A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation*

Peter Bogetoft[1], Ivan Damgård[2], Thomas Jakobsen[2], Kurt Nielsen[1], Jakob Pagter[2], and Tomas Toft[2]

[1] Department of Economics, Agricultural University, Copenhagen
[2] Department of Computer Science, University of Aarhus

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

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, we know 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. 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). Multiparty computation can be used to "implement" such a TTP in such a way that we only need to trust some fraction, say a majority, of the parties. Our goal is to investigate if this can also work in practice, and our work indicates that the answer is yes.

In this paper we give an overview of practical cryptographic protocols which securely implements basic integer operations. Detail of these protocols can be found in [7] and [20]. We also give an overview of specific types of auctions which are practically realizable based upon these protocols. Detail of these auctions can be found in [2], but the details of the applications areas are held confidential due to commercial interests of the industry partners. Finally, we give a report on the empirical results from our prototype implementation.

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. [10, 17, 3, 4, 21, 15]).

* This work is sponsored by the Danish Research Agency.

G. Di Crescenzo and A. Rubin (Eds.): FC 2006, LNCS 4107, pp. 142–147, 2006.

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 traded. Bidding in such an auction ideally involves submitting full schemes or strategies 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. 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—i.e., are used only for the purpose intented—they may deviate from playing the otherwise optimal strategy.

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]. One approach to this is to replace him by a set of n Trusted Third Parties (TTPs), where it is assumed that at most some number t of TTPs are corrupt, so called threshold trust. With this assumption, one can emulate the auctioneer via multiparty computation (MPC) (see e.g. [19, 12, 8]).

3 Contributions and Relation to Previous Work

To our knowledge the only other secure double auction is that of [21]. 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).

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, and our results give strong empirical evidence that our protocols are sufficiently efficient for real-world applications. To some extent this adresses an open challenge from Malkhi et al. [16].

We are currently only aware of similar work by Malkhi et al. [16] and Feigenbaum et al. [9]. 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 [16]). 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.

[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.

4 The Cryptographic Protocols

In our protocols we have (many) *Input Clients*, who supply inputs to the computation and a set of n TTPs, who are responsible for executing the computation, such as computing an auction result. We assume that input clients can communicate privately with the TTPs, and also that TTPs can broadcast information to all TTPs. We want the computation to be secure, even if up to t of the TTPs are corrupted by an adversary. Typical values of (n, t) might be (3,1) or (5,2).

Using Canetti's universal Composability Framework[5], we can specify what we want to achieve as an ideal functionality, which can be thought of as an incorruptible computer which can do the following:

- Confidentially receive as input a set of integers from each input client.
- Execute a built-in program. The program may use the standard integer arithmetic operations and comparisons.
  The program is public and part of the specification of the functionality.
- Send the outputs of the program to the players.

If we use a protocol securely realizing this functionality to play the role of an auctioneer, we obtain an auction with the desired security properties – assuming, of course, that the computation to done by the auctioneer can be specified using integer operations as specified above,

In [7][20], protocols realizing the above functionality are presented. 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 $n/2$ TTPs 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, although this has not yet been implemented.

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.

Since our goal in this paper is to report on the implementation and its implications, we only give a short summary of the protocols here: We use Shamir secret sharing and input clients provide input by distributing shares of the inputs privately to the TTPs. We use the pseudorandom secret sharing technique from [6], 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 using standard techniques.

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 [7],[20] (using ideas from [13]), 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.

5 Double Auction Design

A relatively small fraction of the literature on auctions considers *multi-unit double auctions*, or exchanges, where sellers and buyers reallocate multiple units of a product or a service (Klemperer [14] provides a recent survey of the literature in auctions). Important real world markets are double auctions, e.g. the typical stock exchanges. 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 are accepted, the remaining bids are rejected. The market clearing price is computed as follows: Let I be the number of buyers, J the number of sellers, and i and j be the associated counters. For any 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 $Z_l = AD_l - AS_l, \forall l = 1, 2, \ldots, L$. We then define the market clearing price to be p_l, where l is such that 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 [18]. It is easy to see that this computation can be done using the protocols we described. The correct value of l can be found by binary search using $O(\log L)$ comparisons due to the monotonicity of AD_l, AS_l. Each comparison result can be made public: once the market clearing price is public, it is also known whether $AD_l > AS_l$ for each l.

6 Prototype

We have implemented the cryptographic protocols of [7] as well as the auctions of [2] on top of the protocols. Our main conclusion from implementing this

prototype is that this approach is indeed feasible in practice. A demo of the implementation is found at http://www.sikkerhed.alexandra.dk/uk/projects/scet.htm.

Our setup ignores some practical and theoretical issues that should be handled by a commercial application. These include key management, integrity of the executed programs, etc. Further we introduce a coordinator component, facilitating, e.g., the required broadcast functionality. All code is written on the Microsoft .Net platform using C# , using the communication libraries etc. of this platform.

We present here measurements on multiplication and comparison of 32 bit integers. More details can be found in [11]. The coordinator and all but one TTP were run on seperate Win XP machines (3.1GHz dual core, 2GB ram) placed on the university LAN; the last TTP was run on another Win XP machine (1.7GHz, 512MB ram) accessing the coordinator via a ADSL internet connection (1024/256 bits/s) over a VPN connection. The first table show times (in milliseconds) for doing x multiplications. Parallel execution is faster since the same amount of data can be sent in fewer rounds of communication. This is reflected in the tables below, where our parallel measurements have been fitted into a linear approximation, $ax+b$, to estimate this constant (see [11] for further details).

(n,t)	(3,1)	(5,2)	(7,3)
sequential execution	$42x$	$47x$	$70x$
parallel execution	$3x + 41$	$7x + 43$	$29x + 44$

The next table shows times for doing x comparisons (time in milliseconds).

(n,t)	(3,1)	(5,2)	(7,3)
pre-processing (s)	$420x$	$680x$	$1780x$
pre-processing (p)	$320x + 90$	$580x + 90$	$1700x + 90$
evaluation	$354x$	$405x$	$617x$

Based on the benchmarks of comparisons the double auction certainly seem feasible for a wide range of parameters (say, a price grid of size $L = 2000$—leading to some 11 comparisons—corresponding to the actual numbers of real-world markets as described in [2]).

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. Bogetoft, P., and Nielsen, K. Work in progress. 2005.
3. Brandt, F. Cryptographic protocols for secure second-price auctions. *Lecture Notes in Computer Science 2182* (2001), 154–??
4. 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.

5. Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. http://eprint.iacr.org/2000/067, 2005.
6. 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.
7. Damgård, I., and Toft, T. Work in progress. 2005.
8. 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.
9. Feigenbaum, J., Pinkas, B., Ryger, R. S., and Jain, F. S. Secure Computation of Surveys. In *EU Workshop on Secure Multiparty Protocols* (2004).
10. 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.
11. From, S. L., and Jakobsen, T. Secure Multi-Party Computation on Integers. Master's thesis, Department of Computer Science, University of Aarhus, 2006. In preparation.
12. 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.
13. I.Damgård, N.Fitzi, J.Nielsen, and T.Toft. How to split a shared secret into shared bits in constant-round. *The Eprint archive, nr. 2005/140 www.iacr.org* (2005).
14. Klemperer, P. Auction theory: A guide to the literature. *Journal of Economic Survey 13*, 3 (1999), 227–286.
15. 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*.
16. 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.
17. 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.
18. Nautz, D. Optimal bidding in multi-unit auctions with many bidders. *Economics Letters 48* (1995), 301–306.
19. Shamir, A. How to share a secret. *Communications of the ACM 22*, 11 (November 1979), 612–613.
20. T.Toft. Secure integer computation with application in economics. *Progress Report, available from author, tomas@daimi.au.dk* (2005).
21. 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).