

Fair Computation with Rational Players

Adam Groce and Jonathan Katz*

Department of Computer Science, University of Maryland
{agroce, jkatz}@cs.umd.edu

Abstract. We consider the problem of *fairness* in two-party computation, where this means (informally) that both parties should learn the correct output. A seminal result of Cleve (STOC 1986) shows that fairness is, in general, impossible to achieve for *malicious* parties. Here, we treat the parties as *rational* and seek to understand what can be done.

Asharov et al. (Eurocrypt 2011) recently considered this problem and showed impossibility of rational fair computation for a particular function and a particular set of utilities. We observe, however, that in their setting the parties have no incentive to compute the function *even in an ideal world where fairness is guaranteed*. Revisiting the problem, we show that rational fair computation *is* possible, for arbitrary functions and utilities, as long as at least one of the parties has a strict incentive to compute the function in the ideal world. This gives a novel setting in which game-theoretic considerations can be used to circumvent an impossibility result in cryptography.

1 Introduction

Cryptography and game theory are both concerned with understanding interactions between mutually distrusting parties with potentially conflicting interests. Cryptography typically adopts a “worst case” viewpoint; that is, cryptographic protocols are designed to protect the interests of each party against *arbitrary* (i.e., malicious) behavior of the other parties. The game-theoretic perspective, however, views parties as being *rational*; game-theoretic protocols, therefore, only need to protect against rational deviations by other parties.

Significant effort has recently been devoted to bridging cryptography and game theory; see [9,21] for surveys. This work has tended to focus on two general sets of questions:

“Using cryptographic protocols to implement games” (e.g., [7,10,4,8,25,1,19]). Given a game played by parties relying an external trusted entity (a *mediator*), when can the external, trusted party be replaced by a cryptographic protocol executed by the parties themselves?

“Applying game-theoretic analysis to cryptographic protocols” (e.g., [17,21,13,26,1,15,16]). What game-theoretic definitions are appropriate for computationally bounded, rational players executing a network protocol? Can

* Research supported by NSF awards #0830464 and #1111599.

impossibility results in the cryptographic setting be circumvented if we are willing to take a game-theoretic approach?

Here, we turn our attention to the question of *fair two-party computation* in a rational setting, where fairness means that both parties should learn the value of some function f evaluated on the two parties' inputs. Following recent work of Asharov et al. [2] (see further below), our goal is to understand when fairness is achievable by rational parties running some cryptographic protocol, without the aid of any external trusted entity. Our work touches on both the issues outlined above. Our motivation was to circumvent the strong impossibility result of Cleve [6] for fair two-party computation in a malicious setting. In this sense, our work is a generalization of results on *rational secret sharing* [17,13,26,1,22,23,29,27,3,11], which can be seen as a special case of fair computation for a specific function with parties' inputs provided by a trusted dealer. It is also possible to view the problem of rational fair computation from a different perspective. Specifically, one could define a natural "fairness game" involving a trusted mediator who computes a function f on behalf of the parties (and gives both parties the result), and where parties can choose whether or not to participate and, if so, what input to send to the mediator. One can then ask whether there exists a real-world protocol (replacing the mediator) that preserves equilibria of the original mediated game. Our work demonstrates a close connection between these two complementary viewpoints; see further below.

1.1 Our Results

Our setting is the same as that studied by Asharov, Canetti, and Hazay [2]. Informally, there are two parties P_0 and P_1 who wish to compute a function f of their respective inputs x_0 and x_1 , where the joint distribution of x_0 and x_1 is common knowledge. (In [2] independent uniform distributions were assumed but we consider arbitrary joint distributions.) Following work on rational secret sharing, the parties' utilities are such that each party prefers to learn the correct answer $f(x_0, x_1)$ and otherwise prefers that the other party outputs an *incorrect* answer. Informally, a cryptographic protocol computing f is *fair* if having both parties run the protocol is a (computational) Nash equilibrium with respect to fail-stop deviations,¹ i.e., it is assumed that parties may abort the protocol early, but cannot otherwise deviate.

Asharov et al. show a strong negative result in this context: they give a function f , a pair of distributions on the inputs, and a set of utilities for which they prove there is *no* fair protocol computing f with correctness better than $1/2$. They also show that correctness $1/2$ *can* be achieved for that function and utilities, but their work seemed to suggest that the power of rational fair computation is relatively limited.

Looking more closely at the impossibility result of Asharov et al., we observe that for their specific choices of f , the input distributions, and the utility

¹ We will consider Byzantine deviations as well, but stick to fail-stop deviations here for simplicity.

functions, *the parties have no incentive to run a protocol at all!* Namely, the utility each party obtains by running *any* protocol that correctly (and fairly) computes f is equal to the expected utility that each party obtains if it simply guesses the input of the other party and computes the function on its own (without any interaction).² A cleaner way of stating this is that *even in an ideal world* with a trusted entity computing f with complete fairness, the parties would be indifferent between using the trusted entity or not. In game-theoretic terms, computing f in this ideal world is not a *strict* Nash equilibrium for the specific setting considered in [2]. If running a (real-world) protocol incurs any cost at all, there is thus little hope that parties will prefer to run *any* protocol for computing f .

Asharov et al. rule out rational fair computation for a *specific* function, *specific* input distributions, and a *specific* set of utilities. Are there any settings where rational fair computation (with complete correctness) *is* possible? Assuming the existence of (standard) secure computation, we show a strong, general result for when this is the case:

Main Theorem. (Informal) *Fix f , a distribution on inputs, and utility functions such that computing f in the ideal world (with complete fairness) is a **strict** Nash equilibrium for at least one party. Then, for the same input distributions and utility functions, there is a protocol Π for computing f (where correctness holds with all but negligible probability) such that following Π is a computational Nash equilibrium. This holds in both the fail-stop and Byzantine settings.*

In addition to the fact that we show a positive result, our work goes beyond the setting considered in [2] in several respects: we handle (deterministic) functions over arbitrary domains where parties receive possibly different outputs, and treat arbitrary distributions over the parties' inputs. (In [2], only single-output functions and independent, uniform input distributions were considered.) Moreover, we also treat the *Byzantine* setting where, in particular, parties have the option of changing their inputs; Asharov et al. [2] only treat the fail-stop case.

1.2 Other Related Work

The most relevant prior work is that of Asharov et al. [2], already discussed extensively above. Here we merely add that an additional contribution of their work was to develop formal definitions of various cryptographic goals (with fairness being only one of these) in a game-theoretic context. Their paper takes an important step toward that worthy goal.

As observed earlier, work on rational secret sharing [17,13,26,1,22,23,29,27,3,11] can be viewed as a special case of fair secure computation, where the function

² Specifically, using the input distributions and utility functions from [2], P_0 's utility if both parties (run some protocol and) output the correct answer is 0, whereas if both parties guess then each party is (independently) correct with probability $1/2$ and so the expected utility of P_0 is $\frac{1}{4} \cdot 1 + \frac{1}{4} \cdot (-1) + \frac{1}{2} \cdot 0 = 0$. A similar calculation holds for P_1 .

being computed is the reconstruction function of the secret-sharing scheme being used, and the parties' inputs are generated by a dealer. Depending on the utilities and input distributions, our results would give rational secret-sharing protocols where following the protocol is a (computational) Nash equilibrium. In contrast, most of the work on rational secret sharing has focused on achieving stronger equilibrium notions, in part because constructing Nash protocols for rational secret sharing is trivial in the multi-party setting. (We stress that in the two-party setting we consider here, constructing Nash protocols for rational secret sharing is *not* trivial.) We leave for future work consideration of stronger equilibrium notions for rational fair computation of general functions.

An analogue of our results is given by Izmalkov et al. [24,25,20,18] who, essentially, also show protocols for rational fair computation whenever parties would prefer to compute the function in the ideal world. The main difference is that we work in the cryptographic setting where parties communicate using standard channels, whereas the protocols of Izmalkov et al. require strong *physical* assumptions such as secure envelopes and ballot boxes.

There has recently been a significant amount of work on fairness in the cryptographic setting, showing functions that can be computed with complete fairness [12] and exploring various notions of partial fairness (see [14] and references therein). The relationship between complete fairness and rational fairness is not clear. In particular, completely fair protocols are not necessarily rationally fair: if the distributions and utilities are such that aborting is preferable even in the ideal world then no protocol can be rationally fair (with respect to the same distributions and utilities). In any case, relatively few functions are known that can be computed with complete fairness. Partial fairness is similarly incomparable to rational fairness.

2 Model and Definitions

Given a deterministic function $f : X \times Y \rightarrow \{0, 1\}^* \times \{0, 1\}^*$, we let f_0 (resp., f_1) denote the first (resp., second) output of f , so that $f(x, y) = (f_0(x, y), f_1(x, y))$. We consider two settings where parties P_0 and P_1 wish to compute f on their respective inputs x_0 and x_1 , with P_0 receiving $f_0(x_0, x_1)$ and P_1 receiving $f_1(x_0, x_1)$: an *ideal world* computation of f using a trusted third party, and a *real-world* computation of f using some protocol Π . In each setting, x_0 and x_1 are chosen according to some joint probability distribution D , and in each setting we consider both fail-stop and Byzantine strategies.

The output of P_0 is correct if it is equal to $f_0(x_0, x_1)$ and incorrect otherwise; this is defined analogously for P_1 . The parties' utilities are given by the following table, where the first value in each ordered pair is the utility of P_0 on the specified outcome, and the second is the utility of P_1 :

		P ₁ 's output	
		correct	incorrect
P ₀ 's output	correct	(a ₀ , a ₁)	(b ₀ , c ₁)
	incorrect	(c ₀ , b ₁)	(d ₀ , d ₁)

We assume that parties prefer to output the correct answer, and otherwise prefer that the other party outputs an incorrect answer; i.e., we assume $b_0 > a_0 \geq d_0 \geq c_0$ (and analogously for P_1 's utilities). Asharov et al. [2] assume that the parties' utilities are symmetric, with $b_0 = b_1 = 1, a_0 = a_1 = d_0 = d_1 = 0$, and $c_0 = c_1 = -1$; we consider more general utilities here.

2.1 Execution in the Ideal World

Our ideal world includes a trusted third party who computes f with complete fairness. This defines a natural game that proceeds as follows:

1. Inputs x_0 and x_1 are sampled according to a joint probability distribution D over input pairs. Then x_0 (resp., x_1) is given to P_0 (resp., P_1).
2. Each player sends a value to the trusted third party. We also allow parties to send a special value \perp denoting an abort. Let x'_0 (resp., x'_1) denote the value sent by P_0 (resp., P_1).
3. If $x'_0 = \perp$ or $x'_1 = \perp$, the trusted party sends \perp to both parties. Otherwise, the trusted party sends $f_0(x'_0, x'_1)$ to P_0 , and $f_1(x'_0, x'_1)$ to P_1 .
4. Each party outputs some value, and obtains a utility that depends on whether the parties' outputs are correct or not. (We stress that correctness is defined with respect to the "real" inputs x_0, x_1 , not the effective inputs x'_0, x'_1 .)

In the *fail-stop* setting, we restrict $x'_0 \in \{x_0, \perp\}$ and $x'_1 \in \{x_1, \perp\}$. In the *Byzantine* setting we allow x'_0, x'_1 to be arbitrary.

The "desired" play in this game is for each party to send its input to the trusted third party, and then output the value returned by the trusted party. To fully define an honest strategy, however, we must specify what each party does for every possible value (including \perp) it receives from the trusted third party. We formally define strategy (**cooperate**, W_0) for P_0 as follows:

P_0 sends its input x_0 to the trusted party. If the trusted party returns anything other than \perp , then P_0 outputs that value. If instead \perp is returned, then P_0 generates output according to the distribution $W_0(x_0)$.

The strategy (**cooperate**, W_1) for P_1 is defined analogously. The situation in which P_0 plays (**cooperate**, W_0) and P_1 plays (**cooperate**, W_1) is a (*Bayesian*) *strict Nash equilibrium* if every (allowed³) deviation that has $x'_0 \neq x_0$ with nonzero probability results in a strictly lower expected utility for P_0 , and analogously for P_1 . (The expectation is computed over the distribution of the other party's input.) Since the utility obtained by P_0 when the honest strategies are followed is exactly a_0 , this means that the honest strategies form a Bayesian strict Nash equilibrium if, for every possible input x_0 of P_0 , the expected utility of P_0 is strictly less than a_0 if it sends any (allowed) $x'_0 \neq x_0$ to the third party, and similarly for P_1 .

³ I.e., in the fail-stop case the only allowed deviation is aborting, whereas in the Byzantine case parties are allowed to send arbitrary inputs. In either case a deviating party may determine its output any way it likes.

As long as both parties follow honest strategies, W_0 and W_1 are irrelevant (as they are never used). They are important, however, insofar as they serve as “empty threats” in case of an abort by the other party: namely, P_0 knows that if he aborts then P_1 will determine its own output according to $W_1(x_1)$, and so P_0 must take this into account when deciding whether to abort or not. We now define what it means for the parties to have an incentive to compute f .

Definition 1. Fix f , a distribution D , and utilities for the parties. We say these are incentive compatible in the fail-stop (resp., Byzantine) setting if there exist W_0, W_1 such that the strategy profile $\left((\text{cooperate}, W_0), (\text{cooperate}, W_1) \right)$ is a Bayesian strict Nash equilibrium in the game above.

As discussed in the Introduction, we require the Nash equilibrium to be *strict* to ensure that the parties have *some* incentive to use the trusted party to compute the function. If carrying out the computation with the trusted party is a Nash (but not strict Nash) equilibrium, then the parties are indifferent between using the trusted party and just guessing the output on their own.

We remark that for our positive results it is sufficient for the ideal-world equilibrium to be strict Nash for only *one* of the parties; this follows directly from the proofs and we omit further discussion.

The Setting of Asharov et al. [2]. For completeness, we show that the setting considered by Asharov et al. is *not* incentive compatible. Recall that they fix the utilities such that (1) getting the correct answer while the other party outputs an incorrect answer gives utility 1; (2) getting an incorrect answer while the other party outputs the correct answer gives utility -1 ; and (3) any other outcome gives utility 0. Furthermore (cf. [2, Definition 4.6]), f can be taken to be boolean XOR, with inputs for each party chosen uniformly and independently. We claim that there is no choice of W_0, W_1 for which $\left((\text{cooperate}, W_0), (\text{cooperate}, W_1) \right)$ is a Bayesian strict Nash equilibrium. To see this, fix W_0, W_1 and note that playing $\left((\text{cooperate}, W_0), (\text{cooperate}, W_1) \right)$ gives utility 0 to both parties. On the other hand, if P_0 aborts and outputs a random bit, then regardless of the guessing strategy W_1 employed by P_1 , we see that P_0 and P_1 are each correct with independent probability $1/2$ and so the expected utility of P_0 remains 0. This is an allowed deviation that results in no change to the expected utility.

In contrast, if the utilities are modified so that when both parties get the correct answer they each obtain utility $1/2$ (and everything else is unchanged), the setup *is* incentive compatible in the fail-stop setting. To see this, let W_0, W_1 be the uniform distribution. Playing $\left((\text{cooperate}, W_0), (\text{cooperate}, W_1) \right)$ gives utility $1/2$ to both parties. If, on the other hand, P_0 ever aborts then — no matter how P_0 determines its output — P_0 and P_1 are each correct with independent probability $1/2$. (Recall that P_1 is assumed to guess according to W_1 when we consider possible deviations by P_0 .) The expected utility of deviating is $1/8$, which is strictly smaller than $1/2$; thus, we have a Bayesian *strict* Nash equilibrium. Our results imply that a rational fair protocol *can* be constructed for this setting.

Incentive Compatibility in the Byzantine Setting. The preceding discussion gives an example of an incentive-compatible setup in the fail-stop setting. In Appendix A we show an example of a simple function, a distribution on inputs, and utilities that are incentive compatible in the Byzantine setting.

2.2 Execution in the Real World

In the real world there is no trusted party, and the players instead must communicate in order to compute f . We thus have a real-world game in which inputs x_0 and x_1 are jointly sampled according to D , then x_0 (resp., x_1) is given to P_0 (resp., P_1), and the parties finally execute some strategy (i.e., protocol) and decide on their respective outputs. The goal is to construct a protocol Π such that running the protocol is a (computational) Nash equilibrium. The running times of the parties, as well as the protocol itself, are parameterized in terms of a security parameter n ; however, the function f as well as the parties' utilities are fixed and independent of n . We only consider protocols where correctness holds with all but negligible probability.

We again consider two types of deviations. In the *fail-stop* setting, each party follows the protocol as directed except that it may choose to abort at any point. Upon aborting, a party may output whatever value it likes (and not necessarily the value prescribed by the protocol). We stress that in the fail-stop setting a party is assumed not to change its input when running the protocol. In the *Byzantine* setting, parties may behave arbitrarily (and, in particular, may run the protocol using a different input). In either setting, we will only be interested in players whose strategies can be implemented in probabilistic polynomial-time.

We now define what it means for Π to induce a game-theoretic equilibrium. We consider computational Nash equilibria, rather than computational *strict* Nash equilibria, since the latter are notoriously difficult to define [11]; also, the goal of our work is only to construct real-world protocols that induce a Nash equilibrium. (We define strict Nash equilibria in the ideal world because we use it for our results.) The following definition is equivalent to (a generalized version of) the definition used by Asharov et al. [2, Definition 4.6].

Definition 2. Fix f , a distribution D , utilities for the parties, and a protocol Π computing f . We say Π is a rational fair protocol (with respect to these parameters) in the fail-stop (resp., Byzantine) setting if running the protocol is a Bayesian computational Nash equilibrium in the game defined above.

For example, if we let Π_1 denote the algorithm that honestly implements P_1 's role in Π , then Π is a rational fair protocol in the fail-stop setting if for all (efficient) fail-stop algorithms P'_0 there is a negligible function μ such that the expected utility of $P'_0(1^n)$ (when running against P_1 playing $\Pi_1(1^n)$) is at most $a_0 + \mu(n)$ (and analogously for deviations by P_1). We stress that if P'_0 aborts here, then P_1 determines its output as directed by Π_1 .

Note that it only makes sense to speak of Π being a rational fair protocol with regard to some input distribution and utilities. In particular, it is possible for Π to be rational for one set of utilities but not another.

3 Positive Results for Rational Fair Computation

We show broad positive results for rational fair computation in both the fail-stop and Byzantine settings. Specifically, we show that whenever computing the function honestly is a Bayesian strict Nash equilibrium in the ideal world, then there exists a protocol Π computing f such that running Π is a Bayesian computational Nash equilibrium in the real world.

Our protocols all share a common structure. As in prior work on fairness (in the cryptographic setting) [12,28,14], our protocols have two stages. The first stage is a “pre-processing” step that uses any protocol for (standard) secure two-party computation, and the second stage takes place in a sequence of n iterations. In our work, the stages have the following form:

First stage:

1. A value $i^* \in \{1, \dots\}$ is chosen according to a geometric distribution. This represents the iteration (unknown to the parties) in which both parties will learn the correct output.
2. Values $r_1^0, r_1^1, \dots, r_n^0, r_n^1$ are chosen, with the $\{r_i^0\}_{i=1}^n$ intended for P_0 and the $\{r_i^1\}_{i=1}^n$ intended for P_1 . For $i \geq i^*$ we have $r_i^0 = f_0(x_0, x_1)$ and $r_i^1 = f_1(x_0, x_1)$, while for $i < i^*$ the $\{r_i^0\}$ (resp., $\{r_i^1\}$) values depend on P_0 's (resp., P_1 's) input only.
3. Each r_i^b value is randomly shared as s_i^b and t_i^b (with $r_i^b = s_i^b \oplus t_i^b$), where s_i^b is given to P_0 and t_i^b is given to P_1 .

Second stage: For n iterations, each consisting of two rounds, the parties alternate sending shares to each other. In the i th iteration, P_1 sends t_i^0 to P_0 , enabling P_0 to learn r_i^0 ; then P_0 sends s_i^1 to P_1 , enabling P_1 to learn r_i^1 . When the protocol ends (either through successful termination or an abort by the other party) a party outputs the most-recently-learned r_i .

The key difference with respect to prior work is how we set the distribution of the $\{r_i^b\}$ for $i < i^*$. Here we use the assumption that f, D , and the utilities are incentive compatible, and thus there are “guessing strategies” $W_0(x_0)$ and $W_1(x_1)$ for the parties (in case the other party aborts) that are in equilibrium (see Section 2.1). We use exactly these distributions in our protocol.

3.1 The Fail-Stop Setting

We first analyze the fail-stop setting. We let W_0 (resp., W_1) denote the distribution that P_0 (resp., P_1) uses to determine its output in the ideal world in case P_1 (resp., P_0) aborts, where this distribution may depend on P_0 's input x_0 (resp., P_1 's input x_1). We say W_0 has *full support* if for every x_0 the distribution $W_0(x_0)$ puts non-zero probability on every element in the range of f ; we define this notion analogously for W_1 . We begin with a technical claim.

Lemma 1. *Fix a function f , a distribution D , and utilities for the parties that are incentive compatible in the fail-stop (resp., Byzantine) setting. Then there*

exist W_0, W_1 **with full support** such that $((\text{cooperate}, W_0), (\text{cooperate}, W_1))$ is a Bayesian strict Nash equilibrium in the fail-stop (resp., Byzantine) setting.

Proof. We focus on the fail-stop setting, though the proof follows along the same lines for the Byzantine case. Incentive compatibility implies that there exist W'_0, W'_1 such that the strategy vector $((\text{cooperate}, W'_0), (\text{cooperate}, W'_1))$ is a Bayesian strict Nash equilibrium. Distributions W'_0, W'_1 may not have full support, but we show that they can be modified so that they do. Specifically, we simply modify each distribution so that with some sufficiently small probability it outputs a uniform element from the range of f . Details follow.

When P_0 and P_1 cooperate and both output the correct answer, P_0 obtains utility a_0 . Consider now some input x_0 for P_0 , and let $u_0^*(x_0)$ denote the maximum utility P_0 can obtain if it aborts on input x_0 . (Recall that when P_0 aborts, P_1 chooses its output according to $W'_1(x_1)$. Here, P_0 knows W'_1 as well as the marginal distribution of x_1 conditioned on P_0 's input x_0 .) Because cooperating is a Bayesian strict Nash equilibrium, we must have $u_0^*(x_0) < a_0$. Define

$$u_0^* \stackrel{\text{def}}{=} \max_x \{u_0^*(x)\} < a_0$$

(the maximum is taken over all x that have non-zero probability as input to P_0); i.e., u_0^* denotes the highest expected utility P_0 can hope to obtain when aborting on some input. Define u_1^* analogously with respect to deviations by P_1 . Set

$$\lambda \stackrel{\text{def}}{=} \frac{1}{2} \cdot \min \left\{ \frac{a_0 - u_0^*}{b_0 - c_0}, \frac{a_1 - u_1^*}{b_1 - c_1} \right\} > 0.$$

We define a distribution $W_0(x_0)$ as follows: with probability λ output a uniform element from the range of f , and with probability $(1 - \lambda)$ choose an output according to $W'_0(x_0)$; define W_1 similarly. Note that W_0 and W_1 have full support. We claim that $((\text{cooperate}, W_0), (\text{cooperate}, W_1))$ is a Bayesian strict Nash equilibrium. Assume the contrary; then, without loss of generality, there is an input x_0 such that P_0 obtains expected utility at least a_0 by aborting on input x_0 . (Note that now P_1 chooses its output according to $W_1(x_1)$ when P_0 aborts.) But then, by following the same strategy, P_0 can obtain utility at least $a_0 - \lambda \cdot (b_0 - c_0)$ when playing against a P_1 who chooses his output according to $W'_1(x_1)$. Since $a_0 - \lambda \cdot (b_0 - c_0) > u_0^*$, this is a contradiction to the way u_0^* was defined. \square

Theorem 1. *Fix a function f , a distribution D , and utilities for the parties. If these are incentive compatible in the fail-stop setting, then (assuming the existence of general secure two-party computation for semi-honest adversaries) there exists a protocol Π computing f such that Π is a rational fair protocol (with respect to the same distribution and utilities) in the fail-stop setting.*

Proof. By definition of incentive compatibility, there exist distributions W_0, W_1 (that can depend on x_0 and x_1 , respectively) for which the strategy profile

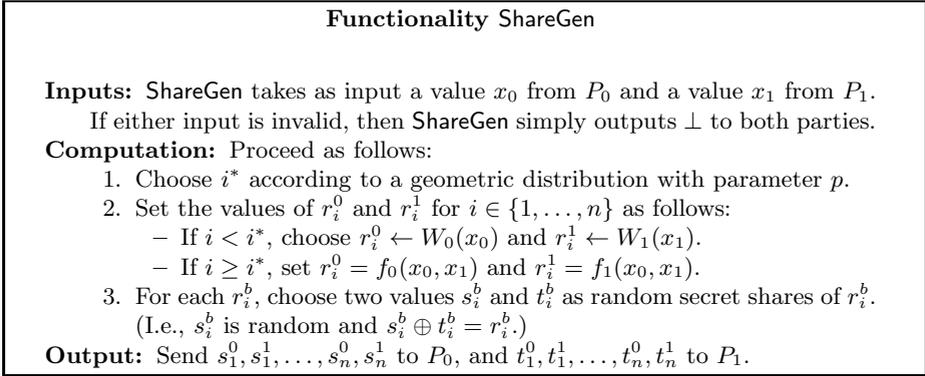


Fig. 1. Functionality ShareGen. The security parameter is n . This functionality is parameterized by a real number $p > 0$.

$((\text{cooperate}, W_0), (\text{cooperate}, W_1))$ is a Bayesian strict Nash equilibrium. By Lemma 1, we may assume that W_0 and W_1 both have full support. We define a functionality ShareGen (cf. Figure 1) based on these distributions; this functionality is parameterized by a real number $p > 0$ that we will set later. We define our protocol Π , that uses ShareGen as a building block, in Figure 2.

Since p is a constant (independent of n), we have $i^* \leq n$ with all but negligible probability and hence both parties obtain the correct answer with all but negligible probability. In the analysis it is easiest to simply assume that $i^* \leq n$; this does not affect the analysis since computational Nash equilibria are robust to negligible changes in the utility. Alternately, one could simply modify ShareGen to enforce that $i^* \leq n$ always (namely, by setting $i^* = n$ in case $i^* > n$).

We will analyze Π in a hybrid world where there is a trusted entity computing ShareGen. One can show (following [5]) that if Π is a computational Nash equilibrium in this hybrid world, then so is Π when executed in the real world

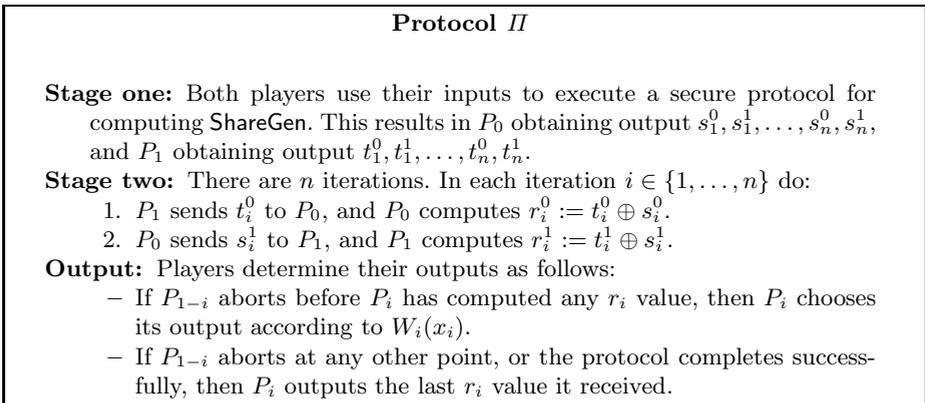


Fig. 2. Formal definition of our protocol

(with a secure protocol implementing ShareGen). Once we have moved to this hybrid world, we may in fact take the parties to be computationally unbounded.

Our goal is to show that there exists a $p > 0$ for which Π (in the hybrid world described above) is a rational fair protocol. We first observe that there are no profitable deviations for a fail-stop P_1 ; this is because P_0 always “gets the output first” in every iteration. (More formally, say P_1 aborts after receiving its iteration- i message. If $i \geq i^*$ then P_0 will output the correct answer and so P_1 cannot possibly get utility greater than a_1 . If $i < i^*$ then P_0 has no information beyond what it could compute from its input x_1 , and P_0 will generate output according to $W_0(x_0)$; by incentive compatibility, P_1 will obtain utility strictly lower than a_1 regardless of how it determines its output.) We are thus left with the more difficult task of analyzing deviations by P_0 .

Before continuing, it is helpful to introduce two modifications to the protocol that can only increase P_0 's utility. First, in each iteration i we tell P_0 whether $i^* < i$. (One can easily see that P_0 cannot increase its utility by aborting when $i^* < i$, and so the interesting question is whether P_0 can improve its utility by aborting when $i^* \geq i$.) Second, if P_0 ever aborts the protocol in some iteration i with $i^* \geq i$, then we tell P_0 whether $i^* = i$ before P_0 generates its output. (P_0 is not, however, allowed to change its decision to abort.)

So, fix some input x_0 for P_0 , and consider some iteration $i < n$. Say P_0 has just learned that $r_i = y$ (for some y in the range of f) and is told that $i^* \geq i$. If P_0 does not abort, but instead runs the protocol honestly to the end, then it obtains utility a_0 . If P_0 aborts, then with some probability α it learns that $i^* = i$; in that case, P_0 may possibly get utility b_0 . Otherwise, with probability $1 - \alpha$ it learns that $i^* > i$. In this latter case, P_0 has no information beyond what it could compute from its input, and P_1 will output a value distributed according to $W_1(x_1)$; hence, incentive compatibility implies that the maximum expected utility of P_0 is $u_0^* < a_0$. (This u_0^* is the same as defined in the proof of Lemma 1; for our purposes all we need is that u_0^* is strictly less than a_0 .) That is, the expected utility of aborting is at most $\alpha \cdot b_0 + (1 - \alpha) \cdot u_0^*$. If

$$\alpha < \frac{a_0 - u_0^*}{b_0 - u_0^*} \quad (1)$$

then $\alpha \cdot b_0 + (1 - \alpha) \cdot u_0^* < a_0$, implying that P_0 has no incentive to deviate. We show that p can be set such that (1) holds.

We have

$$\begin{aligned} \alpha &\stackrel{\text{def}}{=} \Pr[i^* = i \mid r_i = y \wedge i^* \geq i] = \frac{\Pr[i^* = i \wedge r_i = y \mid i^* \geq i]}{\Pr[r_i = y \mid i^* \geq i]} \\ &= \frac{\Pr[i^* = i \mid i^* \geq i] \cdot \Pr[r_i = y \mid i^* = i]}{\Pr[i^* = i \mid i^* \geq i] \cdot \Pr[r_i = y \mid i^* = i] + \Pr[i^* > i \mid i^* \geq i] \cdot \Pr[r_i = y \mid i^* > i]} \\ &= \frac{p \cdot \Pr[r_i = y \mid i^* = i]}{p \cdot \Pr[r_i = y \mid i^* = i] + (1 - p) \cdot \Pr[r_i = y \mid i^* > i]}. \end{aligned}$$

Let $q \stackrel{\text{def}}{=} \min_{x_0, y} \{ \Pr[W_0(x_0) = y] \}$, where the minimum is taken over all inputs x_0 for P_0 and all y in the range of f . Since W_0 has full support, we have $q > 0$. Thus,

$$\begin{aligned} \alpha &= \frac{p \cdot \Pr[r_i = y \mid i^* = i]}{p \cdot \Pr[r_i = y \mid i^* = i] + (1 - p) \cdot \Pr[r_i = y \mid i^* > i]} \\ &\leq \frac{p}{p + (1 - p) \cdot q} \\ &= \frac{p}{p \cdot (1 - q) + q} \leq \frac{p}{q}, \end{aligned}$$

and by setting $p < q \cdot (a_0 - u_0^*) / (b_0 - u_0^*)$ we ensure that (1) holds.

Assuming p is set as just discussed, the above analysis shows that in any iteration $i < n$ and for any value $r_i = y$ received by P_0 in that iteration, P_0 has no incentive to abort. (In fact, P_0 has strict incentive *not* to abort.) The only remaining case to analyze is when $i = n$. In this case it would indeed be advantageous for P_0 to abort when $i^* \geq i$; however, this occurs with only negligible probability and so does not impact the fact that we have a computational Nash equilibrium (which is insensitive to negligible changes in the utility). \square

Although security notions other than fairness are not the focus of our work, we note that the protocol Π presented in the proof of the previous theorem is private in addition to being rationally fair. That is, the parties learn the function output only, but nothing else regarding the other party's input. We omit formal definitions and the straightforward proof.

3.2 The Byzantine Setting

We next consider the Byzantine setting, where in the ideal world a deviating party can change the input it sends to the trusted third party (or may choose to abort, as before), and in the real world a deviating party may behave arbitrarily.

The protocol and proof of fairness in the Byzantine setting are similar to those of the fail-stop setting, however we must modify our protocol to ensure that it will work in the Byzantine setting. In particular, we require **ShareGen** to now apply a message-authentication code (MAC) to each s_i^b and t_i^b value so that parties can detect if these values have been modified. The remaining issue to deal with is the effect of changing inputs; however, we show that if incentive compatibility holds — so parties have disincentive to change their inputs in the ideal world — then parties have no incentive to change their inputs in the real world either.

Theorem 2. *Fix a function f , a distribution D , and utilities for the parties. If these are incentive compatible in the Byzantine setting, then (assuming the existence of general secure two-party computation for malicious adversaries) there exists a protocol Π computing f such that Π is a rational fair protocol (with respect to the same distribution and utilities) in the Byzantine setting.*

Functionality ShareGen

Inputs: ShareGen takes as input a value x_0 from P_0 and a value x_1 from P_1 .

If either input is invalid, then ShareGen simply outputs \perp to both parties.

Computation: Proceed as follows:

1. Choose i^* according to a geometric distribution with parameter p .
2. Choose MAC keys $k^0, k^1 \leftarrow \{0, 1\}^n$.
3. Set the values of r_i^0 and r_i^1 for $i \in \{1, \dots, n\}$ as follows:
 - If $i < i^*$, choose $r_i^0 \leftarrow W_0(x_0)$ and $r_i^1 \leftarrow W_1(x_1)$.
 - If $i \geq i^*$, set $r_i^0 = f_0(x_0, x_1)$ and $r_i^1 = f_1(x_0, x_1)$.
4. For each r_i^b , choose two values s_i^b and t_i^b as random secret shares of r_i^b . (I.e., s_i^b is random and $s_i^b \oplus t_i^b = r_i^b$.)
5. For $i = 1, \dots, n$, set $\text{tag}_i^1 \leftarrow \text{Mac}_{k^1}(i \| s_i^1)$ and $\text{tag}_i^0 \leftarrow \text{Mac}_{k^0}(i \| t_i^0)$.

Output: Send $k^0, s_1^0, s_1^1, \text{tag}_1^1, \dots, s_n^0, s_n^1, \text{tag}_n^1$ to P_0 , and $k^1, t_1^0, \text{tag}_1^0, t_1^1, \dots, t_n^0, \text{tag}_n^0, t_n^1$ to P_1 .

Fig. 3. Functionality ShareGen. The security parameter is n . This functionality is parameterized by a real number $p > 0$.

Proof. By definition of incentive compatibility, there exist distributions W_0, W_1 (that can depend on x_0 and x_1 , respectively) for which the strategy profile $((\text{cooperate}, W_0), (\text{cooperate}, W_1))$ is a Bayesian strict Nash equilibrium. By Lemma 1, we may assume that W_0 and W_1 both have full support. We define a protocol Π based on a functionality ShareGen (cf. Figures 3 and 4), where the latter is parameterized by a real number $p > 0$. These are largely identical to the protocols used in the proof of Theorem 1, with the exception that the secret shares exchanged by the parties are authenticated by a message-authentication code (MAC) as part of the computation of ShareGen, and the resulting tags are verified by the parties (as part of Π). For our proof, we assume the MAC being used is an information-theoretically secure, n -time MAC; a computationally secure MAC would also be fine.

Since p is a constant (independent of n), it is again easy to check that correctness holds with all but negligible probability. As in the proof of Theorem 1, in our analysis we assume that $i^* \leq n$ always and this does not affect our results.

The proof that Π is rationally fair in the Byzantine setting is similar to the proof of Theorem 1, and we assume familiarity with that proof here. Once again, we analyze Π in a hybrid world where there is a trusted entity computing ShareGen on behalf of the parties. We also ignore the possibility of a MAC forgery, and treat a party who sends a different share/tag from the one it received from ShareGen as if that party had simply aborted. This is justified by the fact that a successful forgery occurs with only negligible probability.

Our goal, as in the proof of Theorem 1, is to show that there exists a $p > 0$ for which Π (in the hybrid world described above, and ignoring the possibility of a MAC forgery) is a rational fair protocol. As in the preceding proof, there are no profitable deviations for P_1 . Note that here, P_1 may either abort early or change its input to ShareGen. The former does not help because P_0 always “gets the

Protocol II

Stage one: Both players use their inputs to execute a secure protocol for computing **ShareGen**. This results in P_0 obtaining output $k^0, s_1^0, s_1^1, \text{tag}_1^1, \dots, s_n^0, s_n^1, \text{tag}_n^1$, and P_1 obtaining output $k^1, t_1^0, \text{tag}_1^0, t_1^1, \dots, t_n^0, \text{tag}_n^0, t_n^1$.

Stage two: There are n iterations. In each iteration $i \in \{1, \dots, n\}$ do:

1. P_1 sends t_i^0 and tag_i^0 to P_0 . If $\text{Vrfy}_{k^0}(i || t_i^0, \text{tag}_i^0) = 1$, then P_0 computes $r_i^0 := t_i^0 \oplus s_i^0$. Otherwise, this is treated as if P_1 had aborted.
2. P_0 sends s_i^1 and tag_i^1 to P_1 . If $\text{Vrfy}_{k^1}(i || s_i^1, \text{tag}_i^1) = 1$, then P_1 computes $r_i^1 := t_i^1 \oplus s_i^1$. Otherwise, this is treated as if P_0 had aborted.

Output: Players determine their outputs as follows:

- If P_{1-i} aborts before P_i has computed any r_i value, then P_i chooses its output according to $W_i(x_i)$.
- If P_{1-i} aborts at any other point, or the protocol completes successfully, then P_i outputs the last r_i value it received.

Fig. 4. Formal definition of our protocol

output first” in every iteration; incentive compatibility in the Byzantine setting implies that the latter — whether in combination with aborting early or not — cannot help, either.

We are thus left with analyzing deviations by P_0 . We again introduce two modifications that can only increase P_0 ’s utility. First, in each iteration i we tell P_0 whether $i^* < i$. It follows immediately from incentive compatibility that P_0 cannot increase its utility by aborting when $i^* < i$ (regardless of what input it sends to **ShareGen**), and so we assume that P_0 never does so. Second, if P_0 ever aborts the protocol in some iteration i , then we tell P_0 whether $i^* = i$ before P_0 generates its output. (P_0 may not, however, change its decision to abort.)

There are two cases to analyze: either P_0 sends its actual input x_0 to **ShareGen**, or P_0 sends a different input $x'_0 \neq \perp$. In the former case, the analysis is exactly the same as in the proof of Theorem 1, and one can show that p can be set such that P_0 has no incentive to abort the protocol at any point. It remains to show that, in the second case, P_0 cannot increase its expected utility beyond a_0 , the utility it obtains by running the protocol honestly using its actual input x_0 .

If P_0 substitutes its input and then runs the protocol to the end, then (by incentive compatibility) P_0 ’s expected utility is strictly less than a_0 . Can P_0 do better by aborting early? Fix some input x_0 for P_0 , let $x'_0 \neq \perp$ be the input that P_0 sends to the trusted entity computing **ShareGen**, and consider some iteration $i < n$. (The case of $i = n$ is handled as in the proof of Theorem 1.) Say P_0 has just learned that $r_i = y$ (for some y in the range of f) and is told that $i^* \geq i$. If P_0 aborts, then with some probability α it learns that $i^* = i$; in that case, P_0 may possibly get utility b_0 . Otherwise, with probability $1 - \alpha$ it learns that $i^* > i$. In this latter case, P_0 has no information beyond what it could compute from its input, and P_1 will output a value distributed according to $W_1(x_1)$; hence, incentive compatibility implies that the maximum expected

utility of P_0 is $u_0^* < a_0$. (This u_0^* is the same as in the proof of Theorem 1.) That is, the expected utility of aborting is at most $\alpha \cdot b_0 + (1 - \alpha) \cdot u_0^*$. If

$$\alpha < \frac{a_0 - u_0^*}{b_0 - u_0^*}$$

then $\alpha \cdot b_0 + (1 - \alpha) \cdot u_0^* < a_0$, implying that P_0 does not gain anything beyond what it could have obtained by running the protocol honestly with its actual input.⁴ A calculation exactly as in the proof of Theorem 1 shows that p can be set in such a way that this condition holds.

4 Conclusions and Future Work

Given the stark impossibility results for fairness in a purely *malicious* context [6], it is natural to try to understand whether, or to what extent, fairness is achievable in a *rational* setting. Recent work of Asharov et al. [2] seemed to give a pessimistic answer to this question, as they show a specific case where rational fairness *cannot* be achieved (if correctness better than 1/2 is desired). Our work, in contrast, shows broad feasibility results for rational fairness: roughly, we show that whenever computing the function is a *strict* Nash equilibrium in the ideal world, then it is possible to construct a rational fair protocol computing the function in the real world.

Within the broader context of research at the intersection of game theory and cryptography, our result can be interpreted in two ways:

- Given a “fairness game” defined in an ideal world where there is a trusted entity (i.e., a “mediator”) computing some function on behalf of the parties, a natural question to ask is when a game-theoretic equilibrium in the ideal world can be implemented via a real-world protocol. We do not provide a complete answer to this question, but we do show a partial characterization: roughly, whenever there is a *strict* Nash equilibrium in the ideal world, there is a protocol that induces a computational Nash equilibrium in the real world.
- We show a new setting in which cryptographic impossibility results can be circumvented by assuming rational behavior. Viewed in this light, our results can be seen as a generalization of work on rational secret sharing.

Our work suggests several interesting directions for future research. First, it would be interesting to prove a converse of our result. Fix some f , a distribution on the inputs, and utility functions for the parties. We conjecture that if there exists a rational fair protocol Π for computing f (with respect to this distribution and utilities), then incentive compatibility holds.⁵

⁴ It is possible that, conditioned on the fact that P_0 sent input $x'_0 \neq x_0$ to *ShareGen*, party P_0 can obtain better expected utility by aborting early than by running the protocol to the end. What we claim here, though, is that P_0 will never obtain better expected utility than it would have obtained by *using its actual input* x_0 and then running the protocol to the end.

⁵ As noted earlier, for our positive results it is sufficient for the ideal-world equilibrium to be strict for only one of the parties. With regard to this conjecture, then, the definition of incentive compatibility should be modified appropriately.

It will also be interesting to explore stronger game-theoretic solution concepts in the real world. We construct real-world protocols that induce a computational Nash equilibrium, but one could also aim to construct protocols satisfying some of the stronger equilibrium notions proposed, e.g., in [17,22,23,11].

Another natural extension is the multi-party case. Our protocols can easily be extended to handle *single-player deviations* in that setting; we leave the general case (where parties may collude) as an interesting open question.

Finally, one could consider even more complex settings of the players' utilities, e.g., where the utilities depend on the true output and the actual output of the parties and not just on whether the outputs are correct or incorrect. This would model situations where being "closer" to the right answer is better, or where some answers are more important to get right than others.

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: 25th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 53–62. ACM Press (2006)
2. Asharov, G., Canetti, R., Hazay, C.: Towards a Game Theoretic View of Secure Computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 426–445. Springer, Heidelberg (2011), Full version available at, <http://eprint.iacr.org/2011/137>
3. Asharov, G., Lindell, Y.: Utility dependence in correct and fair rational secret sharing. *Journal of Cryptology* 24(1), 157–202 (2011)
4. Barany, I.: Fair distribution protocols, or how the players replace fortune. *Mathematics of Operations Research* 17, 327–340 (1992)
5. Canetti, R.: Security and composition of multiparty cryptographic protocols. *Journal of Cryptology* 13(1), 143–202 (2000)
6. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: 18th Annual ACM Symposium on Theory of Computing (STOC), pp. 364–369. ACM Press (1986)
7. Crawford, V., Sobel, J.: Strategic information transmission. *Econometrica* 50, 1431–1451 (1982)
8. Dodis, Y., Halevi, S., Rabin, T.: A Cryptographic Solution to a Game Theoretic Problem. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 112–130. Springer, Heidelberg (2000)
9. Dodis, Y., Rabin, T.: Cryptography and game theory. In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.) *Algorithmic Game Theory*, pp. 181–207. Cambridge University Press (2007)
10. Forges, F.: Universal mechanisms. *Econometrica* 58, 1342–1364 (1990)
11. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient Rational Secret Sharing in Standard Communication Networks. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 419–436. Springer, Heidelberg (2010)
12. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. *J. ACM* 58(6) (2011)
13. Gordon, S.D., Katz, J.: Rational Secret Sharing, Revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)

14. Gordon, S.D., Katz, J.: Partial fairness in secure two-party computation. *Journal of Cryptology* 25(1), 14–40 (2012)
15. Gradwohl, R.: Rationality in the Full-Information Model. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 401–418. Springer, Heidelberg (2010)
16. Gradwohl, R., Livne, N., Rosen, A.: Sequential rationality in cryptographic protocols. In: 51st FOCS Annual Symposium on Foundations of Computer Science (FOCS), pp. 623–632. IEEE (2010)
17. Halpern, J., Teague, V.: Rational secret sharing and multiparty computation. In: 36th Annual ACM Symposium on Theory of Computing (STOC), pp. 623–632. ACM Press (2004)
18. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably Secure Devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)
19. Izmalkov, S., Lepinski, M., Micali, S.: Perfect implementation. *Games and Economic Behavior* 71(1), 121–140 (2011), <http://hdl.handle.net/1721.1/50634>
20. Izmalkov, S., Micali, S., Lepinski, M.: Rational secure computation and ideal mechanism design. In: 46th FOCS Annual Symposium on Foundations of Computer Science (FOCS), pp. 585–595. IEEE (2005), Full version available at, <http://dspace.mit.edu/handle/1721.1/38208>
21. Katz, J.: Bridging Game Theory and Cryptography: Recent Results and Future Directions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 251–272. Springer, Heidelberg (2008)
22. Kol, G., Naor, M.: Cryptography and Game Theory: Designing Protocols for Exchanging Information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg (2008)
23. Kol, G., Naor, M.: Games for exchanging information. In: 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 423–432. ACM Press (2008)
24. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely fair SFE and coalition-safe cheap talk. In: 23rd ACM PODC Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 1–10. ACM Press (2004)
25. Lepinski, M., Micali, S., Shelat, A.: Collusion-free protocols. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 543–552. ACM Press (2005)
26. Lysyanskaya, A., Triandopoulos, N.: Rationality and Adversarial Behavior in Multi-party Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 180–197. Springer, Heidelberg (2006)
27. Micali, S., Shelat, A.: Purely Rational Secret Sharing (Extended Abstract). In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 54–71. Springer, Heidelberg (2009)
28. Moran, T., Naor, M., Segev, G.: An Optimally Fair Coin Toss. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 1–18. Springer, Heidelberg (2009)
29. Ong, S.J., Parkes, D.C., Rosen, A., Vadhan, S.: Fairness with an Honest Minority and a Rational Majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)

A Incentive Compatibility in the Byzantine Setting

Here, for the sake of illustration, we present a function f , input distributions, and a set of utilities that are incentive compatible in the Byzantine setting. Let f be the equality predicate over domain $\{0, 1, 2\}$, and assume the parties' inputs are independently and uniformly distributed over this set. The utilities

of the parties are symmetric, and correspond to each player receiving utility 3 from getting the correct answer and (independently) utility 1 if the other player outputs the wrong answer. I.e.,

$$a_0 = a_1 = 3; \quad b_0 = b_1 = 4; \quad c_0 = c_1 = 0; \quad d_0 = d_1 = 1.$$

Let W_0, W_1 be point distributions on the output '0'. (These do not have full support, though Lemma 1 shows that they can be modified so that they do.) We want to show that $\left((\text{cooperate}, W_0), (\text{cooperate}, W_1) \right)$ is a strict Nash equilibrium in the Byzantine setting.

By playing honestly, each party gets utility 3. Consider deviations by P_0 (the case of a deviating P_1 is analogous), and assume without loss of generality that the true input x_0 is equal to 0.

P_0 could send \perp to the trusted third party. In that case, P_0 does best by outputting '0'; its expected utility (recalling that P_1 also outputs '0') is then

$$3 \cdot \Pr[x_1 \neq 0] + 1 \cdot \Pr[x_1 = 0] = \frac{2}{3} * 3 + \frac{1}{3} * 1 = 7/3 < 3.$$

Alternately, P_0 may change its input; without loss of generality, say it uses input $x'_0 = 1$. Here there are two sub-cases, depending on the output returned by the trusted party. If this output is '0', then $x_1 \neq 1$ but P_1 will output '0'. Regardless of what P_0 outputs, it is correct (and gets utility 3) with probability 1/2 and P_1 is incorrect (and so P_0 gets utility 1) with probability 1/2; the expected utility of P_0 is thus 2. If the trusted party returns '1' then P_1 outputs '1' (recall that P_1 outputs what it is given by the trusted party) which is wrong, whereas P_0 learns P_1 's input and so can output the correct answer. Thus, in this case, P_0 obtains utility 4. Overall, then, the expected utility P_0 obtains by changing its input is

$$\frac{2}{3} \cdot 2 + \frac{1}{3} \cdot 4 = 8/3 < 3.$$

We conclude that the given scenario is a (Bayesian) strict Nash equilibrium.