount of electricity traded by the users belonging to each individual supplier in a data oblivious and secure manner.
– A concrete decentralised and privacy-preserving protocol for a local electricity trading market using MPC.
– A security and complexity analysis of our protocol in the context of MPC.
– An implementation, evaluation and analysis of the proposed protocol using a realistic data for various market sizes.
– A list of trade-offs between privacy and efficiency for realistic scenarios.

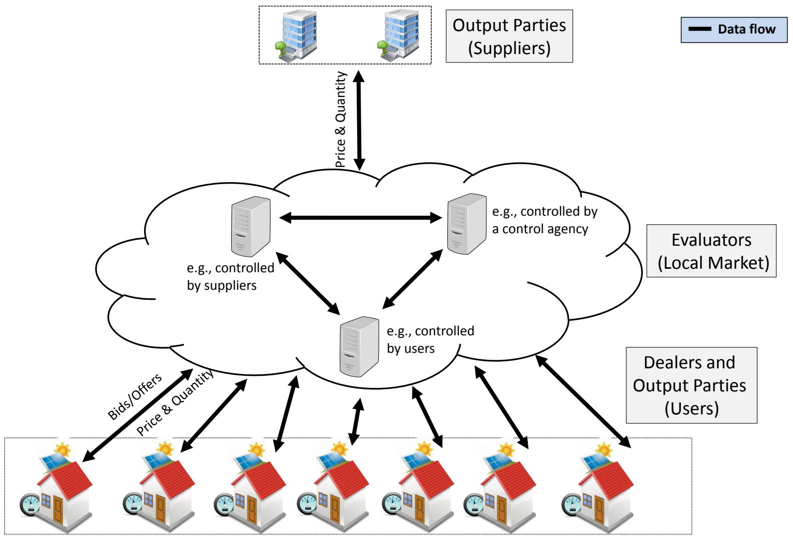
Output Parties (Suppliers)
Data flow
Price & Quantity
e.g., controlled by a control agency
e.g., controlled by suppliers
e.g., controlled by users
Evaluators (Local Market)
Dealers and Output Parties (Users)
Bids/Offers
Price & Quantity

Fig. 1. A local MPC-based market for trading electricity from RESs.

Outline. The rest of the paper is organised as follows: Section 2 elaborates on the system and threat model, assumptions, functional requirements, and also presents our security definition. In Section 3 and 4, we present our proposed protocol and analyse its security, respectively. Section 5 gives the details on our implementation and simulation results, and discusses privacy and efficiency trade-offs . Finally, Section 6 concludes the paper and gives directions for future work.

2 Preliminaries

This section briefly describes the system model, the local electricity market, threat model, the assumptions and requirements introduced in [2, 4] on which our protocol is based.

System Model. As shown in Fig. 1, a local electricity market comprises the following entities: RESs, SMs, users, suppliers and the local electricity market. If users do not trade in the local electricity market, their contracted suppliers are responsible for providing them with electricity.

Local Electricity Market Overview. Here we briefly describe the local electricity trading market proposed by Mustafa et al. [2]. The market operation consists of the following steps.

- **Bid Submission:** Prior to each trading period, users submit their bids to the market to inform the market how much electricity they are willing to sell or buy during the trading period and for what price per unit.
- **Trading Price Computation:** The local market performs a double auction trading and generates the supply and demand curve. The intersection of these two curves is used to determine the trading price, amount of electricity traded, as well as which users will trade on the market.
- **Informing Users/Suppliers:** The market informs (i) the users about the amount of electricity they traded and the trading price, and (ii) the suppliers about the amount of electricity agreed to be traded by their respective users.

Threat Model. Users and suppliers are assumed to be malicious. Users may try to modify data sent by their (or other users') SMs in an attempt to gain financial advantage, whereas suppliers may try to learn and modify users' bids in an attempt to influence the electricity trading price on the market. The local electricity market is honest-but-curious. It follows the protocol specifications, but it may attempt to learn individual users' bids from the protocol transcripts. External entities are malicious. They may eavesdrop data in transit trying to discover confidential data and/or modify the data in an attempt to disrupt the local electricity market and/or the SG.

Assumptions. Taking into account the above presented threat model, the protocol we propose is subject to the following assumptions: (i) each entity (e.g., SM, supplier) has a unique identity, (ii) SMs are tamper-evident, (iii) all entities are time synchronized, (iv) the communication channels between entities are secure and authentic, and (v) users are rational, i.e., they try to buy electricity for the cheapest price but sell their excess electricity at the highest possible price.

Functional Requirements. Our protocol should meet the requirements below.

- The local market should receive users' bids, calculate the clearing price, and inform the users and suppliers about the outcome of the market.
- Each user should learn if their bid was accepted, the market clearing price and the amount of electricity to be traded by them.
- Each supplier should learn the amount of electricity traded by their customers on the local market in each settlement (electricity trading) period.

Privacy Requirements. Our proposed protocol should satisfy the following privacy requirements:

- Confidentiality of users' bids and the amount of electricity traded;
- Users' privacy preservation;
  - RES identity privacy,
  - RES user identity privacy,

  - Trading RES user identity privacy,
  - RES user location privacy,
  - Trading session unlinkability,
- Minimum data disclosure.

We refer the curious reader to [2,4] for details on these requirements.

MPC. Secure MPC allows any set of mutually distrustful parties to compute any function such that no party learns more than their original input and what can be inferred from the output. MPC can be achieved using various cryptographic primitives. Different flavors include secret sharing [24,25], garbled circuits [14] and homomorphic encryption [26]. In short, parties $P_1, ..., P_n$ want to compute $y = f(x_1, ..., x_n)$, where x_i corresponds to the secret input of party P_i, in a distributed fashion with guaranteed correctness such that P_i learns only y and what can be inferred from y.

Security definition under MPC. The security notion can be characterized as follows: a secure protocol over MPC discloses to an adversary the same information, as if the computation were carried out by a trusted (non-corruptible) third party. This security notion implies that a secure MPC protocol emulates the "ideal" (trusted third party) setting, in which the third party would only need to execute a trivial (non-secure) version of the protocols introduced by this work or any other efficient mechanism. This definition allows for a variety of adversarial and communication models, offering various security levels: perfect, statistical or computational. Seminal results prove that any functionality can be calculated with perfect security against active and passive adversaries [24,25]. Recent work focus on efficiency and realistic scenarios, e.g., dishonest majorities [19,27,20]. Furthermore, by using arithmetic circuits, any functionality can be constructed on MPC. Notice that any oblivious functionality built in this way would be as secure as the underlying MPC protocols used for its execution. Finally, note that under this scenario, functionality, also referred to as sub-protocols, like the ones used in this work, can be used for modular composition under the hybrid model introduced by Canetti [28].

Our results make use of the following well known results:

- **Secure Comparison:** Methods for secure comparison using MPC have been proposed in the literature [29,21]. These constructions offer either perfect or statistical security and are constructed under the same assumptions as this work. Moreover, mechanisms as the ones proposed by Catrina et al. [30] introduce inequality tests at a constant complexity.
- **Secure Sorting:** Secure Sorting using MPC can be achieved by sorting networks and other data-oblivious mechanisms with perfect security, e.g., [31], including the randomize shell-sort from Goodrich [32]. Moreover, Hamada et al. [33] introduced a technique to facilitate the use of comparison sorting algorithms, e.g., Quick-sorting or Batcher's merge sort. This technique consists of randomly permuting the vector before sorting, so that the results of

some of the intermediate secure comparisons can be made public. Our secure local market makes use of the results by Hamada et al. for vector sorting.
– **Secure Permutation:** Mechanisms for secure vector permutation using MPC have been introduced [34–36]. Leur et al. [34] analysed various permutation mechanisms, including vector multiplication by a permutation matrix and the use of sorting networks. Additionally, Czumaj et al. [35] proposed alternatives for obliviously permuting a vector in (almost) $\mathcal{O}(n \times log(n))$, where n is the vector size. A recent application by Aly et al. [23] proposes the use of these mechanisms in a similar setting and suggests adaptations.

3 Privacy-preserving Protocol for Electricity Trading

In this section, we propose a protocol for trading electricity on local markets in a privacy-preserving manner. Our protocol employs MPC to guarantee secrecy. Moreover, we devise a series of mechanisms that produce the outputs expected and needed at various process stages, and guarantee secrecy.

In our scheme, bidders provide their private inputs to a virtualized third entity (i.e., the local market) consisting of multiple computational parties. Bidders submit their bids to a series of servers that function as evaluators. The selection and the number of evaluators depend solely on the application and the needs of the parties involved, and it could be as many as the number of parties involved in the computation. This approach, however, would be rather costly in terms of performance. In the current setting, we assume that one computational party could come from the RES owners (bidders), one from the suppliers and a third one from a local control agency, so the need for a trusted third party is eliminated, while still guaranteeing security and correctness. Following [4], we classify the parties involved in our scheme as Dealers, Computational Parties (Evaluators) and Output Parties.

Notations. Our proposed protocol makes use of the square brackets notation to denote either encrypted or secretly shared values [37]. Moreover, assignments that are a result of any securely implemented operation are represented by the use of the infix operator, as follows: $[z] \leftarrow [x]+[y]$. This extends to any operation over securely distributed data, since its result would be of a secret nature as well. Vectors are denoted by capital letters. For a vector, say B, B_i represents its i-th element and $|B|$ its size. The bids originated by SMs are considered as the initial input data. Each bid is a tuple $([q],[p],[d],[s],[b])$ and B is the vector of all bids. The notations for the input and output data of our protocol are shown in Table 1 and Table 2, respectively.

We assume all bid elements belong to $\mathbb{Z}_M$, where M is a sufficiently large number so that no overflow occurs. Moreover, we assume the number of bids or at least an upper bound on them is publicly known. Any other data related to the bid is kept secret. Note that in case the protocol admits a single supply and a single demand bid per SM, the computation of this upper bound is trivial. Markets could opt for enforcing all participants' SMs to submit a bid regardless of

Table 1. Input data

Symbol	Meaning
t_i	i-th time slot
$[q]_i$	Bid's electricity volume in absolute terms
$[p]_i$	Unit price enclosed in the bid
$[d]_i$	Binary value: 1 indicates a demand bid and 0 a supply bid
$[s]_i$	Unique supplier identifier, such that $s \in \{1, .., \|S\|\}$ where S is the set of all suppliers. Our algorithm however requires this value to be encoded using the Launchbury et al. index encoding approach [38]. Here, the identifier is encoded in a $\{0, 1\}$ vector, such that $[s]_{ij} \leftarrow 1$ on the j-th position that corresponds to the suppliers unique identifier, and $[s]_{ij} \leftarrow 0$ otherwise, for all $i \in B$.
$[b]_i$	Bid's unique identifier. Initially public, it serves to link the protocol output (selected/rejected bids) to the bidder

Table 2. Output data

Symbol	Meaning
$[\phi]$	Volume of electricity traded on the market during period t_i
$[\sigma]$	Market's clearance price (price of the lowest supply bid) calculated for period t_i
$[a]_i$	Binary value: 1 indicates the bid i was accepted, 0 otherwise.
$[S]^{\phi}$	Set of the volume of electricity traded by supplier affiliation where $[s]_i^{\phi}$ stands for the summation of all the accepted bids from users affiliated to the supplier i, for all $i \in S$

whether they participate or not at the t_i market clearance. Let $\top$ be a sufficiently big number such that it is greater than any input value from the users, but $\top << M$. In this scenario, non-participating SMs would have to replace their input values by [0] and $[\top]$ accordingly.

System Initialization. Private bids for trading period t_i have to be submitted before the beginning of t_{i-2}. The auction for t_i is computed at period t_{i-2} and the outcome is announced to the users and suppliers before the end of this period. This is done to allow suppliers to trade in the wholesale market during t_{i-1} so they can adjust their wholesale deals according to the local market outcome.

Some pre-computations (randomizations), might be precomputed in an "off-line phase." This can be achieved by the intervention of a trusted dealer that is not directly involved at any level of the computations [19]. The amount and the purpose of the randomly generated numbers depend on the underlying security model and primitives used by the market.

3.1 The Protocol

The description of our secure and privacy-preserving protocol is as follows.

Preprocessing for trading period t_i.

1. `Bidders:` Before the start of period t_{i-2}, each individual bid is prepared by the bidder and sent to the computational parties. In the case of secret sharing, this means share generation, using the Linear Secure Secret Sharing Scheme of choice, e.g., [39], generating as many shares as needed for the computational parties participating in the scheme. This is the only input required from the bidders. Bidders then send the corresponding shares of their bids to the respective computational parties to be evaluated.
2. `Evaluators:` To randomly permute the bidders' input, upon reception, each share (ciphertext) is multiplied with a column of a randomized permutation matrix which was computed beforehand. As mentioned in the previous section, the permutation matrix generation used by Bogdanov et al. [18] can be executed "offline". This is still performed before the start of period t_{i-2}.

Evaluation for trading period t_i.

3. `Evaluation:` The evaluation is performed at period t_{i-2}. In this phase, the clearance price, traded volume and accepted and rejected bids are calculated and identified in a data-oblivious fashion. To achieve this, we make use of Algorithm 1, which gives a detailed overview of our secure auction evaluation. It allows us to identify the clearance price $[\sigma]$, the volume of electricity traded $[\phi]$ and the vector of adjudicated demand and supply bids $[A]$. It achieves this by obliviously calculating the aggregation of the demand bids $[\delta]$, and then iterating over the set of all bids in B using their volume to match $[\delta]$. An extended analysis is provided below.

To access the vector of accepted supply bids, it is enough to compute $[A]_i \times (1 - [d]_i) \times [b]_i$. To find the vector of accepted demand bids, it is sufficient to calculate $(1 - [A]_i) \times ([d]_i) \times [b]_i$.

Inform Bidders and Suppliers (before the end of period t_{i-2}).

4. `Bidders:` To hide the bid order the vector of all bids $[B]$ with the associated $[A]$ vector should be shuffled once more. Then, the evaluators will proceed to use the `open` operation of the underlying MPC primitive, e.g., Linear Secret Sharing, on the electricity price $[\sigma]$ for period t_i, for all $i \in T$. The evaluators should follow this process by opening all the $[b]_j$, for all $j \in B$. Each evaluator E_i, for all i, will send the shares corresponding to the tuple B_{b_j} to the bidder that originated the bid identified by b_j, for all $j \in B$. The bidder can proceed to reconstruct the shares and verify the integrity of the information contained in them and whether his bid was accepted or rejected.
5. `Suppliers:` In the same fashion the evaluators send the shares of the volume aggregation S_i^{ϕ}, for all $i \in S$, to the corresponding supplier. Suppliers also learn the market clearance price. Both bidders and suppliers are informed of the results at period t_{i-2}.

Agorithm 1: Smart Market Clearance.

```
Input: Vector of $n$ bid tuples $B = ([q], [p], [d], [s], [b])$
Output: Clearance price $[\sigma]$, volume of traded electricity $[\phi]$, vector of accepted
        bids $[A]$ of size $|B|$, vector of aggregated volume traded by supplier $S^\phi$
        of size $|S|$
1  for $i \leftarrow 1$ to $n$ do
2      $[\delta] \leftarrow [\delta] + [q]_i \times [d]_i$;
3  end
4  $[\nu] \leftarrow [0]$;
5  $[S^\phi] \leftarrow \{0_1, ..., 0_{|S|}\}$;
6  $[A] \leftarrow \{0_1, ..., 0_{|B|}\}$;
7  for $i \leftarrow 1$ to $n$ do
8      $[c] \leftarrow [\nu] < [\delta]$;
9      $[\sigma] \leftarrow ((1 - [d]_i) \times [c]) \times ([p]_i - [\sigma]) + [\sigma]$;
10     $[\phi] \leftarrow ((1 - [d]_i) \times [c]) \times [q]_i + [\phi]$;
11     for $j \leftarrow 1$ to $|S|$ do
12         $[s]^\phi_j \leftarrow ([s]_{ij} \times ((1 - [d]_i) \times [c]) \times [q]_i + [s]^\phi_j$;
13     end
14     $[a]_i \leftarrow [c]$;
15     $[\nu] \leftarrow [\nu] + [c] \times [q]_i$;
16 end
```

Correctness. The general goal of the protocol is to find the clearance price and to identify the accepted and rejected bids. As previously stated any supply bid below the market clearance price, and any demand bid above this price is automatically accepted and vice versa. It is important to note that the market equilibrium can be identified when the price of a given supply allocation surpasses the price of the next cheapest available demand allocation. In other words, when supply equals demand, the market equilibrium can be identified if the price of the supply is at least the price of the demand.

In our protocol, we proceed to sort all bids regardless of whether they are demand or supply bids. Following Algorithm 1, we then proceed to identify and select bids until the aggregated demand ($[\delta] \leftarrow \sum_i^{|B|} [q]_i \times [d]_i$) is matched (note that to maintain secrecy we iterate over the set of all bids), choosing the bids in ascending order of price. If a supply bid is selected, this implies that there is no supply bid that could be allocated to reduce $[\delta]$, and hence is not part of the market clearance. Using $[d]_i$ cancels the supply bid's effect over $[\delta]$, and provides us with the sufficient tools to identify it. The opposite occurs when a demand bid is selected. At the end of Algorithm 1, the bids that were used to reduce $[\delta]$ can be identified, and correspond to all the supply and demand bids with prices below and above the clearance price, respectively. From this, the set of accepted and rejected bids follows. The clearance price is set to the price of the last selected supply bid.

Complexity. The protocol grows linearly with the number of bids. Given its variability and numbers, it would be the main factor influencing the performance. Moreover, the complexity of Algorithm 1 is $\mathcal{O}(|B| \times |S|)$. The number of suppliers rarely varies over time, and is of limited size.

As an additional note we would like to mention that secure vector permutation can be achieved in $\mathcal{O}(n \times log(n))$, where n is the size of the vector and in our case the vector is the one of the Bids $[B]$. Moreover, as previously mentioned, the sorting methods used by our secure market can achieve $\mathcal{O}(n \times log(n))$.

4 Security Analysis

The MPC mechanisms used in the protocol steps `1-5` presented above constitute a unique arithmetic circuit (addition and multiplication), with no leakage, making privacy straight forward. Moreover, the protocol can be computed with perfect security on the information theoretic model against passive and active adversaries under Canetti's hybrid model [28], by using available MPC protocols such as BGW [24]. this We refer the reader to [40], for a complete set of proofs of security and composability for BGW.

Indeed, as it was previously mentioned, seminal results in BGW [24] and CDD [25] showed that any function can be computed using MPC with the aforementioned security levels by providing secure addition and multiplication under an arithmetic circuit paradigm. Similarly, there are promising results on more restricted models, e.g., dishonest majority [19] with computational security.

Furthermore, there exists privacy-preserving sub-protocols (arithmetic circuits) for sorting, comparison and vector permutation over MPC, that provide the same security guarantees with no leakage, that can be utilized. These are integrated into a single arithmetic circuit that is our protocol i.e. modular fashion. Therefore, the security of our protocol readily follows. In other words, the order of the operations (multiplications and additions) is predetermined beforehand by the publicly available circuit. More precisely, the protocol simulation in our case can be simply achieved by invoking the corresponding simulators of the sub-protocols used, and/or atomic operations in its predefined order.

5 Experimentation and Discussion

The market introduced by this work was implemented and tested under realistic scenario configurations. We executed our experimentation using the MPC Toolkit proposed in [41] which is based on BGW [24]. This library includes all the underlying cryptoprimitives and other sub-protocols we report on, together with our own introduced code. The library was compiled with NTL (Number Theory Library) [42] that itself was compiled using GMP (GNU Multiple Precision Library). These two libraries are used for the modulo arithmetic that is extensively used by the underlying MPC protocols. Each instance of the prototype comprises two CPU threads. One manages message exchanges exclusively and the other executes the protocol and the related cryptographic tasks. The application itself is not memory demanding, with each instance requiring little more than 1 MB of allocated memory at any time, during our most memory demanding test (2500 bids).

Data Generation. We generated the data used in our experiments using a realistic data from Belgium. First we picked a time slot and date, i.e., between 13:00h and 13:30h on 5-th of May 2016, during which 2382 MW solar electricity was generated in Belgium by solar panels with total capacity 2953 MW [43], i.e., on average each solar panel has produced electricity approximately equal to 81.66% of its capacity. The average electricity consumption data of a Belgian household during the same time slot was 0.637 kW [44]. Thus, for each user we generated a random consumption data for this time slot with mean equal to the average consumption data, i.e., 0.637, standard deviation equal to 0.20 and variance equal to 0.04. Then, we randomly chose 30% of the users to have installed a solar panel at their homes, and each of the solar panels is randomly assigned one of the following electricity generation capacities: 2.3, 3.6 or 4,7 kW. After that, we randomly generated the electricity output of each solar panel during this time slot with a mean equal to the capacity of the given solar panel multiplied with the efficiency factor for the time slot, i.e., 81.66%, standard deviation equal to 0.20 and variance equal to 0.04. Once we generated the electricity consumption and generation data for each user with a solar panel, we simply subtracted the latter from the first value to find the amount of excess electricity each user has.

We assumed that there are 10 different suppliers available in the market and randomly assigned a supplier to each user. The retail electricity sell price of the suppliers is set to 0.20 €/kWh and the retail buy price is set to 0.04 €/kWh. For the bid price selection we used the following rational steps. We divided the retail electricity sell and buy price difference into nine ranges each including several (overlapping) prices, e.g., range 2 includes three prices: 0.04, 0.05 and 0.06 €/kWh, whereas range 7 includes four prices: 0.17, 0.18, 0.19 and 0.20 €/kWh. Then, for each user, depending on how much excess electricity he/she has for sell or he/she wants to buy, we picked randomly one of the prices from the appropriate price range. For selecting the appropriate price range we assumed that the users are rational, i.e., if they have a lot of excess electricity to sell, they would choose a lower asking price, so they could sell it all. In contrast, if they have a little excess electricity to sell, they would ask for a higher price since selling it for a cheap price will not allow them to make a high profit by selling it at the market compared to selling it directly to the supplier.

In summary, for each user we generated the following data items: unique user ID, amount of electricity for the bid, bid price, indicator if the bid is a supply or demand bid, and ID of the user's contracted supplier.

Characteristics. Our prototype was built in C++ following an object oriented approach, with modularity and composability in mind. It has an engine that separates communication and cryptographic tasks. Table 3 shows the detailed list of the sub-protocols used in our implementation.

Table 3. List of Primitives used by secure prototype

Primitive	**Protocol**
Sharing	Shamir Secret Sharing [39]
Multiplication	Gennaro et al. [45]
Inequality Test	Catrina and Hoogh [30]
Random Bit Generation	Damgård et al. [29]
Sorting: QuickSort	Hamada et al. [33]
Permutation: Sorting Network	Lai et al. [34]

Security. Our security target was to build a prototype for the classic scenario of semi-honest adversaries under the information theoretic model (private authenticated channels), and threshold corruption. This is achieved by the underlying BGW primitives and Shamir Secret sharing (honest majority). This is a necessary configuration to achieve perfect security as long as the adversary does not corrupt more than halve of the parties. However, the prototype offers statistical security on the size of its input, given that it utilizes the comparison method introduced by Catrina and Hoogh [30]. The security of such method depends on input parameters l and k, where l is the bit-size of the numbers and k a security parameter. Under the assumption that the channel is perfect, this task is also decoupled from the prototype operation.

Environment. We executed our tests on a single 64-bit Linux server with 2*2*10-cores with Intel Xeon E5-2687W microprocessors at 3.1GHz and 25 MB of cache available, and with memory of 256 GB.

Setting. All our tests were performed under a 3-party setting, with two available cores for each instance. We ran our tests starting with a baseline of a realistic scenario with 100 bids and then monotonically increasing the number of bids until they reached 2500 bids. Each test scenario was repeated several times to reduce the impact of the noise e.g., 5 to 10 times. The values reported in the rest of this section correspond to those of the resulting averages.

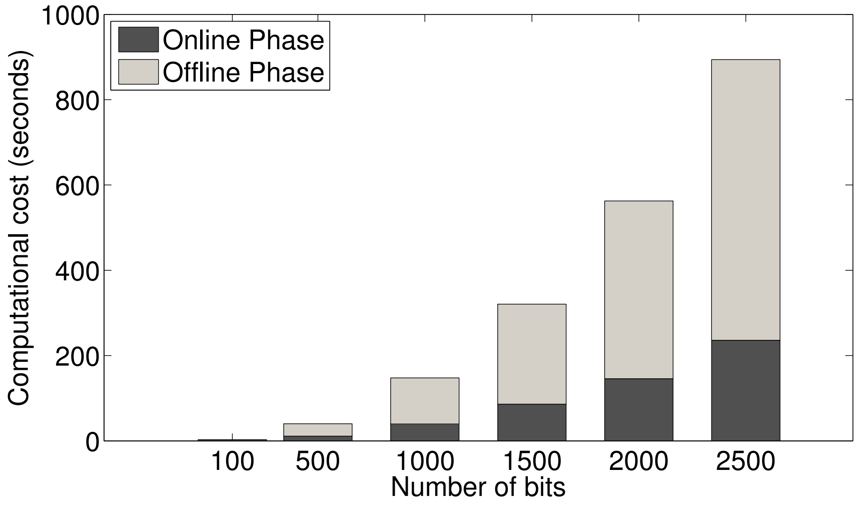

Fig. 2. Numerical results.

Results. The prototype, at its current state, requires bit randomization for the comparison methods. The task of generating such values could be executed before hand, in an "offline" phase. The "online" phase would execute the remaining tasks and would utilize the randomization values generated during the "offline" phase. Figure 2 shows such breakdown for all test scenarios considered. Moreover, we have measured the computational cost at every test instance. For the 2500 bids case, the prototype took 678.50 sec. for either sending or waiting to receive other parties' messages (note that our prototype is synchronous) and 215.52 sec. for other computational tasks, e.g., crypto-primitives. In other words, $\approx 75\%$ of the computational time was dedicated to transmission related tasks.

Our algorithm is capable of solving a 2500-bids instance in less than 15 minutes, and less than 5, when only the online phase is taken into account. Table 4 shows a more complete break down of our results.

Table 4. Overall Results

Bids	**Com. Rounds**	**Comparisons**	**CPU Time (sec)**	**On-line Phase (sec)**
100	$\approx 1.40 \cdot 10^5$	965	2.96	1.01
500	$\approx 1.96 \cdot 10^6$	14628	40.40	11.35
1000	$\approx 7.03 \cdot 10^6$	53508	147.76	39.80
1500	$\approx 15.61 \cdot 10^6$	118956	320.79	86.14
2000	$\approx 26.97 \cdot 10^6$	208132	562.50	145.78
2500	$\approx 43.15 \cdot 10^6$	330912	894.01	235.82

From these results we can conclude the following:

- The 2500 bids instance total time on the online phase is less than 5 minutes, meanwhile when the offline phase is considered, it is less than 15 minutes. In both cases this is less than the typical 30 minutes duration of period t.
- The asymptotic behaviour on the growth of the computational time seems to adjust to the behaviour included in the complexity analysis.
- The performance of the prototype could be improved by the use of techniques such as, PRSS [46], to reduce the cost of generating random bits. Moreover, other optimizations can be put in place based on the experimental setting. This could be the case for a 3-computational parties configuration.
- During our tests, $\approx 95\%$ of the computational time was spent on sorting the bids. Although the number of suppliers rarely changes with time and is relatively low, in our tests we used a fixed value of 10. Given that they are involved in the other stages of the protocol, their influence on the number of suppliers is quite limited. This means that they can be easily adjusted to other realistic scenarios without much overhead, for bigger suppliers settings.

5.1 Discussion, Adaptations and Trade-offs

Our protocol can achieve perfect security, depending on the crypto-primitives and setting. However, this security level comes with an associated price in term of performance. Realistic applications might prefer to have a more efficient solution, if any leakage produced by the optimizations does not pose a security risk for the users. With this in mind, we introduce three simple approaches as trade-offs:

1. Users can use stopping conditions for the central loop of the Algorithm 1, that resolves the auction once the bids are sorted. This stopping condition could encompass the leakage of the comparison $[c]$ at the end of each iteration, and use it in the decisional process. This of course will leak the number of iterations needed to set the clearance price.
2. To reduce the overhead of calculating comparison $[c]$ at every iteration, users can choose to do the following. At each iteration of Algorithm 1, two distinctive secret shared random values on $\mathbb{Z}_p$ are generated for anonymizing $[\nu]$ and $[\delta]$. We simply have to multiply $[\nu]$ and $[\delta]$ by the first value, then we add to both the second value and proceed to open them. Then $[c]$ could be executed in the clear. A more complete treatment of this technique can be found in [47].
3. At the final stages of our protocol, the vector of bids that was previously sorted gets randomly permuted. Our protocol uses a sorting network for this. However, lighter exchange networks can be used instead, which would result in a reduced permutation space and would thus leak some information.

Note that any stopping condition would only help to accelerate the execution of the Algorithm 1, and as mentioned before, the bulk of the processing takes place during the sorting. This also stands for any change on the configuration of the exchange network for the permutation, adding the statistical complexities of finding an adequate gate representation.

6 Conclusions

In this work we proposed an MPC-based privacy-preserving protocol for a trading market that allows users to trade their excess electricity among themselves. Our protocol employs a bidding scheme based upon MPC, and the selection of the bids and the calculation of the clearance price at which the electricity is traded are performed in a decentralised and privacy-preserving manner. We also implemented the protocol in C++ and tested its performance with realistic data. Our simulation results show its feasibility for a typical electricity trading period of, for example, 30 minutes, as the market tasks are performed for 2500 bids in less than five minutes in the online phase. Future work would include topics related to balancing suppliers' accounts based on the private volumes of the electricity that was traded, without violating privacy, will also be addressed. In addition, it would be interesting to assess the impact of the trade-offs proposed by this work as well as the impact of any possible optimization on the underlying MPC implementation of the protocol, e.g., the use of PRSS.

Acknowledgments. This work was supported by KIC InnoEnergy SE via KIC innovation project "Secured smart grid metering architecture (SAGA)", European Commission FP7 project "Harmonized framework allowing a sustainable and robust identity for European Citizens (EKSISTENZ)" grant number: 607049, and the European Commission through the ICT programme under contract FP7-ICT-2013-10-SEP-210076296 (PRACTICE).

References

1. Farhangi, H.: The path of the smart grid. IEEE Power and Energy Magazine **8**(1) (January 2010) 18–28
2. Mustafa, M.A., Cleemput, S., Abidin, A.: A local electricity trading market: Security analysis. In: ISGT-Europe, IEEE (2016)
3. Hart, G.W.: Nonintrusive appliance load monitoring. Proceedings of the IEEE **80**(12) (1992) 1870–1891
4. Abidin, A., Aly, A., Cleemput, S., Mustafa, M.A.: Towards a local electricity trading market based on secure multiparty computation. `http://securewww.esat.kuleuven.be/cosic/publications/article-2664.pdf` (2016)
5. McDaniel, P., McLaughlin, S.: Security and privacy challenges in the smart grid. IEEE Security Privacy **7**(3) (May 2009) 75–77
6. Kalogridis, G., Sooriyabandara, M., Fan, Z., Mustafa, M.A.: Toward unified security and privacy protection for smart meter networks. IEEE Systems Journal **8**(2) (June 2014) 641–654
7. Efthymiou, C., Kalogridis, G.: Smart grid privacy via anonymization of smart metering data. In: SmartGridComm. (2010) 238–243
8. Petrlic, R.: A privacy-preserving concept for smart grids. In: Sicherheit in vernetzten Systemen 18. DFN Workshop. Books on Demand GmbH (2010) 1–14
9. Li, F., Luo, B., Liu, P.: Secure information aggregation for smart grids using homomorphic encryption. In: SmartGridComm. (2010) 327–332

10. Mustafa, M.A., Zhang, N., Kalogridis, G., Fan, Z.: MUSP: Multi-service, user self-controllable and privacy-preserving system for smart metering. In: 2015 IEEE International Conference on Communications (ICC). (June 2015) 788–794
11. Mustafa, M.A., Zhang, N., Kalogridis, G., Fan, Z.: DEP2SA: A decentralized efficient privacy-preserving and selective aggregation scheme in advanced metering infrastructure. IEEE Access **3** (2015) 2828–2846
12. Lee, W., Xiang, L., Schober, R., Wong, V.W.S.: Direct electricity trading in smart grid: A coalitional game analysis. IEEE Journal on Selected Areas in Communications **32**(7) (July 2014) 1398–1411
13. Tushar, W., Yuen, C., Smith, D.B., Poor, H.V.: Price discrimination for energy trading in smart grid: A game theoretic approach. IEEE Transactions on Smart Grid **PP**(99) (2016) 1–12
14. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, IEEE (1982) 160–164
15. 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.: Secure multiparty computation goes live. In: Financial Cryptography, Springer (2009) 325–343
16. Geisler, M.: Cryptographic protocols: theory and implementation. PhD thesis, Aarhus University Denmark, Department of Computer Science (2010)
17. Henecka, W., Kögl, S., Sadeghi, A.R., Schneider, T., Wehrenberg, I.: Tasty: tool for automating secure two-party computations. In: CCS, ACM (2010) 451–462
18. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A Framework for Fast Privacy-Preserving Computations. In: ESORICS. Volume 5283 of LNCS., Springer (2008)
19. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO. Volume 7417 of LNCS., Springer (2012) 643–662
20. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: EUROCRYPT. Volume 6632 of LNCS., Springer (2011) 169–188
21. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: ICALP (2). (2013) 645–656
22. Aly, A., Van Vyve, M.: Securely solving classical network flow problems. In Lee, J., Kim, J., eds.: ICISC 2014. Volume 8949 of LNCS., Springer (2015) 205–221
23. Aly, A., Van Vyve, M.: Practically efficient secure single-commodity multi-market auctions. In: Financial Cryptography. LNCS, Springer (2016)
24. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, ACM (1988) 1–10
25. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC, ACM (1988) 11–19
26. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. (1999) 223–238
27. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure mpc for dishonest majority or: Breaking the SPDZ limits. In: ESORICS. Volume 8134 of LNCS. Springer (2013) 1–18
28. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology **13**(1) (2000) 143–202
29. 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

30. Catrina, O., de Hoogh, S.: Secure multiparty linear programming using fixed-point arithmetic. In: ESORICS. (2010) 134–150
31. Jónsson, K.V., Kreitz, G., Uddin, M.: Secure multi-party sorting and applications. IACR Cryptology ePrint Archive **2011** (2011) 122
32. Goodrich, M.T.: Randomized shellsort: A simple data-oblivious sorting algorithm. J. ACM **58**(6) (December 2011) 27:1–27:26
33. 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
34. Laur, S., Willemson, J., Zhang, B.: Round-efficient oblivious database manipulation. In Lai, X., Zhou, J., Li, H., eds.: Information Security. Volume 7001 of LNCS. Springer (2011) 262–277
35. Czumaj, A., Kanarek, P., Kutylowski, M., Lorys, K.: Delayed path coupling and generating random permutations via distributed stochastic processes. In: SODA '99, SIAM (1999) 271–280
36. Keller, M., Scholl, P.: Efficient, oblivious data structures for mpc. IACR Cryptology ePrint Archive **2014** (2014) 137
37. Damgård, I., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: CRYPTO. Volume 2729 of LNCS., Springer (2003) 247–264
38. Launchbury, J., Diatchki, I.S., DuBuisson, T., Adams-Moran, A.: Efficient lookup-table protocol in secure multiparty computation. In: ICFP, ACM (2012) 189–200
39. Shamir, A.: How to share a secret. Commun. ACM **22**(11) (1979) 612–613
40. Asharov, G., Lindell, Y.: A full proof of the bgw protocol for perfectly secure multiparty computation. Journal of Cryptology (2015) 1–94
41. Aly, A.: Network Flow Problems with Secure Multiparty Computation. PhD thesis, Universté catholique de Louvain, IMMAQ (2015)
42. Shoup, V.: Ntl: A library for doing number theory (2001)
43. Elia. `http://www.elia.be/en/grid-data/power-generation/Solar-power-generation-data/Map` Accessed May, 2016.
44. VREG: Verbruiksprofielen elektriciteit. `http://vreg.be/nl/verbruiksprofielen-elektriciteit` Accessed May, 2016.
45. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In: PODC, ACM (1998)
46. 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 LNCS. Springer Berlin Heidelberg (2005) 342–362
47. Nojoumian, M., Stinson, D.: Efficient sealed-bid auction protocols using verifiable secret sharing. In: Information Security Practice and Experience. Volume 8434 of LNCS. Springer (2014) 302–317