Secure Computing, Economy, and Trust: A Generic Solution for Secure Auctions with Real-World Applications

Peter Bogetoft
Ivan B. Damgård
Thomas Jakobsen
Kurt Nielsen
Jakob Pagter
Tomas Toft

BRICS Report Series RS-05-18

ISSN 0909-0878 June 2005

Secure Computing, Economy, and Trust
A Generic Solution for Secure Auctions with Real-World Applications

Peter Bogetoft[*] Ivan Damgård [†] Thomas Jakobsen[†]
Kurt Nielsen[*] Jakob Pagter[†] Tomas Toft[†]

June 2005

Abstract

In this paper we consider the problem of constructing secure auctions based on techniques from modern cryptography. We combine knowledge from economics, cryptography and security engineering and develop and implement secure auctions for practical real-world problems.

In essence this paper is an overview of the research project which attempts to build auctions for real applications using secure multiparty computation.

The main contributions of this project are: A generic setup for secure evaluation of integer arithmetic including comparisons; general double auctions expressed by such operations; a real world double auction tailored to the complexity and performance of the basic primitives $+$ and $\leq$; and finally evidence that our approach is practically feasible based on experiments with prototypes.

[*]Dept. of Economics, Agricultural University, Copenhagen
[†]Dept. of Computer Science, University of Aarhus

1 Introduction

The area of secure auctions combines three different areas of research: economics (mechanism design), cryptology, and security engineering.

From economy and game theory, it is well known that many forms of auctions and trading mechanisms rely on/can benefit from a trusted third party (TTP), also known as a mediator or social planner. As an example, the presence of a TTP prevents misuse of information that must be available to play the optimal strategy. Possible misuse of this information may significantly alter the game. Also a TTP may coordinate otherwise unattainable correlated equilibria, which may be of greater value to the players than other equilibria. However, in a real application, it will often be the case that such a TTP cannot be found, or is very expensive to establish (since one basically has to counter-bribe it). In short, our goal is to investigate if a distributed software solution based on modern cryptography can provide a realistic and/or cheaper substitute for a single TTP.

Previous research in secure auctions has primarily been of a theoretical nature because the cryptographic building blocks have appeared to be too complex for practical applications.

In this paper we provide an overview of a research project which—among other things—seeks to augment the current line of research on secure auctions with a practically oriented dimension. In the paper we give

- A brief overview of related work on secure auctions.
- An overview of practical cryptographic protocols which securely implements the basic arithmetic operations (including comparisons) based on so-called threshold trust. The full detail of these protocols can be found in (30).
- An overview of two specific auctions which are practically realizable based upon the protocols above. Specifically we present a traditional double auction and a specialized version of this used for exchanging agricultural production contracts. The full detail of these auctions can be found in (31), but the details of the applications areas are held confidential due to commercial interests of the industry partners.
- Evidence that all of the above is indeed practically realizable. First we argue that the complexity of the proposed protocols and mechanisms is low, and second we report empirical results from actual software implementations.

2 Secure Auctions

Secure auctions are emerging as a field of research in its own right. In recent years a number of contributions have been made (e.g. (18; 28; 4; 5; 6; 40; 24)), and the topic is also closely related to another hot topic, namely that of secure voting.

Due to the frequent usage and theoretical simplicity, the research focus has primarily been on sealed-bid on-line auctions, where bidders submit secret bids to the auctioneer who then decides the price and the winner of the auction. If the price equals the winning offer, the auction is usually referred to as a first price auction. This differs from a second price auction, where the highest bidder wins but only pays the second highest bid. Many approaches using cryptography has been proposed. Naive solutions include the simple protection of the communication of bids from bidders to the auctioneer. In these solutions, the auctioneer is considered as trusted.

Such solutions however are vulnerable. In a first price auction, for example, a specific buyer and the auctioneer may collude to ensure that the specific buyer bids just above the second highest bid and hereby wins the auction while paying the least possible to the seller. In a second price auction the seller may collude with the auctioneer and submit an extra bid just below the highest bid hereby ensuring that his object is sold at the highest possible price. This also goes for the general Vickrey Clark Groves approach, where each participant i is given the drop in social welfare of the others caused by i's participation as information rent. As in a second score auction, submitting the right extra bid can ensure the seller extra information rent at no cost. In general, the auction literature contains numerous example of player collusions.

In this paper, our primary motivating application is the case of double auctions with many sellers and buyers (hundreds or thousands), and where a single divisible commodity is being bought and sold. Bidding in such an auction involves submitting your full strategy to an auctioneer, i.e., bidders should specify the quantities they want to sell or buy as a function of the price per unit. Based on the bids, the auctioneer then computes the so called market clearing price, i.e., the price that best balance aggregated demand and supply. Clearly, knowledge of individual bids may be of great value to others, who may use this knowledge to better their own situation. It is important to note that this does not only apply to the current auction going on. A bid contains information about the bidder's general economic situation, and such information can be (mis)used in many other contexts.

Hence, if bidders are not fully convinced that their bids are kept private, they may deviate from playing the otherwise optimal strategy.

The problems described here represent two important trends of security issues in on-line auctions, namely manipulation of the auction result and attempts to gain from knowing individual bids. Assuming that the communication of bids is secure, the auctioneer is the primary target of attacks on both off- and on-line auctions. Hence much work has been done on how to ensure the trustworthiness of the auctioneer[1].

The table below provides an incomplete list of possible security properties for secure on-line auctions in terms of some of the traditional security properties.

	Integrity	Confidentiality	Availability
Bid	bids are unaltered	nobody learn bids	bids can be made
Bidder	n/a	anonymity	n/a
Result	correctness	may be restricted	auction can be completed
Payment	non-repudiation	n/a	n/a

Of course, not all properties are relevant for every auction. As argued above, most of the work cited here, as well as our own work, focus on two properties, namely bid confidentiality and result integrity.

The way these properties are enforced is typically by means of cryptography. Using standard technology, it is straightforward to encrypt and sign bids which are then sent to the auctioneer. The primary problem to address is therefore how we can remove or reduce the need to put all our trust in a single auctioneer? The basic idea is to replace trust in one party by trust in many by *distributing* the responsibility of proper behavior amongst a number of parties. There are different ways to do this.

Semi-trusted TTP (7) and (3) use the notion of a semi-trusted TTP. This is basically an auctioneer who does not collude with any of the participants. For some applications this is useful, but it should be evident from the above examples that this approach is not sufficient in many situations, including our specific applications commented on below.

[1]There are many other threats towards auctions. Most importantly, collusion among the participants also known as bidding rings. Though in auctions with many participants bidding rings are unlikely to be successful.

Two TTPs (28) introduced the idea of splitting the responsibility of solving the auction and actually do the necessary computations among two parties. Their approach uses so-called Yao encryption, (43).

(16) make the same split but use different cryptographic tools. (39) apply the same technique to continuous double auctions.

It has also been suggested to split trust among the auctioneer and the seller, but one should note that this in fact leaves the solution vulnerable to auctioneer–seller collusion as discussed above.

Multiple TTPs This approach is based on threshold trust and multiparty computation (MPC) (see e.g. (36; 21; 13)).

The basic idea is that 1) a value (e.g. a cryptographic key) can be shared among n parties so that any subset of at most t (the threshold) parties cannot reconstruct the value, but any subset of size at least $t+1$ can; and 2) one can make computations on the shared values. One can then replace the TTP with a network of TTPs that together emulate the actions of the single TTP. Instead of trusting 1 TTP one now has to trust that at most t out of n will collude or be hacked. These TTPs could even be the participants in the auction (with a threshold $t = n - 1$). (6) calls this self-trust, as any player need only trust himself.

The notion of self-trust seems very appealing, but has limitations from both practical as well as theoretical perspectives. Theoretically it is not obvious that it is sound from a game theoretic, i.e. auction design, point of view. In (14) it is shown that for two player games one can indeed use cryptography to securely simulate the mediator (i.e. TTP) using self-trust; the crux of the matter is how to punish (in the game) players who cheat in the cryptographic protocol. In cryptographic protocol theory a player caught cheating is eliminated, but this is often not a feasible approach in game theory. Also, from a practical perspective it may not be feasible to perform a computation which require active participation from, say, 5000 producers.

2.1 Contributions and Relation to Previous Work

To our knowledge the only other secure double auction is that of (40). They realise two types of double auctions, McAfee and Yokoo, both of which only auction a single item. Our auctions handle multiple items (in fact, one of our real-life auction handles multiple items of three different goods).

Using our basic protocols as building blocks, we can realise all of the auctions described above which use trust based on multiple TTPs, i.e. trust based on MPC.

The central primitive to implement is secure comparison of integers that are shared among the TTP's. That is, compute which of two numbers is larger, without revealing further information on the numbers. We have implemented two techniques for doing this, one of which is new, and the other is similar to the construction of (28), i.e., it is based on so called Yao-encrypted circuits, which we generalize from the two-party to the multi-party case [2]. A similar generalization was done previously in (15), our solution is tecnically somewhat different but offers the same functionality. In our typical application, it is possible to preproces some data before the auction starts. This makes Yao-encryption attractive because most of the work done does not depend on the actual inputs. However, we found that even using preprocessing, Yao-encryption is not faster than the new technique—which is simpler to manage because it does not require preprocessing. We give details later on how the techniques compare.

From the perspective of implementation this paper contributes the first—to our knowledge—practically feasible implementation of the multiple TTP trust model based on MPC. We are currently only aware of similar work by Malkhi et al. (25) and Feigenbaum et al. (17). Malkhi et al. use a two TTP trust model based on Yao encryption and constructs a full system called FairPlay including a special purpose language and compiler (this system is available on-line, see (25)). They implement several functions in this system and provide benchmarks on performance. The system of Feigenbaum et al. is dedicated to a particular problem, a salary survey. Their procotol supports a multiple TTP trust model, but their current implementation only use two TTPs. In fact, the implementation of Feigenbaum et al. uses parts of the FairPlay system. Other implementations of MPC do exist, for instance within voting . Here implementations using multiple TTPs exist. We are not aware of any implementations of self-trust.

Our implemented system use a multiple TTP trust model, can compute any arithmetic function based on additions and comparisons (and in fact also multiplications). To our knowledge this is the first implementation using a multiple TTP trust model, and our results give strong empirical evidence

[2] Yao-encryption is a technique by which an algorithm (circuit) for computing a function can be "encrypted" in such a way that given the encrypted input, the function can be computed without revealing further information on the inputs

that our protocols are sufficiently efficient for real-world applications. To some extent this adresses an open challenge from Malkhi et al. (25).

We note that the work of (30) and (31) represent individual contributions which are closely related to the work presented here.

3 The Scenario

We have the following players in our protocols

- Input Clients, who supply inputs to the computation. In a typical application, input clients submits bids to an auction. The number of input clients may be several hundreds or even thousands.
- A set of n TTP's, who are responsible for executing the computation, such as computing an auction result. We assume that input clients can communicate privately with the TTP's, and also that TTP's can broadcast information to all TTP's. We want the computation to be secure, even if up to t of the TTP's are corrupted by an adversary. Typical values of (n, t) might be (3,1) or (5,2).

4 Secure Arithmetic

In the following chapter we describe how our security goals are specified and how we reach them.

4.1 Desirable Security Properties

The security properties can be defined in several ways. The traditional approach is to list the security properties that you would like the system to have and then argue that the system indeed realises these properties. A recent alternative is to use the concept of Universal Composability (UC) (8), which goes beyond looking at specific security properties such as bid confidentiality by instead defining security relative to an idealised system.

4.2 Universal Composability

In the UC framework you first describe an idealised version of your system, a so called ideal functionality which by assumption cannot be corrupted by anyone. This functionality acts as a specification of the security properties and input/output behavior you would like your protocol to have. Then you prove under appropriate cryptographic assumptions that your implementation realises the idealised functionality. This - in a well-defined technical sense - means that using the protocol is equivalent to using the ideal functionality. In other words, the protocol inherits all the security properties that the ideal functionality has. This includes those it was designed to have, but also security properties that you were not initially aware of. Another benefit is that the UC framework guarantees composability in the sense that 1) the UC security of a certain protocol is not affected by how it is used and 2) one implementation of an idealised system may be replaced by any other implementation. This composability property allows us to plug'n'play different implementations of the same system.

4.3 An Idealised Functionality for Secure Arithmetic

We informally define our ideal functionality below. For a formal definition as well as proofs that the protocols described below actually provide a UC implementation of this system see (30).

Our ideal functionality can be thought of as a computer which can securely do the following:

- Confidentially receive as input a set of integers from each input client.
- Execute one out of a number of built-in straight-line programs. Which program to execute is determined from the inputs. Each program may use the standard integer arithmetic operations and comparisons (actually, addition and comparison is sufficient for our main applications). An operation like $a \leq b$ by definition returns 1 if the condition is true and 0 otherwise. The programs are public and part of the specification of the functionality. The outputs may be any of the values computed underway.
- Send the outputs of the program to the players.

This may be seen as a definition of a totally trustworthy impartial mediator, or TTP, which can compute any function using the listed instructions. By definition of the functionality, nobody learns the input from the individual participants, the computation is executed correctly, and the intended result and only the result is made public. So if we use this functionality to implement an auctioneer, we clearly have bid confidentiality and result integrity – provided, of course, that the required computations for the auctioneer can be specified as a (set of) program of the form mentioned above. As we shall see, this is indeed possible in several scenarios.

Notice that even if each operation provided by the functionality is straight-line, we can still execute a non straight-line algorithm by calling the functionality several times, letting the choice of operation depend on previous outputs. This assumes, of course, that it is secure makes these outputs public.

4.3.1 Trust and Assumptions

In (30) an implementation of the ideal functionality described above is presented, where the joint operation the n TTPs securely realizes the ideal functionality.

The protocols are shown to be secure under standard cryptographic assumptions, namely existence of a secure public-key cryptosystem and a secure pseudorandom function. Under these assumptions, the protocols can tolerate any set of less than $n/2$ TTP's being *passively corrupt*, i.e. they may share all their information but they continue to follow the protocol. *Active* corruption, where corrupted parties may deviate from the protocol, can also be handled. Using standard methods, less than $n/3$ corruptions can be tolerated at the expense of a factor n in complexity, and still terminate the protocol correctly. However, one may argue that the focus should be on *detecting* that the protocol was not followed, rather than guaranteeing that it terminates correctly. This allows to tolerate less than $n/2$ active corruptions at a small constant factor cost, compared to the passive case. The motivation for handing active corruption this way is that in a typical application, TTP's will be large organizations, public institutions and the like, that have an interest in the auction being completed, and would suffer severe damage to their reputations if they were found to have obstructed the protocol. Hence the guarantee that any form of cheating will be detected, can be expected to strongly motivate TTPs to follow the protocol.

It is even possible to tolerate corruption of all but one TTP, however, in this case it is not possible to guarantee that the protocol terminates, no matter the cost. This requires that we use in addition a threshold homomorphic public-key cryptosystem, and incurs a significant cost for tolerating active cheating.

In general, the protocols are proved secure for a static adversary. This is because we use the pseudorandom secret sharing technique from (11), in order to reduce the required interaction, and this technique is only known to be statically secure. At the expense of some loss of efficiency, we could do without this technique and have adaptively secure protocols.

We essentially assume that the clients giving input always follow the protocol. This assumption could be removed at the expense of some efficiency, however, such participants in a typical application will be bidders in an auction, who take part because it is in their interest to do so. The chosen auction mechanisms make sure that they can expect no economic gain from providing inputs of incorrect form. Hence protecting against dishonest bidders is not our first priority, and is handled only by having the client software check that the inputs are contributed correctly.

4.4 Protocols

Appendix A contains an overview of the protocols from (30) for the interested reader. Since our goal in this paper is to report on the implementation and its implications, we only give a short summary here.

We base the representation of input values on Shamir Secret sharing modulo a prime p. Thus, input clients provide input by distributing shares of the inputs privately to the TTP's (this is implemented using public-key encryption). We use the pseudorandom secret sharing technique from (11), this allows us to create sharings of random values without interaction, and also saves work in several other cases.

This immediately allows addition, multiplication and multiplication by constants modulo p using standard techniques. Say input numbers have bit length at most w bits. Then by choosing the prime p such that it is much larger than 2^w, say $p \simeq 2^{2w}$, we can allow some number of additions and multiplications while avoiding reductions mod p, i.e., we obtain a limited number of *integer* additions and multiplications. This turns out to be sufficient for our purposes.

Comparison is more involved and seems to require that we look at individual bits of a shared number. For instance, if we know about shared numbers a, b that $0 \leq a, b < 2^l$, we can easily compute shares in the number $2^l + a - b$, and we have $a \geq b$ if and only if the $l+1$'st least significant bit of $2^l + a - b$ is set. Converting shares mod p of an unknown number to shares of individual bits is possible, but quite cumbersome (see (1)). In (30), a different approach is taken by observing that it is much easier to compute a *random* shared number together with shares of its individual bits. This can be done in a preprocessing phase. Once the inputs are supplied, we can combine the preprocessed data with the shares of $2^l + a - b$ to get securely the bit we are after. This requires an on-line effort of $O(\log w)$ communication rounds where each TTP receives a total of $O(wn \log n)$ bits. It is even possible to have a constant-round solution, at the expense of (at least) an additional factor w on the communication complexity. However, this will not be an advantage in our particular case.

Thus, the conclusions on the protocols are that secure addition requires no communication, opening of a secret value requires one round of communication, and that comparisons require several rounds, i.e., they are more expensive, but manageable. Thus when designing algorithms, the amount of comparisons should be kept at a minimum.

5 Auction Design

The application of auctions has a very long history but the theory of auctions is rather new and has developed along with the discipline of information economics. Klemperer (23) provides a recent survey. An auction is basically a set of trading rules, and auction design aims at finding rules that ensure a desired outcome. An auction may improve the allocation of goods and services, e.g. by introducing a pricing setting mechanism that leads to more profitable trading, by concentrating the market, or by making the market more transparent.

The central difficulty in auctions is that of private information. The bidders have private information, e.g. about their preferences or production costs. This information is needed - directly or indirectly - to determine an optimal allocation. On the other hand, economic agents cannot be trusted to reveal their private information unless they are given the right incentives to do so. There are at least two reasons why direct or indirect revelation cannot be accomplished. First, the agents may try to act strategically to influence the auction outcome in their own favor. By taking advantage of their information they might possibly get a better share of the reallocation gains. Secondly, even if strategic manipulations does not pay in the auction, e.g. because each agent is sufficiently small to have any significant impact, the agents may still be reluctant to reveal their private information since they may fear that it can be misused in later or parallel markets. Farmers, for example, may not trust a monopsonist processor to run an auction to reallocate production contracts since they may fear that the processor will be able to learn the marginal costs of the individual farmers and use this in subsequent negotiations.

An important group of auctions provide incentives to reveal truthful information. They are typically called direct revelation mechanisms, and a central result in mechanism design states that for any mechanism there exists a direct revelation mechanism that yields the same result. The idea of this so-called revelation principle is that an impartial mediator or social planner uses the revealed information in the best interest of the parties. To avoid that the parties would be better off giving plausibly false information, the planner must restrict the way he uses the information - and he must be able to commit to this restricted usage. In many cases, this is neither trivially nor cheaply accomplished. A mediator may be tempted or bribe to misuse the information he acquires in pursue of his own or a specific bidder's particular interests.

The lack of a trusted impartial social planner - or a trusted third party TTP to use the terminology above - that can compute the optimal outcome is hereby a major obstacle in mechanism design, see also e.g. Rothkopf et al. (33). It follows that the construction of this impartial social planner must be the core issue in research on secure auctions. The idealized system described in section 4.3 illustrates this impartial computer that may imitate the desired social planner.

In spite of the theoretical advantage of the direct revealing mechanisms, other requirements may alter the choice of auction. In more complicated allocation problems, the necessary truthful information may be overwhelmingly large and the computations too complex. However, if the participants understand the mechanism, truth-telling is a simple strategy. This is an important requirement that is emphasized in the so-called Wilson's doctrine on detail-free mechanisms, Wilson (42)[3].

[3]of the general criteria for selecting a mechanism, see e.g. Schotter (35)

5.1 The Double Auction

A relatively small fraction of the literature on auctions considers *multi-unit double auctions* where sellers and buyers reallocate multiple units of a product or a service. These auctions are sometimes called exchanges, we refer to them simple as double auctions or two-sided auctions. Some of the most important real world markets are double auctions, e.g. the typical stock exchanges.

To thoroughly study a double auction one needs an equilibrium model. Attempts have been made to introduce strategic behavior in double auctions by invoking a series of further simplifications, see e.g. Wilson (41) and Amir et al. (2). However, in general, the problem of solving for equilibria in multi-unit auctions is analytically intractable, Gordy (22, p. 450).

The literature on double auctions focuses in particular on three problems: 1) incentive compatibility (i.e. truth-telling must be an optimal bidding strategy), 2) ex post efficiency (i.e. the realization of all trades that improve social welfare) and 3) budget balancing (i.e. the aggregated value sold must equal the aggregated value bought). The two first problems follows from the so-called Myerson-Satterthwaite theorem, Myerson (27). It says that delays and failures are inevitable in private bargaining if the goods start out in the wrong hands. This follows from the central observation that in any two-persons bargaining game the seller have incentives to exaggerate its value and the buyer has incentives to pretend the value is low. There have been a few attempts to design truth-telling double auctions, see McAfee (26) and Yoon (44; 45). Attempts to solve the first two problems is typically at the cost of the third problem of balancing the budget. Fortunately, the magnitude of the three problems diminish as the number of participants grows.

The markets considered in the specific implementations of this project are two-sided and consist of a large number of participants. We therefore assume that the buyers and sellers are non-strategic price-takers. They do not speculate in the price effect of demand and supply reductions. This assumption can be justified by several observations. First, this is a two-sided auction with elastic supply and demand. Any attempt to influence the price has a smaller effect in a two-sided auction than a one-sided auction with in-elastic supply. Second, we consider a market with a high number of participants. This makes every participant marginal. Third, several empirical studies and laboratory experiments have shown that the double auctions are very stable, i.e. they are robust against strategic behavior. Test auctions with as few as 2-3 buyers and 2-3 sellers have generated almost efficient outcomes ((19) and (20)). Fourth, Satterthwaite and Williams (34) show analytically that a double auction modelled as a bayesian game converges rapidly towards ex post efficiency as the market grows.

Consider a large number of both sellers and buyers that meet in a double auction to exchange multiple items of a good. The sellers have well-defined supply schemes represented by a set of quantity-price bids $(s_1, p_1), (s_2, p_2), \ldots, (s_L, p_L)$. Here, s_l is the quantity seller i offer for sale at p_l. In this general representation, the supply scheme consists of L bids, one for each of the L possible bid prices. Likewise the buyers have well-defined demand schemes represented by a set of quantity-price bids $(d_1, p_1), (d_2, p_2), \ldots, (d_L, p_L)$. The demand and supply schemes are assumed to be monotone in the price. That is for any two prices p_h and p_l where $p_h \leq p_l$, we have $s_h \leq s_l$, i.e. a seller will supply at least the same when the price increases, and $d_h \geq d_l$, i.e. a buyer will demand at least the same when the price falls. All trade is executed at the same market clearing price. Bids to buy above and sell below the market clearing price is accepted, the remaining bids are rejected.

Now the aggregated demand/supply is found by summing up the demand/supply for each feasible market clearing price. Let I be the number of buyers, J the number of sellers, and i and j be the associated counters. For any market clearing price p_l, $l = 1, 2, \ldots, L$, the aggregated demand is given by $AD_l = \sum_{i=1}^{I} d_l^i$ and the aggregated supply is $AS_l = \sum_{j=1}^{J} s_l^j$. Also the excess demand is defined as $Z_l = AD_l - AS_l, \forall l = 1, 2, \ldots, L$. The discrete nature of the bids, requires a clearing policy. We will typically say that an (approximate) equilibrium is where Z_l is closest to zero. With price-taking behavior the optimal bidding strategy is simply to submit the true demand and/or supply schemes, see e.g. Nautz (29).

5.1.1 Double Auction Algorithm

Here we explicitly describe the iterative algorithm used to implement the double auction, and we analyze its complexity in terms of computations. The algorithm is implemented on a price grid. For simplicity, we assume that it is an equidistant grid with a distance of 1 between each price.

Consider I general demand schemes and J general supply schemes, and let $\tilde{p}_t$ be the equilibrium candidate in iteration t. An initial minimum price is $p_t^{min} = 0$ and an initial maximum price is $p_t^{max} = \max\{p_1, p_2, \ldots, p_L\}$. Also, given an initial candidate $\tilde{p}_t$, the equilibrium is found by the following algorithm:

Step 1: Compute $Z(\tilde{p}_t) = \sum_{i=1}^{I} d^i(\tilde{p}_t) - \sum_{j=1}^{J} s^j(\tilde{p}_t)$.

Step 2: Determine the sign of $Z(\tilde{p}_t)$. The result is public.

Step 3: If $Z(\tilde{p}_t) < 0$, then $p_{t+1}^{max} = \tilde{p}_t$ and $\tilde{p}_{t+1} = \tilde{p}_t - \lfloor \frac{\tilde{p}_t - p_t^{min}}{2} \rfloor$. Otherwise

Step 4: If $\tilde{p}_t = \tilde{p}_{t+1}$ we stop. Otherwise, we return to Step 1.

Appendix C.2.1 contains a program implementing this algorithm.

The excess demand at the stop price may be positive or negative. An additional rule is needed in this case to clear the market entirely. Note however that the excess demand will typically be very close to zero. To solve this problem in practice, different tie-breaking rules may be applied e.g. a small reduction (or expansion) in the total set of production rights or a rationing of the bids in question a.o.

From a computational point of view, we note that the only operations required are $+$ and $\leq$.

The complexity relies entirely on the number of participants (I buyers and J sellers) and the number of possible prices L. An evaluation of an equilibrium candidate involves $I + J + 1$ additions and 1 comparison. The bisector search requires a maximum of $\log_2 L$ iterations. Altogether a maximum of $\log_2 L(I + J + 1)$ additions and $\log_2 L$ comparisons.

Note that the only information revealed is that a given equilibrium candidate is above or below equilibrium. Clearly, this information is valueless since it follows directly from the public equilibrium price. Of course this presumes that the demand and supply schemes cannot be altered during the iterations.

5.2 The Danisco Case

We consider two different markets for trading production rights. They are both related to a major international processor of farm products, the ingredients producer Danisco. For commercial reasons, we cannot give details about the markets. One market involves a single processing plant and one involves three different processing plants. The main goal is to allocate production contracts to the producers that value them the most - taking into account also their transportation costs to the processing plants.

A production contract is the right to deliver raw products to Danisco for a given period. The raw products are inputs in the production of ingredients of various arts. The relationship between the large number of upstream producers and the single downstream processor is regulated by long term contracts. These contracts are the same for all plants and they define how the price of the raw products depends on their characteristics. In none of the markets are the producers compensated for the transportation cost. Therefore, the closeness to a processing plant is a comparative advantage.

The first market with a single processing plant, consists of several hundreds of potential participants that meet to exchange multiple units of single good production contracts to the same processing plant. This is equivalent to solving a double auction with secure computing as described in Section 5.1. In appendix B we describe and discuss the more complicated market with 3 processing plants.

6 Prototype

We are currently in the process of implementing the cryptographic protocols of (30) as well as the auctions of (31) on top of the protocols. For comparison we have also implemented the problems benchmarked in (25).

Currently our setup ignores most practical as well as a few perhaps more theoretical issues that should be handled by a commercial application. These include key management, integrity of the executed programs, etc.

Our main conclusion from implementing this prototype is that this approach is feasible in practice—for the right problems.

6.1 Architecture and Technological Choices

Our system consist of the following components which are deployed in the obvious way:

- **SmcLibrary**: the component which handles all cryptography. This is used by the TTPs as well as the Coordinator.
- **ScriptEngine**: a component which can basically evaluate JScript scripts which make calls to the SmcLibrary through an API called SmcJScript (described in Appendix C.1).
- **CoordinatorServer**: a component which handles all the “logistics”—announcing computations, receiving input, publicizing the result etc.

- **Participant**: a component which is used by participants for providing input to a computation.

All components are implemented on the Microsoft .Net platform using C# . Programs are written in JScript.Net compatible. An alternative could have been Suns Java[4].

Communication is realised via direct calls to tcp through the .Net libraries using the .Net remoting framework.

Our system use a straight-forward implementation of the cryptographic protocols described, except that we introduce an extra component called a coordinator. First, the coordinator facilitates the necessary broadcast functionality. Second, the coordinator serves as the central component controlling when an auction will be performed etc.—all information which could well be announced on an accompanying web-site of the auctioneer. Third and last, this removes the need for the TTPs to have a fixed position on the net, since they may just register immediately before an auction. By this construction, the TTPs—and in particular their secrets—need not have a fixed net location, making it harder to perform network based attacks.

6.2 Performance

We have performed tests on the primitive operations $+$, $\cdot$, and $<$, as well as the programs Double Auction, PIR. PIR is included for comparison with Malkhi et al. (25). Malkhi et al. also benchmark Median, which we have not currently implemented, and AND as well as the Billionaires problem. The two latter corresponds to our multiplication ($GF(2^8)$), see Appendix A) and comparison, respectively.

We have used two machines for testing: "laptop" (1.1GHz Intel Centrino with 2GB ram) and "desktop" (2.4GHz Intel Celeron with 512MB ram). Typically we have used the laptop for everything except the CoordinatorServer and the desktop as the CoordinatorServer, in this case the machines have been physically separated (one using the university LAN to access the global internet and one using a privately operated ADSL connection) but connected via VPN to the university LAN. This distributed setup incurs a realistic communication price and certainly does not speed up local

[4]Which would perhaps be better in terms of platform independence—a practical requirement argued in (17)—but with the emergence of cross-platform implementations of the CLR (e.g. Mono) the .Net platform may not be that bad.

computation as everything takes place on one rather than many machines. As can be seen below, some benchmarks have also been performed using a local setup where everything, including the CoordinatorServer, was run on the laptop. The purpose of this was to try and estimate actual cost of network communication.

Below, measurements in Z_p refer to 32 bit integers unless otherwise stated.

6.2.1 Performance of Multiplication

The following tables show times (in milliseconds) for multiplication in the distributed setting; (b) refers to batch execution, i.e. performing many multiplications in parallel.

(n,t)	(3,1)	(5,2)	(7,3)	(9,4)
Z_p	141	228	788	7476
$GF(2^8)$	126	197	256	884
Z_p(b)	27	95	618	4308
$GF(2^8)$(b)	5	18	105	607

As expected batch execution pays off. For multiplication we have performed the same measurements in the local setup, to get an idea of the price of communication.

(n,t)	(3,1)	(5,2)	(7,3)	(9,4)
Z_p	18	117	n/a	6634
$GF(2^8)$	5	26	149	881
Z_p(b)	12	94	797	1818
$GF(2^8)$(b)	2	15	111	939

Surprisingly it seems that as the number of TTPs grow, the relative communication overhead gets smaller. When faster implementations are available, we might get more insight on this issue. One explanation may be problems arising from the local machine having to run several TTPs at once.

6.2.2 Performance of Open

The following tables shows times for the open (decryption) operation.

(n,t)	(3,1)	(5,2)	(7,3)
Z_p	176	266	607
$GF(2^8)$	139	215	420
Z_p(b)	360	580	1293
$GF(2^8)$(b)	302	571	1380

We are a bit surprised that batch execution is slower. This is probably because the tests where run at different times, with different available bandwidth. (We aim to have verified this for the final version of this paper.)

6.2.3 Performance of Comparison

As can be seen in Appendix A, the we have two different techniques for doing comparisons. One is referred to as the Yao-based technique, and this naturally split up in two steps: Yao-encryption of a Booleand circuit and evaluation of this circuit (see A for details). The other technique is called BBM.

Our implementation of these comparison techniques is not completed, but we do have results based on preliminary code which has not yet been optimized (e.g. making sure to parallelize where possible). Still, it does provide evidence that comparisons are practically feasible, at least for a small number of TTPs (say, 3 or 5).

The following table shows some measurements using the Yao-based comparison technique for $(n, t) = (3, 1)$; the final column shows the total time in minutes rather than milliseconds.

	Encrypt	Eval	Total	Min
$w = 8$	33698	1081	34779	0.57
$w = 16$	71062	1512	72574	1.2
$w = 24$	113343	2113	115456	1.9
$w = 32$	243971	3355	247326	4.1

For $(n, t) = (5, 2)$ the numbers looks as follows:

	Encrypt	Eval	Total	Min
$w = 8$	n/a	n/a	n/a	n/a
$w = 16$	760587	3675	794262	13
$w = 24$	1640519	5128	1645647	27
$w = 32$	2691670	9694	2701364	45

It is clear that there is a huge gain in preprocessing the Yao-encryption of the circuits to be used.

For the BBM we have the following measurements:

(n,t)	(3,1)	(5,2)	(7,3)	(9,4)
millisecs	3889	8906	48297	579263

6.2.4 Performance of Auctions

We have benchmark the double auction program shown in Appendiks C.2.1.

Based on the benchmarks of comparisons the ordinary double auction certainly seem feasible for a wide range of parameters (say, a price grid of size $L = 2000$—leading to some 11 comparisons—and hundreds of participants, corresponding to the actual numbers of the market with a single processing plant).

Preliminary tests has given the following numbers which indicate, that for $(n, t) = (3, 1)$ the double auction does seem feasible:

L	10	20	40	60	80	500
ms	18592	22352	26998	72554	128835	91993
min	0.30	0.37	0.44	1.20	2.14	1.53

Concerning the market with 3 processing plants. To estimate the number of rounds needed for the auction to converge to equilibrium, we are currently working on a simulation of the suggested tatonnement. With no preliminary results on this simulation, we do not have hard evidence that the auction converge faster than reducing the size of the price grid with one per round, which is not sufficient with a price grid of size, say, $L = 2000$ (since we will need on the order of L^3, i.e., billions of rounds leading to billions of comparisons).

6.2.5 Performance of PIR

We have benchmark the PIR program shown in Appendiks C.2.2, to give an idea of how our work compares to the FairPlay system of Malkhi et al. (25). For $(n, t) = (3, 1)$ where one participant hols a database with 16 elements, computation time is about 1.9 minutes, and for $(n, t) = (5, 2)$ it is about 5.4 minutes. FairPlay can compute this function in roughly 8 seconds.

Also, recall that AND and Billionaires from Malkhi et al. (25) correspond to $\cdot$ (multiplication) and $<$ in this paper. The following table gives an overview (in seconds):

	(3,1)	(5,2)	Malkhi et al. (25)
PIR	113	327	8.65
$\cdot$	0.126	0.197	2.14
$<$	3.8	8.9	4.03

Our solution and that of Malkhi are not entirely comparable, but the trust models are also different as are the ideas behind the implementations. It does seem though that except for PIR they are within the same order of magnitude.

References

[1] Algesheimer, J., Camenisch, J., and Shoup, V. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology - CRYPTO 2002* (Berlin, 2002), M. Yung, Ed., Springer-Verlag, pp. 417–432.

[2] Amir, R., Sahi, S., Shubik, M., and Yao, S. A strategic market game with complete markets. *Journal of economic theory 51* (1990), 126–143.

[3] Baudron, O., and Stern, J. Non-interactive private auctions. In *Financial Cryptography — Fifth International Conference* (2001).

[4] Brandt, F. Cryptographic protocols for secure second-price auctions. *Lecture Notes in Computer Science 2182* (2001), 154–??

[5] Brandt, F. Private public choice, 2003.

[6] Brandt, F., and Sandholm, T. Efficient privacy-preserving protocols for multi-unit auctions. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC)* (2005), A. Patrick and M. Yung, Eds., Lecture Notes in Computer Science (LNCS), Springer.

[7] Cachin, C. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security (CCS)* (1999), pp. 120–127.

[8] Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. http://eprint.iacr.org/2000/067, 2005.

[9] Cheng, J. Q., and Wellman, M. P. The walras algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics 12* (1998), 1–24.

[10] Cheng, S.-F., Leung, E., Lochner, K. M., OMalley, K., Reeves, D. M., Schvartzman, L. J., and Wellman, M. P. Walverine: A walrasian trading agent. University of Michigan, Artificial Intelligence Laboratory, July 2003.

[11] Cramer, R., Damgård, I., and Ishai, Y. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Proceedings of the Second Theory of Cryptography Conference* (2005), pp. 342–362.

[12] Cramton, P. Electricity market design: The good, the bad and the ugly. Proceedings of the Hawaii Internatinal Conference on System Sciences, January 2003.

[13] Damgård, I. B., Cramer, R., and Nielsen, J. B. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology - EuroCrypt 2001 Proceedings* (2001), B. Pfitzmann, Ed., vol. 2045, pp. 280–300.

[14] Dodis, Y., Halevi, S., and Rabin, T. A cryptographic solution to a game theoretic problem. In *CRYPTO2000* (2000), M. Bellare, Ed., vol. 1880 of *Lecture Notes in Computer Science (LNCS)*, pp. 112–130.

[15] Donald Beaver, Silvio Micali, P. R. The round complexity of secure protocols (extended abstract)q. In *STOC 1990* (1990), pp. 503–513.

[16] Elkind, E., and Lipmaa, H. Interleaving cryptography and mechanism design: The case of online auctions. In *Financial Cryptography - Eighth International Conference* (2004), vol. 3110 of *Lecture Notes in Computer Science*, pp. 117–131.

[17] Feigenbaum, J., Pinkas, B., Ryger, R. S., and Jain, F. S. Secure Computation of Surveys. In *EU Workshop on Secure Multiparty Protocols* (2004).

[18] Franklin, M., and Reiter, M. The Design and Implementation of a Secure Auction Service. In *Proc. IEEE Symp. on Security and Privacy* (Oakland, Ca, 1995), IEEE Computer Society Press, pp. 2–14.

[19] Freidman, D. On the efficincy of experimental double auctions markets. *American Economic Review 74*, 1 (1984), 60–72.

[20] Friedman, D., and Ostroy, J. Competitivity in auction markets: An experimantal and theoretical investigation. *Economic Review 105* (1995), 22–53.

[21] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental gamer or a completeness theorem for protocols with honest majority. In *19th Symp. on Theory of Computing (STOC)* (1987), ACM, pp. 218–229.

[22] Gordy, M. B. Hedging winner's curse with multiple bids: Evidence from the portuguese treasury bill auction. *The Review of Economics and Statistics 81*, 3 (1999), 448–465.

[23] Klemperer, P. Auction theory: A guide to the literature. *Journal of Economic Survey 13*, 3 (1999), 227–286.

[24] Lipmaa, H., Asokan, N., and Niemi, V. Secure vickrey auctions without threshold trust. In *Financial Cryptography 2002* (2002), vol. 2357 of *Lecture Notes in Computer Science*.

[25] Malkhi, D., Nisan, N., Pinkas, B., and Sella, Y. Fairplay - A Secure Two-Party Computation System. In *Proceedings of the 13th USENIX Security Symposium* (2004), pp. 287–302.

[26] McAfee, R. A dominant strategy double auction. *Journal of Economic Theory 56*, 2 (1992), 434–450.

[27] Myerson, R. B., and Satterthwaite, M. A. Efficient mechanisms for bilateral trading. *Journal of Economic Theory 1*, 29 (1983), 265–281.

[28] Naor, M., Pinkas, B., and Sumner, R. Privacy preserving auctions and mechanism design. In *1st ACM Conf. on Electronic Commerce* (1999), ACM, pp. 129–139.

[29] Nautz, D. Optimal bidding in multi-unit auctions with many bidders. *Economics Letters 48* (1995), 301–306.

[30] of anonymous authors, A. S. Work in progress. 2005.

[31] of anonymous authors, A. S. Work in progress. 2005.

[32] Pekec, A., and Rothkopf, M. H. Combinatorial auction design. *Management Science 49*, 11 (2003), 1485–1503.

[33] Rothkopf, M., Teisberg, T., and Kahn, E. Why are vickrey auctions rare? *Journal of Political Economy 98*, 1 (1990), 94–109.

[34] Satterthwaite, M. A., and Williams, S. R. The rate of convergence to efficiency in the buyer's bid double auction as the market becomes large. *Review of Economic Studies 56* (1989), 477–498.

[35] Schotter, A. *in Organization With Incomplete Information*, m. majumdar ed. Cambridge University Press, 1998, ch. A Practical Person's Guide to Mechanism Selection: Some Lessons From Experimental Economics.

[36] Shamir, A. How to share a secret. *Communications of the ACM 22*, 11 (November 1979), 612–613.

[37] Vries, S. D., and Vohra, R. V. Combinatorial auctions: A survey. *INFORMS Journal on Computing 15*, 3 (2003), 284–309.

[38] Walras, L. *Eléments d'Économie Politique Pure*. 1874. Translated as: Elements of Pure Economics, by Richard D. Irwin, Homewood, Illinois, 1954.

[39] Wang, C., and fung Leung, H. Anonymity and security in continuous double auctions for internet retails market. In *Proceedings of the 37th Hawaii International Conference on System Sciences* (2004).

[40] Wang, C., Leung, H., and Wang, Y. Secure double auction protocols with full privacy protection. In *Information Security and Cryptography - ICISC 2003: 6th International Conference* (2003).

[41] Wilson, R. Incentive efficiency of double auctions. *Econometrica 53*, 5 (1985), 1101–1115.

[42] Wilson, R. *Advances in Economic Theory, Fifth World Congress*, t. f. bewley ed. Cambridge University Press, 1987, ch. Game-theoretic Analyses of Trading Processses.

[43] Yao, A. C.-C. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science* (1986), pp. 162–167.

[44] Yoon, K. The modified vickrey double auction. *Journal of Economic Theory 101* (2001), 572–584.

[45] Yoon, K. An efficient double auction. Working paper, Korea University, 2003.

A Cryptographic Protocols

In this section we provide an overview of the protocols of (30) implementing the idealised system mentioned in section 4.3. The purpose of this presentation is to give the reader a feeling for what is going on, as well as providing enough insight to understand the complexity of these protocols.

A.1 Secret Sharings

Our implementation builds on the ideas presented in (1), namely computation on secret shared values. This is combined with the techniques from (11) and some new ideas. We start off by describing two ways of sharing a secret value among a set of parties.

Additive sharings allow a number of parties to share a number, say modulo a prime p, by giving each party a number such that the sum of the numbers of all parties add up to the shared value $(\text{mod} p)$. It is clear that if the numbers are uniformly random then no proper subset of all parties will have any knowledge of the shared value, while it can be reconstructed if all parties cooperate. Thus a value can be shared among a number of parties. This form of sharing can be used in any group.

Sharings can be generalized such that if the value is shared among n parties then any t of them have no information on the actual value, while any $t+1$ can reconstruct it fully. We do this using Shamir-sharings — each party is given a unique point on a (random) polynomial, f, of degree t, where the shared value is defined to be $f(0)$. Given $t+1$ points, the polynomial is uniquely defined, thus allowing for computation of the shared value. However if t or less points are known, then all possible values of $f(0)$ are equally likely — ie. no information on the secret is known. For a full description, see (36). Note that this form of sharing can be used to share elements in any field, though the order of the field must be "large enough" compared to the number of participants such that we may have polynomials of sufficiently large degree and can give every party a unique point.

A.2 Simple Computations

Given multiple values shared using Shamir-sharing, we can compute secretly. As an example, consider the two values s_1 and s_2 shared using the two polynomials p_1 and p_2 between the parties P_1, P_2, $\ldots P_n$. If each party is given a unique, non-zero value — e.g. party P_i is given the value i — and this value is used as a basis for P_i's shares, i.e. P_i's shares are $p_b(i)$ for $b \in \{1, 2\}$, then computing a sharing of the sum of s_1 and s_2 is simple: Each party just computes the sum of the original shares and uses this as their share for the sum. Similarly, a shared value can be multiplied by a public constant, and using this, subtraction can be implemented.

More complex operations, e.g. multiplication, can also be implemented, however, this requires that the parties having shares cooperate, i.e. requiring communication between the parties.

In our implementation we use two fields, Z_p, where p is a 65-bit prime, and $GF(2^8)$. The former is used for general computation — shared values will be integers small enough so that additions mod$p$ will not lead to modular reductions and hence we can do the integer operations we need. The field $GF(2^8)$ is used for computation on individual bits. As $GF(2^8)$ has characteristic 2, x-or and logical and are simply addition and multiplication for the elements 0 and 1, both in $GF(2^8)$. Using Z_p allows computation on large values which is more efficient, while using $GF(2^8)$ facilitates computation on individual bits when that is advantageous. The latter is used in the comparison-computations of two shared "integers" discussed in more detail below.

These ideas are essentially standard and were already used, e.g., in (1). However, we improve them considerably by combining with the "pseudo-random secret sharing" technique from (11). This technique allows creating Shamir-sharings of random values without having to communicate, and this allows us to reduce the communication needed in the standard protocols. In fact, we can remove the need for private communication among the TTP's and only send information that may be seen by everyone.

A.3 Comparisons

The most complex operation we need is the comparison, i.e. $\leq$, which maps a pair of numbers into some representation of $true$ and $false$, naturally these could be represented by values in any fields, e.g. by one and zero. We have two techniques for comparisons, both of which are different from the tecnique of (1). We call the first the Bit-by-bit method (BBM), the second uses a Yao-circuit — an encrypted, binary circuit, in our case an addition circuit for binary representations of several numbers. This technique enables us to secretly evaluate any circuit in a constant number of rounds, though the actual size of the communication may depend on other parameters — in our case, the size of the circuit and therefore also the size of the communication depends on the size of the numbers, as well as the number of trusted parties. For more details on Yao-circuits, see (43) and (15). We note that the multiparty solution of (15) differs from ours in that we realize the required encryption via multiparty computation on shared keys and data, while (15) does this by having each player know a part of the key and perform his part of the encryption locally. Our solution leads to more work to create the encrypted circuit, but this is not a problem, since this work is done in a preprocessing phase. The advantage is that the encrypted circuit becomes much smaller, leading to svaings in the on-line work.

A quick sketch of our algorithms follows below. Note that they are only intended to give a brief overview, thus concreteness has been sacrificed for simplicity, however, every step of the algorithms is still present in the overview.

- Given two shared (binary) numbers, the parties construct a sharing of a number a defined as the difference of the numbers plus 2^B, where B is the maximal bit length of our numbers.
- In the case of BBM we note that the answer to the original comparison is also the answer of the comparison of $a \geq 2^B$.

  Or - since the compared numbers are smaller then 2^B - the question is whether the bit at position B in a is 1. We assume without loss of generality that the numbers we are comparing are not equal[5], and proceed as follows:

  - Compute a sharing of a random number b unknown to all parties, and a sharing of its B least significant bits $b_0, \dots b_{B-1}$. This can be done using standard methods. b should have bit length $B + \kappa$ bits where κ is a security parameter.
  - Compute and open $a + b$ (which statistically hides a, since b was random and κ bits longer than a). Now observe that the question whether the bit at position B in a is set can be decided by subtracting b from $a + b$ and figuring out whether a carry occurs at position B. This can in turn be decided from the B least significant bits of $a + b$ and b via a circuit of logarithmic depth using carry look-ahead techniques. Since we know all bits

[5]This assumption can be dropped at the expense of an additional bit by noting that that $x > y \Leftrightarrow 2x + 1 \geq 2y$.

  of $a + b$ and have sharing of the bits of b, this leads to a secure protocol with a logarithmic number of rounds.

- In the case of Yao encryption, we also exploit that it is enough to decide whether the bit in position B of a is 1. We now proceed as follows:
  - Create an additive sharing of a as in (1).
  - Each party now shares the individual bits of their part of the additive share in the field $GF(2^8)$.
  - The parties add the numbers represented by the (secret) bits just published — this is done by evaluating a Yao-circuit for addition.
  - As the bits output by the circuit simply represent the sum of the values of the additive shares, we can obtain the B'th least significant bit of a.

Addition, subtraction and multiplication by public constants are very easy operations requiring no communication at all and the creation of additive sharings in the first step above can be optimized using (11) so it only requires each TTP to broadcast two values. For BBM, the bulk of the work is partly to compute the shares of b and its bits, and partly the computation of the carry bit in the final step. For Yao encryption, the buld of the work lies in constructing the encrypted Yao-circuit.

It is important to note that our algorithm gives the option of an online vs offline tradeoff, namely that the shared b and the encrypted Yao-circuit are independent of the numbers to be computed on and can therefore be constructed in advance. In a typical application, it will be known well in advance when an auction is going to take place, and which TTP's will be involved. Assuming that the encrypted circuits are constructed in advance, evaluating these circuits on given inputs requires only local computation and no communication. This produces for both methods a quite dramatic improvement of the on-line performance over (1).

A.4 Complexity of the System

When considering the complexity of computations, communication is assumed to be the "scarce resource" rather than raw computing power. The reasoning behind using communication complexity is that the computations occur naturally in rounds, and logically a round must end before any result computed can be used in a subsequent computation — for an example, consider the description of the algorithms used for comparisons. When all parties must both send data to and receive data from each other before the computation can continue, it naturally leads to "idle time" in the computation, when parties wait for the next data to process. Thus considering the number of rounds executed is a good measurement of the complexity.

Note that as we consider rounds of communication, multiple operations may be executed simultaneously, if they do not depend on each others output, essentially giving us "free computations". In practice, however, this requires computing power and, more data to be sent and received. Thus even though round-complexity is our primary concern when considering the system, computational complexity is not ignored.

Complexity of addition: As described above, addition of shared numbers is simply implemented as each party adding two local values. Thus no communication at all is required. Multiplication by a globally known constant is similar to this, but rather than adding the values, each party multiplies his sharing with the constant.

Complexity of opening a secret value: Opening a shared value — i.e. extracting the actual value from the shares of the parties — simply consists of all parties broadcasting their share of the value. Anyone listening may then compute the secret.

Complexity of multiplication Above it was mentioned, that it is possible to multiply values using standard methods, though this requires a single round of communication. The reason for this is that although we can create a sharing of the product of two sharings simply by having all parties multiply their local shares, the degree of the polynomial used for the sharing becomes too large. Thus a round is spent on creating a sharing with a polynomial of lower degree.

Complexity of comparison: In contrast to the above operations, a comparison of two numbers requires much communication. However, there are only four steps in the Yao algorithm, each taking only a constant number of rounds, thus everything can be done in a constant number of rounds. The BBM approach requires a number of rounds proportional to the logarithm of the bit-length of the numbers computed on. This is all described in more detail in (30).

Space complexity The space-complexity of our system — i.e. how much space is used for a given sharing — is good. For “integer” values, we may choose the field in which values are shared such that it has a fairly tight bound around the values. As noted above, we share numbers in Z_p, where p is a 65-bit prime. This has been done to allow use of 32-bit numbers, resulting in a doubling of the size. For sharings of individual bits, where we use $GF(2^8)$, the blowup is clearly greater, however it is hard to choose a smaller field. For Shamir-sharings, the field must be sufficiently large to accommodate a polynomial of sufficiently high degree, as well as a point for all parties to base their part of the sharings on. Thus settling on $GF(2^8)$ is a compromise — the blowup is relatively small, while we can still accommodate a few hundred parties.

B The Danisco case: market with 3 plants

The focus in this Section is on the more complicated market with 3 processing plants. This market may consist of more than thousand potential participants that meet to exchange multiple units of 3 different goods corresponding to production contracts with the three processing plants.

The possible interdependency between the three type of contracts adds a new layer of complexity to the auction design. A widely discussed example of these issues is the trade of licenses for using radio spectrums in the US. If a city is divided into two licenses having both licenses is worth far more than the separate values of the two. On the other hand the value of two spectrum licenses for two different cities may very well be independent. This problem of handling goods that can be both complements and substitutes on the same market is not an easy task. The general approach, known as combinatorial auctions, allows the bidders' to bid on any combination of items, which in itself may be an overwhelming task. Also, the problem of selecting the winner and setting the price is in general NP-hard. Fortunately, most problems may be treated either by restricting the allowed combinations or by algorithms that finds reasonable solutions. For a survey on combinatorial auctions see e.g. Vries and Vohra (37) and Pekec and Rothkopf (32).

In general the 3 different production contracts are both substitutes and complements. Conditional sell and buy can be optimal for some producers due to *economies* or *diseconomies of scale* in both production and transportation. (31) describe this market and suggest a simplified market that does not expose the bidders to any combinatorial bidding. The suggested restrictions are:

- A buyer can bid on all 3 types but only buy one type of contract
- Given the resulting market clearing prices (the equilibrium), each buyer's most preferred contract is selected
- A seller can not condition a sale of one type on the sale of another type of contract

A tendency to economies of scale in production and transport and the fact that most producers only hold a single type of contract, suggest that the restricted flexibility have little influence on the efficiency of the market. It is for this simplified auction market that we suggest a secure auction. The suggested auction provides two important simplifications. First, the required information is traditional monotone demand and supply schemes, which makes the bidding simple and the required information suitable for practice. Second, the simplifications make the contracts mutual substitutes, which simplifies the price formation. If the contracts are mutual substitutes it is well-known that a unique equilibrium can be found by a Walrasian tatonnement (38). The suggested auction is a closed auction, where the participants submit the required information once and for all. Therefore, the challenge is to find the Walrasian equilibrium with as few secure computations as possible.

In this paper we suggest an algorithm that evaluates parallel equilibria candidates. That is $\tilde{P} + \theta e$, where $\tilde{P} = (\tilde{p}_A, \tilde{p}_B, \tilde{p}_C)$, $e = (1, 1, 1)$ and θ is a multiple of the size of the price grids, $p_l - p_{l-1} (= 1)$. A somewhat similar approach is applied in some versions of the open so-called *simultaneous ascending clock auction* used for selling power capacity, see e.g. Cramton (12). For more on discrete computations of Walrasian equilibria see e.g. Cheng et al. (9; 10).

To evaluate any equilibrium candidate, $\tilde{P} = (\tilde{p}_A, \tilde{p}_B, \tilde{p}_C)$, each buyer's optimal response is determined and subsequently the excess demand and supply on each of the three markets are calculated. We define the (approximate) equilibrium as the price vector $\hat{P} = (\hat{p}_A, \hat{p}_B, \hat{p}_C)$ that results in the smallest aggregated excess demand and no excess supply on any market:

$$\hat{P} = \arg\min_{\tilde{P}} \left\{ \sum_{k=A}^{C} Z_k(\tilde{P}) | Z_k(\tilde{P}) \geq 0 \forall k = A, B, C \right\} \tag{1}$$

Now, for any equilibrium candidate $\tilde{P}$, a buyer's preferred market is the one that gives him the largest surplus. If his demand - price relation in market k is given by $(d_k^l, p_k^l), l = 1, .., L$, his surplus is:

$$\pi_k(\tilde{p}_k) = \sum_{l:p_k^l \geq \tilde{p}_k} (d_k^l - d_k^{l-1})(p_k^l - \tilde{p}_k) \forall k = A, B, C \tag{2}$$

where $d^0 = 0$. For each possible $\tilde{P}$ the largest of the three corresponding surpluses defines his preferred market. Observe that a complete bidding strategy for a buyer can hereby be represented by a $L \times 6$ matrix, representing the quantity and resulting surplus for each of the L possible prices on each of the three type of contracts [6]. To avoid secure comparisons in selecting the optimal response, the $3L$ surplus numbers are replaced by random numbers that keeps the same ranking[7]. Also, for the equilibrium candidates, $\tilde{P}$, where no contract is attractive (all surpluses are negative), the index number selects a 0 quantity of the relatively most preferred contract. Hereby no information beyond the actual trade is revealed, e.g. it is only possible to say that a given buyer's willingness to pay is below the equilibrium price.

Since the 3 contracts are mutual substitutes we can find a lower bound on the set of possible equilibrium candidates. Let the minimum price vector P^{min} be $\tilde{P} + \theta^* e$ where θ is the largest integer thus there is no excess supply on any markets:

$$\theta^* = \arg\max_{\theta} \{\tilde{P} + \theta e | Z_k(\tilde{P} + \theta e) \geq 0, \forall k = A, B, C\} \tag{3}$$

To see that P^{min} is a global minimum, note that a decrease in the price on any one market results in weakly lower prices on the other markets. The same reasoning may be used to find a global maximum P^{max}. The interval (box) between P^{min} and P^{max} constitute a possibility set W in which the equilibrium must be found. This set may also be refined into bounds on the individual market prices. For market k we have

$$\theta_k^* = \arg\max_{\theta_k} \{\tilde{P} + \theta_k e | Z_k(\tilde{P} + \theta_k e) \geq 0\} \forall k = A, B, C \tag{4}$$

as an upper bound. With parallel price vectors, Z_k will be monotone in θ and the smallest positive value (smallest excess demand) may be found

[6]The complete representation of L^3 possible $\tilde{P}$ is to much information to submit over the Internet.

[7]This approach provides information about what processing plant a buyer prefer at some $\tilde{P}$, which may provide partial information about the buyer's transportation cost. However, a much more precise estimate of the transportation cost can be found based on public data.

by a simple bi-sector search. A better guess of an equilibrium is found by adjusting the relative prices such that with excess demand the price is adjusted upwards and downwards with excess supply. For example, if $\tilde{P}_A$ is at the P^{min} level, then the prices on market B and C are adjusted upwards[8]. The possibility set W is public information and an indication of the convergency[9].

As W shrinks the valuable information contained in the resulting Z_k become smaller. Therefore, in order to settle the final adjustments the Z_k columns may be revealed for a sufficiently small W. For example, with 5 price grids between min and max, the remaining 5^3 possible outcomes may be computed and the Z_k numbers revealed. Then the (approximate) clearing price, $\hat{P}$, can be selected by simple enumeration.

B.1 Danisco Auction Algorithm

Here we explicitly describe the algorithm used for implementing the double auction and we analyze its complexity in terms of computations.

Let $\tilde{P}_t = \{\tilde{p}^A, \tilde{p}^B, \tilde{p}^C\}$ be an equilibrium candidate and $\hat{P} = \{\hat{p}^A, \hat{p}^B, \hat{p}^C\}$ the equilibrium. Consider $3I$ general demand schemes (the demand for any possible prices l on each of the three markets) and $3J$ general supply schemes (the supply for all possible prices l on the three markets). Given the initial equilibrium candidate, lower and upper bounds on the equilibrium price vector, P^{min} and P^{max}, are found (essentially) by applying the double auction algorithm three times. Consider an initial candidate $\tilde{P}_t$ and three initial minimum prices ($p^{min}_{k,t} = 0$) and three initial maximum prices ($p^{max}_{k,t} = \max\{p_1, p_2, \ldots, p_L\}$). For each $k = A, B, C$ the following algorithm is applied:

Step 1: Each buyer's best response ($d^i_k(\tilde{P}_t) \forall i = \{1, 2, \ldots, I\}$) is found by the largest corresponding surplus (represented by an arbitrary index that is made public for the specific $\tilde{P}_t$)

Step 2: Excess demand is computed $Z_k(\tilde{P}_t) = \sum_{i=1}^{I} d^i_k(\tilde{P}_t) - \sum_{j=1}^{J} s^j_k(\tilde{P}_t)$.

[8] If $\tilde{P}_A$ equals e.g. $\tilde{P}_B$, then only the price on market C is adjusted upwards.

[9] It may be possible reduce W considerably by an arbitrage argument. The price difference between any two processing plants must fall short of the transportation costs between them since it may otherwise pay to simply ship between the two markets. This provide a good first guess of the initial $\tilde{P}$.

Step 3: The sign of $Z_k(\tilde{P}_t)$ is found and made public

Step 4: If $Z_k(\tilde{P}_t) < 0$, the $p^{max}_{k,t+1} = \tilde{p}_{k,t}$ and $\tilde{P}_{t+1} = \tilde{P}_t - e\lfloor\frac{\tilde{p}_t - p^{\min}_{k,t}}{2}\rfloor$, otherwise $p^{min}_{k,t+1} = \tilde{p}^k_t$ and $\tilde{P}_{t+1} = \tilde{P}_t + e\lfloor\frac{p^{\max}_{k,t} - \tilde{p}_t}{2}\rfloor$.

Step 5: If $\tilde{P}_t = \tilde{P}_{t+1}$ stop, otherwise return to Step 1

Again, note that we only use the operations $+$ and $\leq$, and that some innocent information is made public.

The three intermediate equilibria define three price vectors P^A, P^B and P^C. Taking coordinate wise minimum and maximum defines the public possibility set (W) in which the equilibrium is to be found. To find the equilibrium the relative prices in $\tilde{P}$ is adjusted according to a Walrasian tatonnement. The new relative prices is evaluated according to the algorithm above. However, we only have to consider the possibility set found in previous rounds.

The initial round requires 3 times as many computations as the single double auction: $3(\log_2 L(I + J + 1))$ additions and $3\log_2 L$ comparisons, besides selecting the optimal response in Step 1 (which involve no secure computing). The number of computations in the next rounds depend entirely on the size of the possibility set from the previous rounds. (31) suggest different heuristics for this convergency.

As in a single double auction, the information that a given equilibrium candidate is above or below equilibrium is valueless since it follows directly from the public equilibrium price. Also, as mentioned above, the extra information about a buyer's optimal choice of contract for the considered equilibrium candidates, is of no or little value in this setting.

C Programs

Here we present the programs mentioned above together with the API that provides the operations needed. These programs are to be executed by each TTP. As described in Section 6, the programs are written in JScript and the API is implemented in C#. These are glued together through .NET, as the JScript is compiled and executed from C#.

C.1 API

All variables are in reality pointers. Functions where a result is expected but the return type is `void`, return the result using the final argument.

The methods `equalYao` and `equalBBM` simply make to comparisons to determine the equality relation.

```
public class SmcJScript
{
   public int ResultId { get; set; }
   public int newVariable();
   public int[][] getAllInput();
   public void open(int); // decrypt
   public void open(int[]);
   public void add(int, int, int);
   public void multiply(int, int, int);
   public void multiply(int[], int[], int[]);
   public void prepareYaoComparison();
   public void compareYao(int, int, int);
   public void equalYao(int, int, int);
   public void equalBBM(int, int, int);
   public void comparatorBBM(int, int);
   public void compareBBM(int, int, int);
   public void constantInteger(int, int);
   public void constantBoolean(int, int);
   public int bool2Integer(int);
   public int getBoolean(int);// decrypt a boolean
               //value and return it as an integer
   // ...
}
```

C.2 Programs

C.2.1 Double Auction

```
/* Double Auction
  *
  * Let the be n prices: p0, p1, ..., pn-1 - this will
  * be the input of the first (pseudo) participant. Each
  * partipant submits 2n values, where either the first
  * or the last n are all 0. The first n are used if the
  * participant is a buyer and thelatter n if the
  * participant is a seller.
  *
  * For each price, the participant specify the quantum
  * he will buy if this was the clearing price.
  *
  * The aggregated sum is the addition of all inputs.
  *
  * The clearing price is where supply meets demand, i.e.
  * the price where the aggregates buyer quanta equal the
  * aggregated seller quanta. This is the same as the
  * price where the difference between buy and sell is
  * closest to zero
  *
  * The result is the clearing price
  */

public class SecureFunction
{
   public static function eval(smc : SmcJScript)
   {
     var inputs : int[][] = smc.getAllInput();
     doubleAuction(smc,inputs);
   }

   public static function
     doubleAuction(smc : SmcJScript, inputs : int[][])
   {
     var aggregatedBids : int[] = addBids(smc,inputs);
     binSearchZero(smc,aggregatedBids);
     var priceIndex = smc.ResultId;
     smc.open(inputs[0][priceIndex]);
     smc.ResultId = inputs[0][priceIndex];
   }

   public static function
     addBids(smc : SmcJScript, inputs : int[][])
   { // inputs[0]is prices
     var n = inputs[1].length;
     var res : int[] = new int[n];
     for(var i=0; i<res.length; i++)
     {
       res[i] = smc.newVariable();
       smc.constantInteger(res[i],0);
       for(var j=1; j<inputs.length; j++)
       { // take i'th bid for each participant
         smc.add(res[i],inputs[j][i],res[i]);
       }
     }
     return res;
   }

   // first n/2 are decreasing, last n/2 increasing;
   // so their difference is decreasing
   public static function
     binSearchZero(smc : SmcJScript, agg : int[])
   {
     var tmp = smc.newVariable();
     var n = agg.length/2;
     var low = 0;
     var high = n; // indexing start at 0
     var mid = 0;

     // | not here  | could be here... | not here
     // +--------------------------------------------+
     // |           | |                |    |        |
     // +--------------------------------------------+
     //              low                high
     while (low < high)
     {
       if ((low + 1) == high) high = low; // stop
       else {
         mid = Math.ceil((low + high) / 2);
         smc.compareBBM(agg[mid+n], agg[mid], tmp);
         var res = smc.getBoolean(tmp);
         if (res == 1) low = mid;
         else high = mid;
       }
     }
     smc.ResultId = low; // the array idx of the root
   }
}
```

C.2.2 PIR

```
/* PIR - Private Information Retreaval
  *
  * Participant 0 holds database (dictionary)
  *  - n items, each indexed by a key (originally 16 items
  * w/ 6 bit key and 24 bits data)
  *  - [0..n-1] are keys
  *  - [n..2n-1] are values
  *
  * Participant 1 provides index + encryption key
  *
  * The result is the indexed element added to the provided
  * encryption key
  */

public class SecureFunction
{
   public static function eval(smc : SmcJScript)
   {
     var inputs : int[][] = smc.getAllInput();
     pir(smc,inputs);
   }

   public static function
     pir(smc : SmcJScript, inputs : int[][])
   {
     var result = smc.newVariable();
     smc.constantInteger(result,0);
     // add participant 1’s encryption key
     smc.add(result,inputs[1][1],result);
     var masks = computeMasks(smc,inputs);
     var n = masks.length;
     for(var i=0 ; i<n ; i++)
     {
       var tmp = smc.newVariable();
       smc.multiply(masks[i],inputs[0][n+i],tmp);
       smc.add(result,tmp,result);
     }
     smc.ResultId = result;
   }

   private static function
     computeMasks(smc : SmcJScript, inputs : int[][])
   {
     // half of inputs are keys, half are values
     var masks : int[] = new int[inputs[0].length/2];
     for(var i=0 ; i<masks.length ; i++)
     {
       smc.equalBBM(inputs[0][i],inputs[1][0],masks[i]);
       masks[i] = smc.bool2Integer(masks[i]);
     }
     return masks;
   }
}
```

Recent BRICS Report Series Publications