Private Public Choice*

Felix Brandt
Computer Science Department
Technical University of Munich
brandtf@cs.tum.edu

Technical Report FKI-247-03
March 2003

Abstract

The fields of social choice theory and mechanism design deal with the aggregation of conflicting preferences in a group of agents, may they be electronic or human. Two central problems in these areas are the *social choice problem* and the *mechanism design problem*. We argue that the protection of individual preferences has not been considered so far and introduce the *preference protection problem*, which we aim to solve by applying a fruitful subfield of cryptography called secure multiparty computation. Similar to the implementation of social choice functions in mechanisms, our new view on public choice adds another level to the model by introducing the emulation of mechanisms by cryptographic protocols. This enables the private and secure execution of mechanisms without trusted third-parties by distributing the computation of the mechanism outcome on the participating agents. It is shown that security against computationally bounded adversaries is possible whereas general mechanisms can not be emulated by protocols that are secure against unbounded adversaries. We then investigate how to construct efficient special-purpose protocols, such as a protocol that emulates the Clarke tax mechanism.

1 Introduction

With the growing number of electronic markets on the net, there comes a growing demand for protection of privacy in electronic mechanisms. Instead of simply having to rely on the trustworthiness of system operators, cryptography provides the tools to ensure privacy in other ways. This report intends to bring the fields of mechanism design and secure multiparty computation together by generalizing our results already obtained in the area of cryptographic auction protocols [Brandt, 2002; 2003]. We aim at constructing secure social choice mechanisms by distributing the mechanism computation on the participants themselves. This is achieved by using multiparty computation that is not based on any trusted fraction (threshold) assumptions. We show that the main reason for thresholds in the cryptographic model is a robustness requirement that can be loosened in our case. As a consequence, the correct and private execution of mechanisms can be guaranteed in the absence of trusted third-parties. We say that a protocol is *fully private* if it is secure despite any collusion of participants. In [Naor *et al.*, 1999] the first (and, to the best of our knowledge, only) scheme to privately evaluate mechanisms has been proposed.

*Slightly revised version

Besides some similarities, their approach substantially differs from ours as they use two third-parties that are assumed not to collude.

This report consists of two parts. The two following sections deal with the general feasibility of secure and private mechanisms under various conditions, whereas the fourth Section describes a cryptographic protocol that emulates the Clarke tax mechanism by computing taxes for each agent without revealing additional information (summed up valuations, identities of pivotal agents, and so forth). The report concludes with a brief outlook in Section 5.

2 Public Choice

The aggregation of conflicting preferences in a group of agents is one of the central topics of economics and multiagent systems. Two major problems have been considered in this context so far [Mas-Colell *et al.*, 1995].

Social choice problem [1] The problem is to find a function that "*fairly*" aggregates conflicting preferences. The most important theorem in this context, Arrow's impossibility theorem, states it is impossible to find such a function with unrestricted preferences under quite reasonable assumptions. When only allowing restricted preferences like so-called single-peaked preferences, fair social choice functions can be specified.

Mechanism design problem In order to be able to apply a fair social choice function, agents need to submit their preferences. The mechanism design problem is to construct mechanisms that urge self-interested agents to reveal preferences *truthfully*. Similarly to Arrow's theorem, the Gibbard-Satterthwaite theorem states the impossibility of finding such a mechanism for general preferences. However, there are solutions for restricted sub-domains, e.g. the Clarke tax mechanism for quasilinear preferences.

We believe that, similar to the recent extensions to mechanism design regarding computational aspects [Parkes, 2001; Sandholm, 2000; Larson and Sandholm, 2001], there is a need to consider privacy issues in mechanism design. In particular, we are interested in making existing mechanisms secure and private without having to rely on trusted third parties. Classically, the existence of a central institution that receives all preferences and resolves the mechanism is assumed. However, neither the correctness of the result nor the privacy of the individual inputs can be guaranteed. Especially, incentive-compatible mechanisms might deter agents from participating as they require the submission of true valuations. Confidentiality of these valuations is essential for future negotiations and its revelation can be disastrous. We therefore introduce the "preference protection problem".

Preference protection problem The problem is to enable the correct execution of a mechanism without trusted third-parties while preventing agents to learn the preferences of other participants.

We suggest that a subfield of cryptography called "secure multiparty computation" is the key to solve the preference protection problem. Similar to the implementation of social choice functions in mechanisms, our new view on public choice adds another level to the model by introducing the emulation of mechanisms with cryptographic protocols (Figure 1).

[1] This problem is called "implementation problem" in [Parkes, 2001]. However, in the economic literature "implementation problem" refers to a different problem.

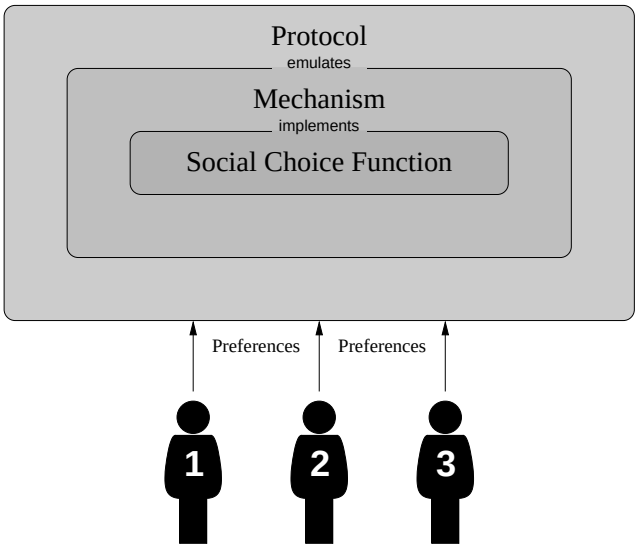

Figure 1: Cryptographic Mechanism Design

3 Secure Multiparty Computation

Secure multiparty computation (MPC) [Cramer and Damgård, 2002; Franklin *et al.*, 1992; Cramer, 2000] deals with protocols that allow n parties to jointly compute a function $f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$ on their individual private inputs x_i, so that agent i only learns y_i but nothing else. A classic example is the so-called "millionaires' problem" [Yao, 1986] in which two millionaires want to determine who is richer without revealing their wealth.

The common model defines passive adversaries (or "eavesdropping adversaries") as agents that follow the protocol but try to derive additional information. Active adversaries, on the other hand, try to violate privacy and correctness by every means including the sending of faulty messages. Furthermore, there are two classes of protocols. The security of computational protocols is based on complexity assumptions, i.e., they are only safe against computationally polynomially bounded adversaries. Unconditional (or information-theoretic) protocols provide perfect security given that agents can communicate via private channels. Typically, secure MPC is accomplished by having each agent distribute shares of his individual input on the other participants. This has to be carried out in conjunction with a commitment scheme, so that agents can verify the correctness of shares. This primitive is called "verifiable secret sharing". In the following, the participants verifiably evaluate a Boolean circuit representing function $f(\cdot)$ with their shares as inputs and new shares as outputs. When the evaluation of the circuit is finished, agents broadcast their resulting shares and reconstruct the final result.

3.1 Unconditional MPC

Let us first consider unconditional multiparty computation and its applicability to secure mechanism design. Without making any intractability assumptions, verifiable secret sharing can only be accomplished when more than one third of the participants are honest. It has been proven that the secure computation of essential Boolean gates like OR and AND is impossible in the unconditional model. However, this can be achieved when only a minority of (passive) adversaries can pool their

knowledge. Furthermore, broadcasting, i.e. sending one message to all other agents, is not generally possible (without a trusted third-party) because it has to be guaranteed that all agents receive the *same* message. It has been shown in [Lamport *et al.*, 1982] that reliable broadcasting can be achieved in the presence of $\lfloor \frac{n-1}{3} \rfloor$ (active) adversaries in the unconditional case. Finally, agents that quit the protocol in progress render it impossible to complete the computation of $f(x_1, x_2, \ldots, x_3)$ in their absence. This is a particular problem in the final stage of a protocol as exactly simultaneous share revelation is not feasible. As a consequence, an agent is able to reconstruct the result by using the shares that have been published so far and then decide not to release his share, thus leaving the other agent uninformed about the result. However, if a majority of the participants is assumed to be cooperating, the shares can be distributed in a way that allows any majority of agents to reconstruct the original values. This ensures robustness as no minority quitting the protocol can prevent the correct execution of the protocol.

Definition 1 (Robustness)
A protocol is (strongly) *robust* if the correct computation of a function $f(x_1, x_2, \ldots, x_n)$ with private inputs $x_1, x_2, \ldots, x_n$ can always be completed.

Robustness obviously implies the critical property of fairness.

Definition 2 (Fairness)
A protocol is *fair* if no agent can learn y_i and then prevent the other participants from learning $y_1, y_2, \ldots, y_{i-1}, y_{i+1}, y_{i+2}, \ldots, y_n$.

Concluding, unconditionally secure MPC is possible if there are not more than $\lfloor \frac{n-1}{3} \rfloor$ active adversaries. Recapitulating, the reasons for thresholds in unconditional multiparty computation are:

1. Robustness, threshold: $\frac{n}{2}$
2. Feasibility of secure broadcasting, threshold: $\frac{n}{3}$
3. Feasibility of secure OR, threshold: $\frac{n}{2}$
   (even with only passive adversaries)
4. Feasibility of verifiable secret sharing, threshold: $\frac{n}{3}$
   ($\frac{n}{2}$ with error probability and broadcast channel)

Now, let us try to make weak assumptions that might allow unconditional secure MPC without thresholds. First of all, **robustness** against active adversaries in MPC is defined to allow correct completion of the computation even if active adversaries do not follow the protocol. Even when $\lfloor \frac{n-1}{2} \rfloor$ cheaters were forced to quit the protocol, there are enough agents left to compute $f(x_1, x_2, \ldots, x_n)$, including the inputs of malicious participants. When presuming that active adversaries can be detected and "kicked out", *including* their inputs, this leads to a weaker, but for our purpose sufficient, notion of robustness.

Definition 3 (Weak robustness)
A protocol is *weakly robust* if the correct computation of a function $f(X)$ of inputs supplied by non-adversaries $X \subseteq \{x_1, x_2, \ldots, x_n\}$ can always be completed.

Of course, this only makes sense if $f(\cdot)$ is defined for any number of inputs up to n. A weakly robust protocol terminates after at most $n - 1$ iterations. If participation in a mechanism is voluntary, the outcome function of a mechanism is defined for an arbitrary number of inputs n. To give an example, function $f(\cdot)$ can be the outcome function of a Vickrey auction, i.e. a function that computes the

identity of the highest bidder and the amount of the second highest bid given the individual bids as inputs. Clearly, this function is defined for any number of inputs greater than one. We will give another example in Section 4.

Public verifiability of the protocol is sufficient to provide weak robustness and verifiability can be easily achieved by using zero-knowledge proofs. Unfortunately, when abandoning strong robustness, we also lose "fairness". Typically, in the end of a protocol run, each participant holds a share of the result. As simultaneous publication of these shares is impossible, a malicious agent might quit the protocol after having learned the result but before others were able to learn it. There are various techniques to approximate fairness by gradually releasing parts of the secrets to be swapped. Another possibility is to introduce a third-party that publishes the outcome after it received all shares. This third-party does not learn confidential information. It is only assumed not to leave the protocol prematurely. We learned that in auctions with a single seller, it is practical to assign this role to the seller [Brandt, 2002; 2003].

Providing a secure **broadcast** channel can eliminate the second threshold. As shown in [Pedersen, 1991] **verifiable secret sharing** can only provide *unconditional* security of either the shares' correctness or the secret, but not both. The latter seems much more practical since it means that the individuals' preferences can *never* be revealed. A malicious agent, however, can manipulate the protocol by applying super-polynomial computational power *during* the protocol. The impossibility of securely evaluating OR (and AND) gates, however, cannot be removed. This leads to the following proposition.

PROPOSITION 1 (UNCONDITIONAL MECHANISM EMULATION)
It is impossible to emulate arbitrary mechanisms by fully private protocols in the unconditional model, even when assuming weak robustness, providing a broadcast channel, and accepting the possibility of manipulation by computationally unbounded cheaters.

Be aware that, like the impossibility of strategy-proof implementations for *general* preferences in the Gibbard-Satterthwaite Theorem, Proposition 1 only states the impossibility of a general mapping from mechanisms to protocols, i.e., many mechanisms cannot be emulated by fully private protocols. However, there are some primitive mechanisms that can be emulated under the assumptions of Proposition 1, e.g., the sum of n input values can be computed fully private, weakly robust in the unconditional model if we accept the (theoretical) possibility of manipulations by computationally unbounded participants. Some MPC protocols work on finite fields instead on binary values. In these arithmetic protocols, addition (and thus XOR and NOT gates) are feasible while multiplication of shares is impossible (multiplication could be used to build OR or AND gates). Another example for an unconditional, fully private protocol is the Dutch auction. This protocol emulates the first-price sealed-bid auction without any intractability assumptions.

As unconditional protocols require private channels, there is a problem of message disputes. A participant that did not send a message may claim that he did, while on the other hand, a participant may state that he did not receive a message that he in fact received.

It is reasonable to isolate this conflict at the beginning of the protocol by applying the following procedure [Cramer *et al.*, 1997]. The two parties agree on a (symmetric) encryption key K and an information-theoretic secure commitment to this key, and broadcast a signed copy of the commitment. If both published commitments are equal, the two parties can henceforth communicate by broadcasting messages encrypted with the private key K. If the commitments are different, the dispute

Adversary	polynomially bounded	unbounded
passive	$n-1$	$\lfloor \frac{n-1}{2} \rfloor$
active	$\lfloor \frac{n-1}{2} \rfloor$	$\lfloor \frac{n-1}{3} \rfloor$

Table 1: General Secure Multiparty Computation Bounds

has to be resolved before the protocol itself begins.

3.2 Computational MPC

When allowing intractability assumptions, most of the reasons why unconditional MPC is impossible can be removed. The classic results are based on the existence of trapdoor one-way permutations[2] like the problem of factoring large composite numbers, or the decisional Diffie-Hellman problem (related to the difficulty of computing discrete logarithms). In this setting, primitives like broadcasting [Lamport *et al.*, 1982] and verifiable secret sharing [Pedersen, 1991], and the secure computation of OR gates are feasible without threshold assumptions. With the aid of our notion of weak robustness, this yields the following proposition.

PROPOSITION 2 (COMPUTATIONAL MECHANISM EMULATION)
Any mechanism can be emulated by a fully private, weakly robust protocol in the computational model.

The naive emulation of a mechanism can be extremely inefficient because general cryptographic multiparty computation protocols work on single bits and have excessive complexities[3]. E.g., the general purpose MPC protocol proposed in [Crépeau *et al.*, 1995] takes $\mathcal{O}(n^2l^3D)$ rounds and has a computational complexity of $\mathcal{O}(n^2l^3C)$ operations where l is a security parameter, C the size, and D the depth of the boolean circuit. Therefore, the design of efficient, specialized protocols remains a problem.

Table 1 shows the classic results of proven bounds of adversaries tolerable in general secure multiparty computation[4]. The results for the computational case have been proposed in [Goldreich *et al.*, 1987]. The bounds for unconditional adversaries have been found simultaneously by [Ben-Or *et al.*, 1988] and [Chaum *et al.*, 1988].

Recently, probabilistic homomorphic encryption has attracted attention in the context of MPC [Cramer *et al.*, 2001]. It allows more efficient MPC by sharing just one secret key instead of all input values. The computation can be performed directly on encrypted values. This results in just $\mathcal{O}(D)$ rounds and $\mathcal{O}(nlC)$ sent bits. However, this is currently only possible for factorization based encryption schemes like Paillier encryption [Paillier, 1999]. The joint generation of secret keys needed for such schemes is quite inefficient [Algesheimer *et al.*, 2002; Boneh and Franklin, 1997; Damgård and Koprowski, 2001], especially when requiring full privacy. On the other hand, key generation is only needed once at the beginning of a protocol and this kind of MPC can be very effective for large circuits. It would be nice to build an MPC scheme on a discrete logarithm based encryption technique like

[2]All the assumptions needed in the computational model can be reduced to the existence of "oblivious transfer" which can be achieved by noisy channels, trapdoor functions, or quantum channels.

[3]Faster implementations like [Gennaro *et al.*, 1998] rely on the assumption that a majority of the participants is honest.

[4]$\lfloor \frac{n-1}{2} \rfloor$ active adversaries are tolerable in the unconditional case when allowing non-zero error probability and a broadcast channel.

ElGamal [ElGamal, 1985] because distributed key generation is much simpler in such cryptosystems [Gennaro *et al.*, 1999]. Homomorphic encryption MPC and classic secret sharing MPC have in common that multiplications, and thus AND and OR gates, require more efforts (in terms of round and computational complexity) than additions. Generally, additions can be performed without any overhead, whereas multiplications need extra rounds of communication. For this reason, we sometimes say "slow" gates when speaking of multiplication, AND and OR gates in the following.

4 Efficient Mechanism Emulation

In this section, we present basic techniques that enable the construction of efficient protocols. We focus on the optimization of round complexity rather than bandwidth and computational complexity. Many mechanisms require operations like max or maxarg that cannot be evaluated by arithmetic circuits consisting of addition and multiplication gates. The obvious alternative is to use Boolean circuits that work on binary representations of the input values. However, the naive construction of a Boolean circuit that computes the outcome of a somewhat complex mechanism will be extremely inefficient because elementary operations like add and max require lots of slow gates.

As multiplications are expensive, we are using arithmetic circuits that work on binary vectors whose size is linear in the number of possible values in contrast to the standard radix representation that produces logarithmic sized bit vectors. This technique leads to a higher bandwidth demand, but enables protocols that use much less multiplications. In fact, we have been using this method to design a protocol that securely emulates the $(M+1)$st-price auction mechanism in a constant number of rounds, i.e. without *any* slow gates [Brandt, 2003].

As additions can be performed without any overhead, the computation of linear combinations of secrets can be executed in a single round. When computing on vectors of encrypted values, this means that besides addition and subtraction of vectors, multiplication with (known) matrices is feasible. Despite the impossibility of efficiently multiplying two secret values, we found a method that enables the multiplication with a random number that is unknown to any subset of participants (see [Brandt, 2003; 2002] for details) in a single round. We will denote the multiplication of vector components with random values by using a random multiplication matrix. Please note that the components M_j are jointly created and unknown to the agents.

$$\mathsf{R}^* = \begin{pmatrix} M_k & 0 & \cdots & 0 \\ 0 & M_{k-1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & M_1 \end{pmatrix} \qquad \text{(random multiplication matrix)}$$

Given these operations, we now sketch how to design a cryptographic protocol that emulates the Clarke tax mechanism.

Suppose we have n agents that intend to choose between m project choices. The private value of choice $l \in \{1, 2, \ldots, m\}$ to agent $i \in \{1, 2, \ldots, n\}$ is denoted by $v_i(l) \in \{1, 2, \ldots, k\}$. The mechanism outcome consists of the project choice g^* and the individual taxes tax_i that are defined by the following equations.

$$g^* = \mathsf{maxarg}_g \left((\mathsf{add}_{i=1}^n (v_i(g)))_{g=1}^m \right)$$
$$tax_i = \mathsf{sub}(\mathsf{add}_{h=1,h\neq i}^n (v_h(g^*)), \mathsf{max}_{l=1}^m (\mathsf{add}_{h=1,h\neq i}^n (v_h(l))))$$

We therefore need add, max, maxarg, and sub as building blocks for our secure emulation of the Clarke tax mechanism.

The protocol uses the vector representation mentioned above, e.g., the value of project choice l to agent i is denoted by[5]

$$\vec{v}_i(l) = (v_{i1}(l), v_{i2}(l), \ldots, v_{ik}(l)) = (\underbrace{0, \ldots, 0}_{v_i(l)-1}, 1, \underbrace{0, \ldots, 0}_{k-v_i(l)}) \quad .$$

The jth component of a vector is denoted by $v_{ij}(l)$.

At the beginning of the protocol, the first agent publishes m encrypted vectors containing his valuations for the m different project choices. He proves the correctness of these vectors by showing $\forall j \in \{1, 2, \ldots, k\} : \; v_{ij}(l) \in \{0, 1\}$ and $\sum_{j=1}^{k} v_{ij}(l) = 1$ in zero-knowledge manner (see [Brandt, 2002] for the detailed zero-knowledge protocols).

4.1 Addition

Given vector $\vec{v}_1$, a second agent can add his value v_2 by shifting up the given vector v_2 times, re-randomizing and publishing the resulting vector $\vec{v}$ and proving its correctness like above. Another proof of correctness is needed to guarantee that the vector was indeed shifted up. This can be done by computing $\vec{w} = (\mathsf{U}\vec{v}_1 + \vec{v})\,\mathsf{R}^*$ where

$$\mathsf{U} = \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix} \quad . \qquad \text{(upper triangular matrix)}$$

$\vec{w}$ is jointly decrypted so that only the first agent can read it. If $w_j \neq 0$ for any j lower than his private value v_1, the second agent did not correctly add his value. This can be proven by agent 1 without revealing his private value. Altogether, the addition of n private values requires n rounds. The summed up value of choice l is denoted by $\vec{s}(l)$.

4.2 Maximum and Maximum Argument

The maximum of m given vectors $\vec{s}(l)$ can be determined by computing $\left(\bigodot_{l=1}^{m} \mathsf{U}\vec{s}(l)\right)\mathsf{D}$ where

$$\mathsf{D} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix} \qquad \text{(differential matrix)}$$

and $\odot$ denotes the component-wise multiplication of vectors, i.e.

$$\begin{pmatrix} x_k \\ x_{k-1} \\ \vdots \\ x_1 \end{pmatrix} \odot \begin{pmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_1 \end{pmatrix} = \begin{pmatrix} x_k y_k \\ x_{k-1} y_{k-1} \\ \vdots \\ x_1 y_1 \end{pmatrix} \quad .$$

[5]To save space, vector components are listed horizontally (bottom-up).

	Input	Output	Multiplications	Mult. Rounds
add	vector, private value	vector	–	–
max	m vectors	vector	$m \cdot k$	$\log(m)$
maxarg	m vectors	index	–	–
sub	2 vectors	number	–	–

Table 2: Subprotocols

Altogether, the computation of the maximum needs $m \cdot k$ multiplications that can be executed parallely in $\log(m)$ multiplication rounds. As the number of different project choices m is usually quite small, this complexity should be tolerable.
The maximum argument of m values $\vec{s}(l)$ can be determined by computing $a_l = \left(\sum_{j=1}^{k} s_j(l) - w_j\right) r_l$ where r_l are jointly created random numbers and $\vec{w}$ is the maximum of the given vectors. $a_l = 0$ if $\vec{s}(l)$ contained the highest value (or one of the highest values in case of ties).

4.3 Subtraction

Finally, in order to compute the individual taxes, we need to subtract two vectors. Assuming that $\vec{s}_1$ is greater than $\vec{s}_2$, we compute $d = \sum_{j=1}^{k} \left(\mathsf{U}\left(\vec{s}_2 - \vec{s}_1\right)\right)_j$. d then contains an encryption of the difference of both values.

Table 2 summarizes the properties of the proposed building blocks. maxarg requires that the maximum has been computed before. As we need the maximum value to calculate the taxes, this generates no extra efforts (in this type of mechanism). Putting all these modules together, we can privately compute individual taxes according to the Clarke tax mechanism. The only part that needs multiplications is the maximum computation and these can be heavily parallelized (even when taking into account that $n + 1$ maxima have to be computed to calculate all taxes) so that only $\log(m)$ multiplication rounds are needed in total.

5 Conclusion

We investigated how secure multiparty computation can be used to emulate mechanisms with cryptographic protocols. In contrast to a common assumption in mechanism design, the world does *not* end after a mechanism terminated. Knowledge about agents' preferences can be of great value for future negotiations and therefore has to be protected appropriately. We have shown that security against computationally bounded adversaries is possible whereas general mechanisms can not be emulated with protocols that are secure against unbounded adversaries. We then described the construction of a special-purpose protocol that emulates the Clarke tax mechanism by using as few multiplications as possible. The resulting protocol is quite efficient in terms of round complexity. The drawback, however, is that bandwidth consumption is linear in k (the number of different valuations). In the future, we intend to further investigate the combination of cryptography and mechanism design and to construct protocols that efficiently emulate other mechanisms like tractable instances of combinatorial auctions or the dAGVA mechanism.

References

[Algesheimer *et al.*, 2002] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology - Proceedings of the 22th Annual International Cryptology Conference (CRYPTO)*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432. Springer, 2002.

[Ben-Or *et al.*, 1988] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 1–10. ACM Press, 1988.

[Boneh and Franklin, 1997] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology - Proceedings of the 17th Annual International Cryptology Conference (CRYPTO)*, volume 1294, pages 425–439. Springer, 1997.

[Brandt, 2002] F. Brandt. A verifiable, bidder-resolved auction protocol. In R. Falcone, S. Barber, L. Korba, and M. Singh, editors, *Proceedings of the 5th International Workshop on Deception, Fraud and Trust in Agent Societies (Special Track on Privacy and Protection with Multi-Agent Systems)*, pages 18–25, 2002.

[Brandt, 2003] F. Brandt. Fully private auctions in a constant number of rounds. In *Proceedings of the 7th Annual Conference on Financial Cryptography (FC)*, Lecture Notes in Computer Science. Springer, 2003. to appear.

[Chaum *et al.*, 1988] D. Chaum, C. Crépeau, and I. Damgård. Multi-party unconditionally secure protocols. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 11–19. ACM Press, 1988.

[Cramer and Damgård, 2002] R. Cramer and I. Damgård. Multiparty computation - An introduction. Lecture Notes, University of Aarhus, Department for Computer Science, 2002.

[Cramer *et al.*, 1997] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - Proceedings of the 14th Eurocrypt Conference*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[Cramer *et al.*, 2001] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology - Proceedings of the 18th Eurocrypt Conference*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300. Springer, 2001.

[Cramer, 2000] R. Cramer. Introduction to secure computation. Lecture Notes, University of Aarhus, Department for Computer Science, 2000.

[Crépeau *et al.*, 1995] C. Crépeau, J. van de Graaf, and A. Tapp. Comitted oblivious transfer and private multiparty commputation. In *Advances in Cryptology - Proceedings of the 15th Annual International Cryptology Conference (CRYPTO)*, volume 963 of *Lecture Notes in Computer Science*, pages 110–123, 1995.

[Damgård and Koprowski, 2001] I. Damgård and M. Koprowski. Practical threshold RSA signatures without a trusted dealer. In *Advances in Cryptology - Proceedings of the 18th Eurocrypt Conference*, volume 2045 of *Lecture Notes in Computer Science*, pages 152–165. Springer, 2001.

[ElGamal, 1985] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[Franklin *et al.*, 1992] M. Franklin, Z. Galil, and M. Yung. An overview of secure distributed computing. Technical Report TR CUCS-008-92, Columbia University, 1992.

[Gennaro *et al.*, 1998] R. Gennaro, M. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the 17th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 101–111. ACM Press, 1998.

[Gennaro *et al.*, 1999] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology - Proceedings of the 16th Eurocrypt Conference*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310. Springer, 1999.

[Goldreich *et al.*, 1987] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 218–229. ACM Press, 1987.

[Lamport *et al.*, 1982] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[Larson and Sandholm, 2001] K. Larson and T. Sandholm. Computationally limited agents in auctions. In *Theoretical Aspects of Reasoning about Knowledge (TARK)*, pages 169–182, 2001.

[Mas-Colell *et al.*, 1995] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, Inc., 1995.

[Naor *et al.*, 1999] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139. ACM Press, 1999.

[Paillier, 1999] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Proceedings of the 16th Eurocrypt Conference*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[Parkes, 2001] D. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 2001.

[Pedersen, 1991] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology - Proceedings of the 11th Annual International Cryptology Conference (CRYPTO)*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[Sandholm, 2000] T. Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce, Special issue on Intelligent Agents for Electronic Commerce*, 4(3):107–129, 2000.

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