Multiparty Computation Goes Live*

Peter Bogetoft[§], Dan Lund Christensen [¶], Ivan Damgård[‡], Martin Geisler [‡], Thomas Jakobsen[¶], Mikkel Krøigaard [‡], Janus Dam Nielsen[‡], Jesper Buus Nielsen [‡], Kurt Nielsen[†], Jakob Pagter[¶], Michael Schwartzbach [‡], and Tomas Toft[††]

[†] Inst. of Food and Eesource Economics, University of Copenhagen
[‡] Department of Computer Science, University of Aarhus
§Dept. of Economics, Copenhagen Business School
¶The Alexandra Institute
†† CWI Amsterdam and TUE Eindhoven

Abstract. In this note, we briefly report on the first large-scale and practical application of multiparty computation, which took place in January 2008.

1 Introduction and History

Multiparty computation (MPC) is an extremely general subject, and a protocol enabling general secure multiparty computation is a very strong tool that can – in principle – solve almost any cryptographic protocol problem.

In multiparty computation, one considers a number of players $P_1, ..., P_n$, who initially each possess some inputs $x_1, ..., x_n$, and we then want to securely compute some function f on these inputs where $f(x_1, ..., x_n) = (y_1, ..., y_n)$ such that P_i learns y_i but no other information. This should hold, even if players exhibit some amount of adversarial behavior.

The goal can be accomplished by some interactive protocol π that the players execute. Intuitively, we want that executing π is equivalent to to having a trusted party T that receives privately x_i from P_i, computes the function, and returns y_i to each P_i. This "equivalence" is not only intuition, but can be formalized using, for instance, Canetti's Universal Composability framework[5].

The general theory of MPC was founded in the late 80-ties [16, 3, 7]. The theory was later developed in several ways in many papers by different authors – see for instance [21, 18, 8]. An overview of the theoretical results known can be found in [6].

Despite the obvious potential that MPC has in solving a wide range of problems, we have seen virtually no practical applications of MPC in the past. This is probably in part due to the fact that direct implementation of the first general protocols would lead to very inefficient solutions. Another factor has been a general lack of understanding in the general public of the potential of the technology.

* This work was sponsored by the Danish Strategic Research Council.

A lot of research has gone into solving the efficiency problem, both for general protocols [11, 17, 9] and for special types of computations such as voting [4, 12].

The authors of this paper have been involved in two research projects SCET (Secure Computing, Economy and Trust) [1] and SIMAP (Secure Information Management and Processing) [2], that have also aimed at improving the efficiency of multiparty computation, this time with an explicit focus on range of economic applications, which we believe are particularly interesting for practical use. In the economic field of mechanism design the concept of a trusted third party has been a central assumption since the 70-ties [15, 19, 10]. Ever since the field was initiated it has grown in momentum and turned into a truly cross disciplinary field. Today, many practical mechanisms require a trusted third party. In particular, we have considered:

- Various types of auctions. This is not limited to only standard highest bid auctions with sealed bids but also includes, for instance, variants with many sellers and buyers, so called double auctions - essentially scenarios where one wants to find a fair market price for a commodity given the existing supply and demand in the market.
- Benchmarking, where several companies want to combine information on how their businesses are running, in order to compare themselves to best practice in the area. The benchmarking process is either used for learning, planning or motivation purposes. This of course has to be done while preserving confidentiality of companies' private data.

When looking at such applications, we found that the computation needed is basically elementary arithmetic on integers of moderate size, typically around 32 bits. More concretely, quite a wide range of the cases require only addition, multiplication and comparison of integers. The known generic MPC protocols can usually handle addition and multiplication very efficiently. What they really do is actually operations modulo some prime p, because the protocols are based on secret sharing over Z_p. So by choosing p large enough compared to the input numbers, we can avoid modular reductions and get integer addition and multiplication.

This is efficient because each number is shared "in one piece" using a linear secret sharing scheme, so that secure addition, for instance, requires only one local addition by each player. Unfortunately, this also implies that comparison is much harder. A generic solution would express the comparison operation as an arithmetic circuit over Z_p, but this would be far too large to give a practical solution, because the circuit would not have access to the binary representation of the inputs. So instead we developed special purpose techniques for comparison. This enables comparison in constant-round with unconditional security[13] and also a logarithmic round solution that is more practical for the size of numbers we are interested in.

[1] see http://sikkerhed.alexandra.dk/uk/projects/scet.htm.
[2] see http://sikkerhed.alexandra.dk/uk/projects/simap.htm.

The SIMAP project goes a step further and has additionally developed a domain specific programming language smcl [20]. This language allows you to express the desired computation, and specify which information should be available to which players at any given time. Such a program can then be compiled to code that will run on the players' machines and execute the appropriate protocols.

2 The Application Scenario

In this section we describe the practical case in which our system has been deployed.

In Denmark, several thousand farmers produce sugar beets, which are sold to the company Danisco, which is the only sugar producing company on the Danish market. Farmers have contracts that give them production rights, that is, a contract entitles a farmer to produce a certain amount of beets per year and deliver them to Danisco. These contracts can be traded between farmers, but trading has historically been very limited and has been done only via bilateral negotiations.

In recent years, however, the EU drastically reduced the support for sugar beet production. This and other factors meant that there was now an urgent need to reallocate contracts to farmers where productions pays off best. It was realized that this was best done via a nation-wide exchange, a double auction. The details of this mechanism and the particular business case can be found in [1, 2]. Briefly, the goal is to find the so called market clearing price, which is a price per unit of the commodity that is traded. What happens is that each buyer specifies, for each potential price, how much he is willing to buy at that price, similarly sellers say how much they are willing to sell at each price [3]. All bids go to an auctioneer, who computes, for each price, the total supply and demand in the market. Since we can assume that supply grows and demand decreases with increasing price, there is a price where total supply equals total demand, and this is the price we are looking for. Finally, all bidders who specified a non-zero amount to trade at the market clearing price get to sell/buy the amount at this price.

This could in principle be implemented with a single trusted party as the auctioneer. However, in our scenario, we have some additional security concerns implying that this is not a satisfactory solution: Bids clearly reveal information on a farmer's economic position and his productivity, and therefore farmers would be reluctant to accept Danisco acting as auctioneer, given its position in the market. Even if Danisco would never misuse its knowledge of the bids in future price negotiations, the mere fear of this happening could affect the way farmers bid and lead to a suboptimal result of the auction. On the other hand, contracts in some cases act as security for debt that farmers have to Danisco,

[3] In real life, a bidder would only specify a small number of prices, namely those where the quantity he wants to trade changes, and by how much. The quantities to trade at other prices then follow from this.

and hence the farmers' organization DKS running the auction independently would not be acceptable for Danisco. Finally, the common solution of delegating the legal and practical responsibility by paying e.g. a consultancy house to be the trusted auctioneer would be a very expensive solution.

It was therefore decided to implement an electronic double auction, where the role of the auctioneer would be played by a multiparty computation done by representatives for Danisco, DKS and the SIMAP project. A three party solution was selected, partly because it was natural in the given scenario, but also because it allowed using efficient information theoretic tools such as secret sharing, rather than (much) more expensive cryptographic methods such as homomorphic encryption.

3 The Auction System

In the system that was deployed, a web server was set up for receiving bids, and three servers were set up for doing the secure computation. Before the auction started, a public/private key pair was generated for each computation server, and a representative for each involved organization stored the private on a USB stick, protected under a password.

Each bidder logged into the webserver and an applet was downloaded to his PC together with the public keys of the computation servers. After the user typed in his bid, the applet secret shared the bids, and encrypted the shares under the server public keys. Finally the entire set of ciphertexts were stored in a database by the webserver.

As for security precautions on the client side, we did not explicitly implement any security against cheating bidders, other than verifying their identity. This is because the method used for encrypting bids implicitly gives some protection: it is a variant of the non-interactive VSS based on pseudorandom secret sharing presented in [14]. Using this method, an encrypted bid is either obviously malformed, or is guaranteed to produce consistently shares values. This means that the only cheating that is possible, is to submit bids that are not monotone, i.e., bids where, for instance, the amount you want to buy does not decrease with increasing price, as it should. It is easy to see that this cannot be to a bidders advantage. As a final word on the client-side security, we considered security against third-party attacks on client machines as being the user's responsibility, and so did not explicitly handle this issue.

After the deadline for the auction had passed, the servers were connected to the database and each other, and the market clearing price was securely computed, as well as the quantity each bidder would buy/sell at that price. The representative for each of the involved parties triggered the computation by inserting his USB stick and entering his password on his own machine.

The computation was based on standard Shamir secret sharing over a field $GF(p)$ where p was a 64 bit prime. Standard protocols with passive security were used for addition and multiplication, while a variant of the protocol from [13] was used for secure comparison. We settled for passive security because our

most important goal was to avoid that any party would need access to bids in cleartext at any point, and passive security already achieves this.

The system worked with a set of 4000 possible values for the price, meaning that after the total supply and demand has been computed for all prices, the market clearing price could be found using binary search over 4000 values, which means about 12 secure comparisons.

The bidding phase ran smoothly, with very few technical questions asked by users. The only issue was that the applet on some PC's took up to a minute to complete the encryption of the bids. It is not surprising that the applet needed a non-trivial amount of time, since each bid consisted of 4000 numbers that had to be handled individually. A total of 1200 bidders participated in the auction, each of these had the option of submitting a bid for selling, for buying, or both.

The actual computation was done January 14 and lasted about 30 minutes. Most of this time was spent on decrypting shares of the individual bids, which is not surprising, as the input to the computation consisted of about 9 million individual numbers.

As a result of the auction, about 25.000 tons of production rights changed owner.

To the best of our knowledge, this was the first large-scale and genuinely practical application of multiparty computation.

4 Evaluation and Potential

How successful have we been with the auction system, and does the technology have further potential in practice?

Other than the fact that the system worked and produced correct results, it is of course important what users think. In this connection, we can note the results of an on-line survey that was conducted simultaneously with the bidding phase. Here, about 80% of the respondents said that it was important to them that the bids were kept confidential, and also that they were happy about the confidentiality that the system offered. We find that, in particular, the fact that confidentiality is seen as important is very interesting. Also Danisco and DKS have been satisfied with the system, and say that they may well run the auction again in coming years.

In judging the further potential of multiparty computation, it is important to ask what motivated, at the end of the day, DKS and Danisco to try using such a new and untested technology? One important factor was simply the obvious need for a nation-wide exchange for production rights, which had not existed before, so the opportunity to have a cheap electronic solution - secure or not - was certainly a major reason. We do believe, however, that security also played a role. If Danisco and DKS would have tried to run the auction using conventional methods, one or more people would have had to have access to the bids, or control over the system holding the bids in cleartext. As a result, some security policy would have had to be agreed, answering questions such as: who should have access to the system and when? who has responsibility if data leaks, and what

are the consequences? Since the parties have conflicting interests, this would have lead to very lengthy discussions, possibly bringing the whole project to a halt. Alternatively, the parties might have found a solution in collaboration with a consultancy house as mediator, but this would have been a more expensive solution, and the parties would still have had to agree on whether the mediator's security policy was satisfactory. As it happened, there was no need for this kind of negotiations at all, since the multiparty computation ensured that no one needed to have access to bids at any point.

Our conclusion is that the ability of multiparty computation to keep secret *everything* that is not intended to be public, really is useful in practice, because it short-circuits discussions and concerns about which parts of the data are sensitive and what common security policy one should have for handling such data. In contrast, if some part of the system - *even a secure hardware device* - has access to the private data in cleartext, one is forced to administrate that part via a security policy that all parties can agree on. It may be time-consuming, expensive or even impossible to reach such an agreement if parties have conflicting interests. We therefore expect that multiparty computation will turn out to be useful in many practical scenarios in the future.

References

1. Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter and Tomas Toft: *A Practical Implementation of Secure Auctions based on Multiparty Integer Computation*. Proc. of Financial Cryptography 2006, Springer Verlag LNCS.
2. P. Bogetoft, K. Boye, H. Neergaard-Petersen, K. Nielsen: *Reallocating sugar beet contracts: Can sugar production survive in Denmark?*, European Review of Agricultural Economics 2007 (34):, pp. 1–20.
3. M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10.
4. Cramer, Gennaro and Schoenmakers: *A Secure and Optimally Efficient Multi-Authority Election Scheme*, Proc. of EuroCrypt 1997
5. R.Canetti: *Universally Composable Security*, The Eprint archive, www.iacr.org.
6. Cramer and Damgård: *Multiparty Computation, an Introduction*, in Contemporary Cryptology, Advanced courses in Mathematics CRM Barcelona, Birkhäuser.
7. D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19.
8. R. Cramer, I. Damgård and U. Maurer: *Multiparty Computations from Any Linear Secret Sharing Scheme*. In: Proc. EUROCRYPT '00.
9. R. Cramer, I. Damgård, S. Dziembowski, M: Hirt and T. Rabin: *Efficient Multiparty Computations With Dishonest Minority*, Proceedings of EuroCrypt 99, Springer Verlag LNCS series.
10. P. Dasgupta, P. Hammond, E. Maskin: *The Implementation of Social Choice Rules: Some General Results on Incentive Compatibility*, Review of Economic Studies 1979 (46):, pp. 27–42.

11. I. Damgård and J. Nielsen: *Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption*, Proc. of Crypto 2003, Springer Verlag LNCS.
12. Ivan Damgrd, Mads Jurik: *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System.* Public Key Cryptography 2001: 119-136
13. Ivan Damgrd, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, Tomas Toft: *Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation.* Proc. of TCC 2006, pp. 285-304, Springer Verlag LNCS.
14. Damgård and Thorbek: Non-Interactive Proofs for Integer Multiplication, proc. of EuroCrypt 2007.
15. A. Gibbard: *Manipulation of Voting Schemes: A General Result*, Econometrica 1973 (41):, pp. 587–601.
16. O. Goldreich, S. Micali and A. Wigderson: *How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority*, Proc. of ACM STOC '87, pp. 218–229.
17. R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography*, Proc of ACM PODC'98.
18. M. Hirt, U. Maurer: *Complete Characterization of Adversaries Tolerable in General Multiparty Computations*, Proc. ACM PODC'97, pp. 25–34.
19. R.B. Myerson: *Incentives Compatibility and the Bargaining Problem*, Econometrica 1979 (47):, pp. 61–73.
20. Janus Dam Nielsen and Michael I. Schwartzbach: *A domain-specific programming language for secure multipartycomputation*, Proceedings of Programming Languages and Security (PLAS), 2007, ACM press
21. T. Rabin, M. Ben-Or: *Verifiable Secret Sharing and Multiparty Protocols with Honest majority*, Proc. ACM STOC '89, pp. 73–85.