On Correctness and Privacy in Distributed Mechanisms

Felix Brandt and **Tuomas Sandholm**
{brandtf,sandholm}@cs.cmu.edu

Carnegie Mellon University, Pittsburgh PA 15213, USA

Abstract. Mechanisms that aggregate the possibly conflicting preferences of individual agents are studied extensively in economics, operations research, and lately computer science. Perhaps surprisingly, the classic literature assumes *participating agents* to act selfishly, possibly untruthfully, if it is to their advantage, whereas the *mechanism center* is usually assumed to be honest and trustworthy. We argue that cryptography offers various concepts and building blocks to ensure the secure, *i.e.*, correct and private, execution of mechanisms. We propose models with and without a center that guarantee correctness and preserve the privacy of preferences relying on diverse assumptions such as the trustworthiness of the center or the hardness of computation. The decentralized model in which agents jointly "emulate" a virtual mechanism center is particularly interesting for two reasons. For one, it provides privacy without relying on a trusted third-party. Second, it enables the provably correct execution of *randomized* mechanisms (which is not the case in the centralized model). We furthermore point out how untruthful and multi-step mechanisms can improve privacy. In particular, we show that the fully private emulation of a preference elicitor can result in unconditional privacy of a (non-empty) subset of preferences.

1 Introduction

Mechanisms that aggregate the possibly conflicting preferences of individual agents are studied extensively in economics, operations research, and lately computer science. Distributed mechanisms are used in such diverse application areas as auctions, voting, resource sharing, routing, or task assignment.

In the heart of a mechanism lies the so-called "mechanism center" or "mechanism infrastructure" to which agents privately send reports about their preferences and that computes the mechanism outcome (*e.g.*, allocation of goods, election winner, network route, *etc.*). The main focus of existing work has been on creating mechanisms whose outcome has various desirable properties (efficient computability has been added recently) and in which agents have an incentive to submit their preferences truthfully. Perhaps surprisingly, the classic literature assumes *participating agents* to act selfishly, possibly untruthfully, if it is to their advantage, whereas the *mechanism center* is usually assumed to be honest and trustworthy, even when it has an incentive to be untruthful (*e.g.*, by overstating the second-highest bid in Vickrey auctions if it gains a fraction of the selling price).

In this paper, we investigate how to ensure the *correctness of the mechanism outcome* and the *privacy of individual preferences* by using various building blocks that

have been developed in the field of cryptography over the years. As a matter of fact, cryptography and mechanism design have major objectives in common. To a large extent, both fields are concerned about agents who deviate from given distributed mechanisms (respectively protocols) in an undesirable manner (regarding some definition of fairness, social welfare, *etc.*). However, while cryptography traditionally deals with "*worst-case*" adversaries, mechanism design assumes adversaries to be *rational* (according to some definition of utility). One could say that mechanisms allocate utility whereas cryptographic protocols allocate information. The proper behavior of agents in a mechanism is usually enforced by making deviations "uneconomic", *i.e.*, no *utility* can be gained by manipulating the mechanism. In cryptography, on the other hand, the correctness of a protocol is ensured by forcing agents to prove the correctness of their behavior and by ensuring that *no* strategy, regardless of rationality, will lead to a gain in *information*.

Both approaches have their advantages. Clearly, the cryptographic adversarial model is much stronger. On the other hand, proving the correctness of each protocol step usually incurs large overhead. Moreover, the notion of "incorrectness" in cryptography merely denotes messages that are malformed or invalid *per se*. Thus it cannot be used to elicit truthful behavior in the sense of mechanism design. It is impossible to actually verify whether an agent submitted his *true* preferences as this information is only known to the agent himself. At best, a mechanism can provide *incentives* to be truthful. However, providing such incentives can rule out the availability of other desirable properties like Pareto-optimality, individual rationality, or budget-balance. As a solution to circumvent these impossibility results, computational intractability has recently been used to make manipulation, *i.e.*, untruthful strategic behavior, *hard* rather than *impossible* [BTT89,CS03c]. Interestingly, using computational intractability as a barrier against manipulation has a long tradition in cryptography since Diffie and Hellman's seminal paper [DH76].

In this paper, we will use cryptographic primitives to provide correctness and privacy in distributed mechanisms. Some of these primitives are based on computational intractability while others are not. Correctness and privacy are defined as follows. Correctness means that in the end of a mechanism each agent is convinced that the outcome was computed correctly whereas privacy states that an agent does not learn anything about others' preferences that he cannot infer from the (correct) outcome and his own preferences. Correctness and privacy are not only complementary but also deeply intertwined (see Section 3). In mechanisms, privacy of preferences is crucial not only because sensible information might be of importance for future mechanisms or negotiations but also because a lack of privacy heavily affects the equilibria of mechanisms in execution. We will use the following notations. Agent i's preferences are denoted by $\theta_i \in \Theta_i$. The outcome function of a mechanism is $f : \Theta_1 \times \Theta_2 \times \cdots \times \Theta_n \rightarrow O$. θ is a short notation for $(\theta_1, \theta_2, \ldots, \theta_n)$. n is the number of agents participating in a mechanism.

The remainder of this paper is structured as follows. Section 2 summarizes existing work at the boundary between cryptography and mechanism design. Section 3 introduces basic security models that ensure correct mechanism execution, with or without

a center. In Section 4, we consider randomized, multi-step, and untruthful mechanisms in the context of correctness and privacy. The paper concludes in Section 5.

2 Related Work

In [MPS03], a connection between mechanism design and multiparty computation has been established for a purpose that is slightly different from the one we pursue in this paper. The authors integrate "cryptographic objectives" such as the wish to keep other agents from learning one's preferences, the wish to learn other agents' preferences, the wish to know the correct mechanism outcome, and the wish to prevent others from knowing it into the utility functions of agents and then investigate the possibility of incentive-compatible mechanism design in this novel framework. A similar approach without a mechanism center was recently analyzed in [HT04]. [MT99] consider a model where agents are not directly connected to the center, but are rather nodes in a more general communication network.

To our knowledge, the first paper to explicitly present a security model and generic protocol for arbitrary mechanisms was [NPS99]. This model basically consists of two centers that are assumed not to collude. The decentralized model presented in Section 3.2 is based on preliminary results lately proposed in [Bra03b]. The importance of considering the privacy of agents in distributed mechanisms has been stated in [FNR+02,FS02].

In recent years, there has been a large body of research on cryptographic auction protocols, *i.e.*, protocols that privately compute the outcome of sealed-bid auction mechanisms (*e.g.*, [Bra03a,Kik01,NPS99]). These special-purpose protocols implicitly contain security models (of which almost all are based on using more than one center).

3 Security Models

In this section, we investigate which sets of assumptions (general model, communication channels, existence of one-way functions, *etc.*) allow the provably correct execution of deterministic single-step mechanisms. In particular, we examine if *unconditional* privacy, *i.e.*, privacy that can never be breached, even when unbounded computational power becomes available, can be achieved in a given model.

In order to enable the notion of unconditional privacy in the first place, we have to make the following distinction. While we (in some models) allow unbounded computational power to breach privacy *after* the protocol/mechanism terminated, super-polynomial computational power is *not* available *during* the protocol. The deep relation between correctness and privacy makes this assumption necessary.[1] Nevertheless, this assumption seems reasonable since the time needed to perform super-polynomial computation is presumably longer than the typically short execution time of a mechanism. Furthermore, we generally assume the availability of a public key infrastructure. In some cases, we will assume "private channels" between certain parties. These are

[1] Otherwise, privacy could be breached by violating correctness (*e.g.*, by forging perfect zero-knowledge arguments)

authenticated means of communication between two parties that are unconditionally secure without relying on any computational assumption. Quantum channels, for example, would meet this definition.

It has turned out that the existence of almost all cryptographic primitives can be reduced to the existence of certain notions of one-way functions. A *one-way function* is a function that can be evaluated efficiently (in polynomial time), but there is no polynomial-time algorithm that can invert the function more accurately than guessing at random. A *one-way permutation* is a one-way function that is a bijective mapping of a domain onto itself. A *trapdoor permutation* is a one-way permutation that, given some extra information (the trapdoor), can be inverted in polynomial time. To give two examples, it has been shown that secure digital signatures exist if (and only if) there are one-way functions. Secure public-key encryption, on the other hand, is known to be feasible if trapdoor permutations exist.

The actual existence of one-way functions would imply $\mathcal{P} \neq \mathcal{NP}$. However, the reverse is not true: Although it might be hard to invert a function in the *worst-case*, it can be easy in many practical instances or even in the *average-case*. This uncertainty plus the technological assumptions one has to rely on—even when one-way functions exist—motivate the exploration of security that is based on alternative models such as in unconditionally secure multiparty computation [BGW88,CCD88].

The approach of the subsequent sections is as follows. We believe correctness to be fundamentally important and thus only consider models that guarantee correctness. On top of that, we investigate which sets of assumptions are necessary to provide differing degrees of privacy, *e.g.*, privacy that relies on a trusted center or privacy that relies on computational intractability. The proofs in the following sections will make use of cryptographic primitives such as *commitment schemes*, (computational) *zero-knowledge proofs*, and *perfect zero-knowledge arguments*[2]. We refer to cryptographic textbooks (*e.g.*, [Gol01]) for definitions of these building blocks.

3.1 Centralized Mechanism Execution

Let us first start by trying to obtain a provably correct single center without caring for privacy. It might seem to be sufficient to provide private channels, *e.g.*, based on public-key cryptography, from each agent to the center in order to transmit the preferences and a signature scheme that allows agents to sign their submissions. The center would then be able to prove the correctness of the outcome by just broadcasting all signed preferences. However, the center could collude with an agent and make him sign and send preferences that the center chooses *after* having seen preferences that were submitted earlier.[3] For this reason, publishing a so-called commitment to one's preferences *prior* to submitting the preferences becomes inevitable. By applying zero-knowledge proofs to the commitment values, we essentially get computational privacy for free, *i.e.*, without having to make further assumptions.

[2] Perfect zero-knowledge arguments are a variant of zero-knowledge proofs in which there is no information revelation even to computationally unbounded adversaries.

[3] Even if communication from the center to any agent could be prohibited, a manipulated agent would be able to send various signed messages. The center could then choose the appropriate one after it has seen all other preferences.

Theorem 1. *Correct deterministic mechanism execution can be guaranteed given that trapdoor permutations exist. Privacy can be breached by the center or exhaustive computation.*

Proof. Agents broadcast unconditionally binding commitments to preferences. Both the commitment scheme and the broadcasting can be based on the existence of one-way functions (and the availability of a signature scheme infrastructure) [LSP82,Nao89]. In order to prevent an agent from copying somebody else's commitment (and thus his preferences), each agent proves in computational zero-knowledge (which can be based on one-way functions as well [BOGG+88]) that he knows how to open the commitment, *i.e.*, he proves that he knows what he committed to. These proofs are executed sequentially to avoid the copying of proofs.[4] After all agents have submitted their commitments (this can be publicly verified due to the broadcasting), agents send information on how to open their commitments to the center using public-key encryption (based on trapdoor permutations). The center privately opens all preferences and rejects malformed by publicly opening the commitment value (on the broadcast channel). The center cannot reject preferences illegitimately as the commitment value is uniquely defined. Finally, it privately computes and declares the outcome with an accompanying proof of correctness in computational zero-knowledge. □

In order to obtain privacy that does not rely on intractability, we also consider a model that is based on somewhat stronger assumptions.

Theorem 2. *Correct deterministic mechanism execution can be guaranteed given that there are private channels from each agent to the center and one-way permutations exist. Privacy can only be breached by the center.*

Proof. Agents broadcast unconditionally hiding commitments to their preferences and sequentially prove their correctness using perfect zero-knowledge arguments. After that, agents send information on how to open their commitments to the center on private channels. The center then privately opens the preferences and rejects malformed by publicly opening the corresponding commitment. In the following, it privately computes the outcome and then declares the outcome with an accompanying argument of correctness in perfect zero-knowledge. All these operations can be based on one-way permutations. □

Zero-knowledge proofs and arguments allow the center to privately send parts of the outcome to each agent (and prove its correctness), so that a single agent learns only the part of the outcome that it is required to know. This can be of advantage in auctions so that *only* winning bidders learn about the goods they are awarded and the prices they have to pay while still guaranteeing correctness [Bra03a].

3.2 Decentralized Mechanism Execution

In order to decentralize the trust that agents need to have in a single center, the computation of the outcome can be distributed across several distinct centers. This is just a straightforward application of secure multiparty computation

[4] There are more efficient, yet less intuitive, ways of achieving the same functionality.

[GMW87,BGW88,CCD88]. It is possible to generate shares of a secret piece of information so that a single share is "worthless", *i.e.*, it reveals no information at all, but all shares put together uniquely determine the original piece of information.[5] Assume that agents distribute shares of their preferences so that each center receives one share from each agent. Multiparty computation protocols allow the centers to jointly compute the mechanism outcome using these shares. Depending on the protocol and the underlying security model, privacy may rely on computational intractability. In any case, privacy also relies on the assumption that a coalition of *all* centers is ruled out. When all centers collude and share their information, they can reconstruct each agent's private preferences. Nevertheless, this model is used in almost all existing cryptographic auction protocols. Especially the special case for two centers, as introduced by [Yao82], has been widely used.

In this section, we aim to obtain a more satisfying level of privacy by omitting the center(s) completely and letting agents resolve the mechanism by themselves. In other words, the mechanism's *participants* engage in a multiparty computation protocol. The key observation underlying this model is that if there is a coalition of *all* agents, there are no preferences left to be private. Thus, if the protocol is designed so that a coalition of up to $n-1$ agents does not gain any information, collusion becomes pointless (in order to breach privacy). This will be called "full privacy" in the following.

Definition 1 (Full privacy). *A protocol is* fully private[6] *if no information about any agent's preferences can be uncovered, other than what can be inferred from the outcome and all remaining preferences.*

By introducing full privacy, we shift the focus from mechanisms to *protocols*. These protocols enable agents to jointly determine the outcome of the mechanism by exchanging messages according to some predefined rules without revealing any information besides the outcome. We say that a protocol fully privately *emulates* a mechanism. When relying on computational intractability, *any* mechanism can be emulated by fully private protocols.

Proposition 1. *Correct mechanism execution can be guaranteed without a center given that trapdoor permutations exist. Privacy can only be breached by exhaustive computation.*[7]

It turns out that replacing intractability assumptions with the existence of unconditionally private channels (like in Section 3.1), only enables the fully private emulation of a *restricted* set of mechanisms. These mechanisms will be called "simple mechanisms" in the following.

[5] As an easy example, consider a single secret bit that is shared by choosing n bits at random so that the exclusive-or of these n bits yields the original bit. A single share, and even up to $n-1$ shares, reveal no information at all about the secret bit.

[6] In cryptographic terms, a fully private protocol is $(n-1)$-private, which means that a coalition of up to $n-1$ agents is incapable of breaching privacy.

[7] Propositions 1 and 2 are based on modifying the classic results of multiparty computation [GMW87,BGW88,CCD88] for a setting of full privacy by introducing "weak robustness". Due to a very strict security model, the original results rely on a fraction, *e.g.*, a majority, of the agents being trusted (see [Bra03b] for more details).

Definition 2 (Simple Mechanism). *A mechanism is* simple *if its outcome function is privately computable in the sense of [Kus89,CK89].* E.g., *the only Boolean outcome functions that are privately computable are of the form* $f(\theta) = B_1(\theta_1) \oplus B_2(\theta_2) \oplus \cdots \oplus B_n(\theta_n)$ *where* $B_i(\theta_i)$ *are Boolean predicates and* $\oplus$ *is the Boolean exclusive-or operator.*

There is yet no complete characterization of privately computable functions (except for special cases like Boolean [CK89] and 2-ary functions [Kus89]). However, by using known necessary conditions for private computability, it has been shown that first-price sealed-bid auctions are simple mechanisms whereas second-price sealed-bid auctions are not [BS04].

Proposition 2. *Correct mechanism execution can be guaranteed for simple mechanisms without a center given that there are private channels between all agents and one-way permutations exist. Privacy cannot be breached.*[7]

As in the previous section, both models also allow the (correct) computation of *different* outcomes (or parts of an outcome) for each agent, *e.g.*, so that losing bidders in an auction do not learn the selling price.

4 Intertwining Cryptography and Mechanism Design

In this section, we will relax three restrictions that we made so far, namely that mechanisms are deterministic, single-step, and incentive-compatible. The revelation principle, a central theorem of mechanism design, suggests that one restrict attention to direct-revelation mechanisms, *i.e.*, truthful, single-step mechanisms, as all remaining mechanisms can be simulated by (possibly randomized) direct-revelation mechanisms. Although this a striking result in mechanism design, its consequences are debatable as it does not consider the following important aspects: communication complexity, computational abilities of the agents and the center (see also [CS03b]), and, which is our main concern here, privacy of agents' preferences. But first of all, let us consider the effects of randomization on the correctness of a mechanism.

4.1 Randomized Mechanisms

Randomized mechanisms, *i.e.*, mechanisms in which the outcome function is probabilistic, are of increasing interest. It has been shown in [CS02a] that the automated design of an optimal deterministic mechanisms for a constant number of agents is $\mathcal{NP}$-complete in most settings whereas the design of randomized mechanisms for the same purpose is always tractable (by linear programming). Furthermore, randomized mechanisms are always as good or better than deterministic ones in terms of the expected value of the designer's objective (*e.g.*, a 2-item revenue maximizing auction has to be randomized [AH00,CS03a]). Finally, as mentioned above, the revelation principle only holds if we allow for the possibility that the resulting direct-revelation mechanism may be randomized.

Randomization has severe consequences on the notion of correctness. Whereas a single mechanism center can prove the correctness of the outcome of a deterministic mechanism (see Section 3.1), this is not possible for randomized mechanisms. There is no way a mechanism center can actually *prove* that it is using real random numbers in order to compute the outcome (without introducing a third-party that is assumed to reliably supply random data). The advantages of randomized mechanisms (*e.g.*, that manipulating the mechanism can be $\mathcal{NP}$-hard [CS03c]) in fact rely on the trustworthiness of the mechanism center. These advantages become void if the center is corrupt. Forcing the center to commit to its random choice before the agents submit their preferences only reduces the problem as the center might still choose a random value that is beneficial to itself (and possibly colluding agents).[8]

In the decentralized model proposed in Section 3.2, on the other hand, the "competition" between agents allows the unbiased joint generation of random numbers (unless *all* agents collude).

Theorem 3. *Randomized mechanisms can be emulated correctly by computationally fully private protocols. Randomized simple mechanisms can be emulated correctly by unconditionally fully private protocols.*

Proof. The following construction builds on a cryptographic primitive that is called "coin tossing into the well" [Blu82]. Before computing the mechanism outcome, each agent broadcasts an unconditionally hiding commitment to a freely chosen bit-string r_i and proves the correctness of the commitment (*i.e.*, that he knows what is inside) with a perfect zero-knowledge argument. These proofs are arranged sequentially to avoid proof duplication as in Theorem 1. After that, agents commit to their preferences and start emulating the mechanism. In the following computation (of the outcome), the agents can use $r = r_1 \oplus r_2 \oplus \cdots \oplus r_n$ (which can be privately computed, even in the unconditionally private model according to Definition 2) as a source of pure random data. We stress the fact, that this procedure works for the computationally private emulation of arbitrary mechanisms as well as the unconditionally private emulation of simple mechanisms. As the agents generate the random bit-string *before* committing to their preferences, the correctness of the (randomized) outcome can still be guaranteed if *all* agents collude after knowing the submitted preferences of each other for sure (in the form of commitments). This is desirable as there are mechanisms in which it might be beneficial for all agents to manipulate the randomization of the outcome after they know their submissions.[9] □

[8] Even though *truthfulness* is preserved when the random choice is known beforehand in *strongly truthful* mechanisms [MV04], other properties such as revenue maximization might be lost when agents are able to manipulate the random choice by colluding with the mechanism center.

[9] There might also be randomized mechanisms where the opposite is true, *i.e.*, *all* agents benefit from a manipulation of their preferences after they know the common random string, but we doubt that they are of any significance.

4.2 Multi-Step Mechanisms

Multi-step mechanisms are mechanisms in which the center gradually asks questions to the agents in multiple rounds until enough preferences have been elicited to compute the outcome (see *e.g.*, [CS01]). Ascending (*e.g.*, English) or descending (*e.g.*, Dutch) auctions are special cases of this definition. Besides the limited preference revelation, multi-step mechanisms have an important advantage that is not considered in cryptography: Agents do not need to completely determine their own preferences. This is important because determining one's own preferences may be tedious task and can even be intractable [San93].

While executing multi-step mechanisms in the single center models presented in Section 3.1 is straightforward, it is interesting to examine whether the fully private emulation of multi-step mechanisms is possible. By fully privately emulating a multi-step mechanism, we can improve the level of privacy guaranteed in Proposition 1 because some preferences might never be elicited and thus remain unconditionally private. However, the main problem is that queries may implicitly contain information on agents' preferences revealed so far. We define a certain class of mechanisms which always benefit from private elicitation.

Definition 3 (Privately Elicitable Mechanism). *A mechanism is called* privately elicitable *if it satisfies the following two conditions:*

- *There are cases in which a subset of the preferences is sufficient to compute the mechanism outcome.*[10]
- *There is a function that maps the mechanism outcome and the preferences of one agent to a subset of this agent's preferences consisting of all preferences that have not been elicited from him.*

The second condition ensures that agents do not learn information about others' preferences from the queries they are asked (see the proof of Theorem 4 for details). English auctions, for example, are privately elicitable mechanisms: The first condition holds because it is irrelevant (for the mechanism outcome) how much the highest bidder would have bid, as long as it is made sure that everybody else bid less than him. The second condition is satisfied because the only prices not "offered" to a bidder are prices above the selling price.

Theorem 4. *Privately elicitable mechanisms can be emulated by fully private protocols so that it is impossible to reveal* all *preferences, even by exhaustive computation.*

Proof. Without loss of generality, the preference elicitor can be emulated by the following protocol. We jointly and iteratively compute a query function and a stop-function for several rounds, and once, at the end of the protocol, evaluate the outcome function. Let $\gamma_i \in \Gamma_i$ be the (iteratively updated) set of agent i's statements on his preferences, $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, and $\Gamma = \Gamma_1 \times \Gamma_2 \times \cdots \times \Gamma_n$. All γ_i are initialized as empty sets. The following procedure is repeated round by round. The query function $q : \{1, 2, \ldots, n\} \times \Gamma \to Q$, which is fully privately computed by all agents (using

[10] Otherwise, preference elicitation would be pointless, regardless of privacy. Almost all practical mechanisms satisfy this condition.

Proposition 1), outputs a private query for each agent (Q is some set of available queries including an empty query $\perp$). Agents reply to these queries by publicly committing to their answer on a broadcast channel. Thus, γ_i is defined as the set of commitments agent i made so far. Whenever no more information is needed from a particular agent i, $q(i, \gamma_i) = \perp$. Agents reply to that by committing to an "empty answer". Agents then jointly compute the Boolean stop function $s : \Gamma \to \{0, 1\}$ and proceed to the next round (by asking more queries) if it is 0, or compute the outcome function $f' : \Gamma \to O$ if $s(\gamma) = 1$.

So far, all agents get to know the number of rounds, *i.e.*, the maximal number of queries asked to a single agent, and could infer information from that. Sometimes this information can be inferred from the outcome (*e.g.*, in an English or Dutch auction). However, as this is not the general case, the number of rounds needs to be hidden. For this reason, we execute the protocol for the maximal number of rounds that *could* be needed to compute the outcome and use the modified query function

$$q'(i, \gamma) = \begin{cases} q(i, \gamma) & \text{if } s(i, \gamma) = 0 \\ \perp & \text{otherwise} \end{cases} .$$

The protocol could leak information contained in the queries because a query can depend on information revealed by other agents so far. Assume that all queries to agent i only depend on his own previous replies (separate preference elicitation). Then, the only information he learns from the elicitation process is that some subset of his preferences is inessential given the preferences of the other agents. However, according to the definition of privately elicitable mechanisms, this subset can be inferred from the mechanism outcome anyway. What remains to be shown is that there *always* is a protocol that hides some part of the preferences unconditionally. If we define the query function to ask completely at *random* for information that has not been revealed by that agent so far (satisfying separate preference elicitation), some preferences always remain unelicited (in expectation).

There certainly exist particular mechanisms that allow for more efficient elicitation protocols than this general proof construction. Also, in some specific mechanisms, queries may depend on others' preferences (*not* satisfying separate elicitation) if the information revealed through the queries can be inferred from the outcome. □

Together with Proposition 1 and Proposition 2 this result gives a nice classification of private mechanisms: *All mechanisms* can be emulated guaranteeing computational privacy, *privately elicitable mechanisms* can additionally provide unconditionally privacy of a (non-empty) subset of preferences, and *simple mechanisms* provide unconditional privacy of all preferences. A striking advantage of private elicitation over the approach given in Proposition 2 is that it enables unconditional privacy of some preferences *without* assuming private channels.

It is important to note that elicitation can invalidate strategy equilibria existing in the single-step version of a mechanism if the queries asked to an agent depend on other agents' preceding answers [CS02b]. When preference elicitation is used to implement a mechanism that would be a dominant-strategy direct-revelation mechanism if implemented as a single-step mechanism, then each agent's best (even in hindsight) strategy

is to act truthfully *if* the other agents act truthfully [CS01]. In other words, truthful strategies form an ex post equilibrium. Ex post equilibria are not as robust as dominant-strategy equilibria, but are more robust than Bayesian Nash equilibria in that they are prior-free.

The emulation of multi-step mechanisms is a powerful tool to guarantee strong privacy (partially unconditional) *and* reduce the amount of agents' deliberation required to evaluate their preferences. For example, consider a veto voting mechanism: Preferences are single bits (veto or not) and the outcome function is defined as $f(\theta) = \bigvee_{i=1}^{n} \theta_i$. This mechanism is not simple according to Definition 2. As a consequence, there is no unconditionally fully private veto protocol. However, we can construct a protocol in which *most* preferences remain unconditionally private by using Theorem 4. The protocol consists of n rounds. In each round, a randomly selected agent that has not been queried so far is privately asked for his preference (veto or not). All other agents receive empty queries and reply with empty queries. Once an agent privately submits a veto, *all* agents receive empty queries in the following rounds. Since the query function is computed fully privately, only some agents (those who are queried) learn some probabilistic information on the number of vetoers.

4.3 Untruthful Mechanisms

Untruthful mechanisms may not only lead to greater social welfare in some settings (relying on computational assumptions) [CS03b], but they can also support the protection of preferences. As a matter of fact, the probably most prominent truthful mechanism, the Vickrey auction, is said to be rare because bidders are reluctant to reveal their preferences truthfully as required by the dominant strategy [RTK90]. This problem and the possibility of an untruthful mechanism center (which is stated as the other major reason for the Vickrey auction's rareness) can be tackled by the techniques presented in Section 3.2. Yet, even in the centralized model, preferences can be protected (at least partially) by inducing strategic behavior in a mechanism. We sketch two different ways how to achieve this, of which the second one relies on computational intractability.

When the mapping from preferences to a strategy is a *non-injective* function, *i.e.*, different preferences can yield the same strategy, it is obviously impossible to uniquely invert a strategy. This naturally is the case when the set of strategies S_i is smaller than set of preferences ($|\Theta_i| > |S_i|$). For instance, in most voting protocols with more than two candidates, a complete ranking of candidates (the preferences) is mapped to a single strategic vote (the strategy). For example, when considering the US presidential election in 2000 (plurality voting with three candidates), it is impossible to tell if someone who voted for Gore, truthfully preferred Gore over Nader or not (even given the prior beliefs of that voter, *e.g.*, that Nader would be far behind). The same argument applies to most other voting protocols.

Based on these considerations, it might be interesting to construct mechanisms that induce equilibria consisting of "*one-way strategies*". Here, the mapping from preferences to a strategy is computationally easy while the inversion is intractable (preferably even given the beliefs that the agent had). This requires that $|\Theta_i|$ is exponentially large or somehow enriched, possibly by random data padding. We do not know whether such

mechanisms can be constructed (for relevant problems), but note that this might be another interesting application of computational intractability in mechanism design.

5 Conclusion

In this paper, we suggested that the fields of cryptography and mechanism design have several similarities and can both greatly benefit from each other. We proposed two security models for centralized mechanism execution in which the center proves its correctness (in zero-knowledge) and that provide differing degrees of privacy. However, in these models, privacy always has to rely on the trustworthiness of the mechanism center. For this reason, we showed how participating agents can emulate a (correct) "virtual" mechanism center by jointly computing the mechanism outcome without a trusted third-party. The emulation of a restricted class of mechanisms, so-called simple mechanisms, can provide unconditional privacy of preferences whereas all mechanisms can be emulated so that privacy can only be breached by unbounded computation. In these models, privacy builds upon the fact that it can be ruled out that *all* agents are forming a coalition in order to breach privacy. Furthermore, the decentralization allows for the provable correctness of *randomized* mechanisms. Table 1 summarizes the proposed models for secure mechanism execution. Even though the centralized models provide questionable privacy, there are certainly applications in which they would be favored over the decentralized models due to practical considerations (efficiency, lack of communication between agents, robustness, *etc.*).

	Center	Privacy can be breached by	Requirements
Th. 1	yes	**center** or computation	trapdoor perm., **det. mech.**
Th. 2	yes	**center**	$A \to C$-channels, one-way perm., **det. mech.**
Pr. 1	no	computation	trapdoor perm.
Pr. 2	no	—	$A \to A$-channels, one-way perm., **simple mech.**

$A \to C$-channels: private channels from each agent to the center
$A \to A$-channels: complete network of private channels between all agents

Table 1. Comparison of security models with provable correctness.

In addition to the revelation principle criticisms stated in [CS03b], we pointed out that multi-step and untruthful mechanisms can drastically improve *privacy* in a variety of social choice settings. In particular, we identified a class of mechanisms, so-called privately elicitable mechanisms, for which there are fully private protocols that emulate a preference elicitor so that a part of the preferences is never elicited and thus remains unconditionally private and does not have to be determined by the agents.

Acknowledgements

This material is based upon work supported by the Deutsche Forschungsgemeinschaft under grant BR 2312/1-1, by the National Science Foundation under grants IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678, and a Sloan Fellowship.

References

[AH00] C. Avery and T. Hendershott. Bundling and optimal auctions of multiple products. *Review of Economic Studies*, 67:483–497, 2000.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pages 1–10. ACM Press, 1988.

[Blu82] M. Blum. Coin flipping by telephone. In *Proc. of 24th IEEE Spring Computer Conference*, pages 133–137. IEEE Press, 1982.

[BOGG+88] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In *Proc. of 14th CRYPTO Conference*, volume 403 of *LNCS*, pages 37–56. Springer, 1988.

[Bra03a] F. Brandt. Fully private auctions in a constant number of rounds. In R. N. Wright, editor, *Proc. of 7th Conference on Financial Cryptography*, volume 2742 of *LNCS*, pages 223–238. Springer, 2003.

[Bra03b] F. Brandt. Social choice and preference protection - Towards fully private mechanism design. In *Proc. of 4th ACM Conference on Electronic Commerce*, pages 220–221. ACM Press, 2003.

[BS04] F. Brandt and T. Sandholm. (Im)possibility of unconditionally privacy-preserving auctions. In *Proc. of 3rd AAMAS Conference*, 2004. To appear.

[BTT89] J. Bartholdi, III, C. A. Tovey, and M. A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multi-party unconditionally secure protocols. In *Proc. of 20th STOC*, pages 11–19. ACM Press, 1988.

[CK89] B. Chor and E. Kushilevitz. A zero-one law for Boolean privacy. In *Proc. of 21st STOC*, pages 36–47. ACM Press, 1989.

[CS01] W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions. In *Proc. of 3rd ACM Conference on Electronic Commerce*, pages 256–259. ACM Press, 2001.

[CS02a] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proc. of 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 103–110, 2002.

[CS02b] V. Conitzer and T. Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Proc. of 18th AAAI Conference*, pages 392–397. AAAI Press, 2002.

[CS03a] V. Conitzer and T. Sandholm. Applications of automated mechanism design. In *Proc. of UAI workshop on Bayesian Modeling Applications*, 2003.

[CS03b] V. Conitzer and T. Sandholm. Computational criticisms of the revelation principle. In *Proc. of 5th Workshop on Agent Mediated Electronic Commerce (AMEC)*, LNCS. Springer, 2003.

[CS03c] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proc. of 18th IJCAI*, pages 781–788, 2003.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[FNR+02] J. Feigenbaum, N. Nisan, V. Ramachandran, R. Sami, and S. Shenker. Agents' privacy in distributed algorithmic mechanisms. Position Paper, 2002.

[FS02] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proc. of 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13. ACM Press, 2002.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. of 19th STOC*, pages 218–229. ACM Press, 1987.

[Gol01] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.

[HT04] J. Halpern and V. Teague. Rational secret sharing and multiparty computation. In *Proc. of 36th STOC*. ACM Press, 2004.

[Kik01] H. Kikuchi. (M+1)st-price auction protocol. In *Proc. of 5th Conference on Financial Cryptography*, volume 2339 of *LNCS*, pages 351–363. Springer, 2001.

[Kus89] E. Kushilevitz. Privacy and communication complexity. In *Proc. of 30th FOCS Symposium*, pages 416–421. IEEE Computer Society Press, 1989.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[MPS03] R. McGrew, R. Porter, and Y. Shoham. Towards a general theory of non-cooperative computation. In *Proc. of 9th International Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, 2003.

[MT99] D. Monderer and M. Tennenholtz. Distributed games: From mechanisms to protocols. In *Proc. of 15th AAAI Conference*, pages 32–37. AAAI Press, 1999.

[MV04] A. Mehta and V. Vazirani. Randomized truthful auctions of digital goods are randomizations over truthful auctions. In *Proc. of 5th ACM Conference on Electronic Commerce*, pages 120–124. ACM Press, 2004.

[Nao89] M. Naor. Bit commitment using pseudorandomness. In *Proc. of 9th CRYPTO Conference*, volume 435 of *LNCS*, pages 128–137. Springer, 1989.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. of 1st ACM Conference on Electronic Commerce*, pages 129–139. ACM Press, 1999.

[RTK90] M. H. Rothkopf, T. J. Teisberg, and E. P. Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98(1):94–109, 1990.

[San93] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of 10th AAAI Conference*, pages 256–262, 1993.

[Yao82] A. C. Yao. Protocols for secure computation. In *Proc. of 23th FOCS Symposium*, pages 160–164. IEEE Computer Society Press, 1982.