A Computational Game-Theoretic Framework for Cryptography

Joseph Y. Halpern
Cornell University
halpern@cs.cornell.edu

Rafael Pass
Cornell University
rafael@cs.cornell.edu

December 31, 2009*

Abstract

We develop a definition of protocol security relying on game-theoretic notions of implementation. We show that a natural special case of this this definition is equivalent to a variant of the traditional cryptographic definition of protocol security; this result shows that, when taking computation into account, the two approaches used for dealing with "deviating" players in two different communities—*Nash equilibrium* in game theory and *zero-knowledge "simulation"* in cryptography—are intimately related. Other special cases of our definition instead lead to more practical protocols and circumvent known lower bounds with respect to the cryptographic notion of security.

*First draft from April 2007. Halpern is supported in part by NSF grants ITR-0325453, IIS-0534064, IIS-0812045, and IIS-0911036, and by AFOSR grants FA9550-08-1-0438 and FA9550-09-1-0266, and ARO grant W911NF-09-1-0281. Pass is supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR Award FA9550-08-1-0197, and BSF Grant 2006317.

1 Introduction

It is often simpler to design and analyze mechanisms when assuming that players have access to a trusted mediator through which they can communicate. However, such a trusted mediator can be hard to find. A central question in both cryptography and game theory is investigating under what circumstances mediators can be replaced—or *implemented*—by simple "unmediated" communication between the players. There are some significant differences between the approaches used by the two communities to formalize this question.

The cryptographic notion of a *secure computation* [Goldreich, Micali, and Wigderson 1986] considers two types of players: *honest* players and *malicious* players. Honest players are assumed to faithfully execute the prescribed protocol using their intended input; malicious players, on the other hand, are assumed to do anything in their power to undermine the security of honest players. Roughly speaking, a protocol Π is said to securely implement the mediator $\mathcal{F}$ if (1) the malicious players cannot influence the output of the communication phase any more than they could have by communicating directly with the mediator; this is called *correctness*, and (2) the malicious players cannot "learn" more than what can be efficiently computed from only the output of mediator; this is called *privacy*. These properties are formalized through the *zero-knowledge simulation paradigm* [Goldwasser, Micali, and Rackoff 1989]: roughly, we require that any "harm" done by an adversary in the protocol execution could be simulated by a polynomially-bounded Turing machine, called the *simulator*, that communicates only with the mediator. Three levels of security are usually considered: *perfect*, *statistical*, and *computational*. Perfect security guarantees that correctness and privacy hold with probability 1; statistical security allows for a "negligible" error probability; and computational security considers only adversaries that can be implemented by

The traditional game-theoretic notion of implementation (see [Forges 1986; Forges 1990]) does not explicitly consider properties such as privacy and correctness, but instead requires that the implementation preserve a given Nash equilibrium of the mediated game. Roughly speaking, the game-theoretic notion of implementation says that a strategy profile $\vec{\sigma}$ implements a mediator $\mathcal{F}$ if, as long as it is a Nash equilibrium for the players to tell the mediator their type and output what the mediator recommends, then $\vec{\sigma}$ is a Nash equilibrium in the "cheap talk" game (where the players just talk to each other, rather than talking to a mediator) that has the same distribution over outputs as when the players talk to the mediator. In other words, whenever a set of parties have incentives to tell the mediator their inputs, they also have incentives to honestly use $\vec{\sigma}$ using the same inputs, and get the same distribution over outputs in both cases.

The key differences between the notions are that the game-theoretic notion does not consider privacy issues and the cryptographic notion does not consider incentives: the game-theoretic notion talks about preserving Nash equilibria (which cannot be done in the cryptographic notion, since there are no incentives), while the cryptographic notion talks about security against malicious adversaries.

Although the cryptographic notion does not consider incentives, it is nonetheless stronger than the game-theoretic notion. More precisely, the game-theoretic notion of implementation; that is, all perfectly-secure implementations are also game-theoretic implementations.[1] A corresponding implication holds for statistically- and computationally-secure implementations if we consider appropriate variants of game-theoretic implementation that require only that running Π is an ϵ-Nash equilibrium, resp., a "computational" ϵ-Nash equilibrium, where players are restricted to using polynomially-bounded Turing machines; see [Dodis, Halevi, and Rabin 2000; Dodis and Rabin 2007; Lepinski, Micali, Peikert, and Shelat 2004].[2]

[1] For completness, we formalize this in Proposition 4.1.

[2] [Dodis, Halevi, and Rabin 2000; Lepinski, Micali, Peikert, and Shelat 2004] consider only implementations of correlated equilibrium, but the same proof extends to arbitrary mediators as well.

The converse implication does not hold. Since the traditional game-theoretic notion of implementation does not consider computational cost (indeed, more generally, traditional solution concepts in game theory do not take computation into account), it cannot take into account issues like computational efficiency, or the computational advantages possibly gained by using Π, issues that are critical in the cryptographic notion. Another difference is that the game-theoretic definition does not consider coalitions of players.

There have been several recent works that attempt to bridge these two notions (e.g., [Abraham, Dolev, Gonen, and Halpern 2006; Dodis, Halevi, and Rabin 2000; Gordon and Katz 2006; Halpern and Teadgue 2004; Izmalkov, Lepinski, and Micali 2008; Kol and Naor 2008; Shoham and Tennenholtz 2005])). Most notably, Izmalkov, Lepinski and Micali [Izmalkov, Lepinski, and Micali 2008] (see also [Kol and Naor 2008]) present a "hybrid" definition, where correctness is defined through the game-theoretic notion,[3] and privacy through the zero-knowledge paradigm. In other words, the privacy part—which is defined through the cryptographic paradigm—recognizes that computation is costly to players. But the incentive part—which is defined through the game-theoretic paradigm—does not. As a consequence, if computation is a costly resource for the players, none of the earlier definitions provide explicit guarantees about players' incentives to correctly execute a protocol with their intended input. For instance, a player might not want to execute a protocol if doing so is too computationally expensive. Similarly, a player i might want to change its input if executing the protocol gives some other player j a computational advantage in determining player i's input.

We suggest a different approach, based on the game-theoretic approach. Roughly speaking, we say that Π implements a mediator $\mathcal{F}$ if for *all* games G—including games where computation is costly—that use $\mathcal{F}$ for which (the utilities in G are such that) it is an equilibrium for the players to truthfully tell $\mathcal{F}$ their inputs, running Π on the same set of inputs (and with the same utility functions) is also an equilibrium and produces the same distribution over outputs as $\mathcal{F}$.[4] To model games where computation is costly, we rely on (and extend) a framework we introduced in a companion paper [Halpern and Pass 2008], which generalizes earlier approaches in the literature (e.g., [Rubinstein 1986; Ben-Sasson, Kalai, and Kalai 2007]). Roughly speaking, whereas in traditional games, the utility of a player only depends on the types and the actions of players, in a computational game, the players' utilities depend also on the *complexities* of the strategies of the players; the complexity of a strategy—represented as a Turing machine—could for instance, represent the running time of, or space used by, the machine on a particular input. (To provide security with respect to coalitions of players, we also allow G to be a coalitional game [Neumann and Morgenstern 1947].)

Note that by requiring the implementation to work for all games, not only do we ensure that players have proper incentives to execute protocols with their intended input, even if they consider computation a costly resource, but we get the privacy and correctness requirements "for free". For suppose that, when using Π, some information about i's input is revealed to j. We consider a zero-sum game G where a player j gains some significant utility by having this information. In this game, i will not want to use Π. However, our notion of implementation requires that, even with the utilities in G, i should want to use Π if i is willing to use the mediator $\mathcal{F}$. (This argument depends on the fact that we consider games where computation is costly; the fact that j gains information about i's input may mean that j can do some computation faster with this information than without it.) As a consequence, our definition gives a relatively simple (and strong) way of

[3] In fact, Izmalkov, Lepinski, and Micali [2008] consider an even stronger notion of implementation, which they call *perfect implementation*. See Section 3 for more details.

[4] While the definitions of implementation in the game-theory literature (e.g., [Forges 1986; Forges 1990]) do not stress the uniformity of the implementation—that is, the fact that it works for all games—the implementations provided are in fact uniform in this sense.

formalizing the security of protocols, relying only on basic notions from game theory.

Perhaps surprisingly, we show that, under weak restrictions on the utility functions of the players (essentially, that players never prefer to compute more), our notion of implementation is equivalent to a variant of the cryptographic notion of *precise secure computation*, recently introduced by Micali and Pass [2006]. Roughly speaking, the notion of precise secure computation requires that any harm done by an adversary in a protocol execution could have been done also by a simulator, using the same complexity distribution as the adversary. In contrast, the traditional definition of secure computation requires only that the simulator's complexity preserves the *worst-case* complexity of the adversary. By considering specific measures of complexity (such as worst-case running time and space) we can obtain a game-theoretic characterization of the traditional (i.e., "non-precise") notion of secure computation.

This result shows that the two approaches used for dealing with "deviating" players in two different communities—*Nash equilibrium* in game theory, and *zero-knowledge "simulation"* in cryptography—are intimately connected; indeed, they are essentially equivalent in the context of implementing mediators, once we take the cost of computation into account. It follows immediately from our result that known protocols, such as those in [Ben-Or, Goldwasser, and Wigderson 1988; Canetti 2001; Goldreich, Micali, and Wigderson 1987; Izmalkov, Lepinski, and Micali 2008; Micali and Pass 2006; Micali and Pass 2007], satisfy our game-theoretic notion of implementation. Moreover, lower bounds for the traditional notion of secure computation immediately yield lower bounds for implementations.

Our equivalence result might seem like a negative result: it demonstrates that considering only rational players (as opposed to arbitrary malicious players) does not facilitate protocol design. We emphasize, however, that for the equivalence to hold, we must consider implementations with only weak restrictions on the utility functions. In some many settings, it might be reasonable to consider stronger restrictions on the utility functions of players: for instance, that players strictly prefer to compute less, that players do not want to be caught "cheating", or that players might not be concerned about the privacy of part of their inputs. As we show, it is easier to provide implementations for such (restricted) classes of games, allowing us to circumvent classical impossibility results (e.g., [Cleve 1986]) for the traditional notion of secure computation. We believe that this generality is an important advantage of a notion of security that does not rely on the zero-knowledge simulation paradigm.[5] Indeed, our work has already lead to several followups: Micali and Shelat [2009] consider costly computation in the context of secret-sharing, Rosen and Shelat [2000] consider it in the context of concurrent security, and Miltersen et al. [2009] provide an alternative approach for capturing privacy using utility.

2 A Computational Game-Theoretic Framework

2.1 Bayesian Games

We model costly computation using *Bayesian machine games*, introduced by us in a companion paper [Halpern and Pass 2008]. To explain our approach, we first review the standard notion of a *Bayesian game*. A Bayesian game is a game of incomplete information, where each player makes a single move. The "incomplete information" is captured by assuming that nature makes an initial move, and chooses for each player i a *type* in some set T_i. Player i's type can be viewed as describing i's private information. For ease of exposition, we assume in this paper that the set N of players is

[5] We mention that almost all general notions of security have been based on the zero-knowledge simulation paradigm. One notable exception is the definition of *input-indistingushable computation* of Micali, Pass and Rosen [2006]. This notion is useful in circumventing impossibility results regarding concurrency, but still suffers from impossibility results [Cleve 1986].

always $[m] = \{1, \ldots, m\}$, for some m. If $N = [m]$, the set $T = T_1 \times \ldots \times T_m$ is the *type space.* As is standard, we assume that there is a commonly-known probability distribution Pr on the type space T. Each player i must choose an action from a space A_i of actions. Let $A = A_1 \times \ldots \times A_n$ be the set of action profiles. A Bayesian game is characterized by the tuple $([m], T, A, \Pr, \vec{u})$, where $[m]$ is the set of players, T is the type space, A is the set of joint actions, and $\vec{u}$ is the utility function, where $u_i(\vec{t}, \vec{a})$ is player i's utility (or payoff) if the type profile is $\vec{t}$ and action profile $\vec{a}$ is played.

In general, a player's choice of action will depend on his type. A *strategy* for player i is a function from T_i to $\Delta(A_i)$ (where, as usual, we denote by $\Delta(X)$ the set of distributions on the set X). If σ is a strategy for player i, $t \in T_i$ and $a \in A_i$, then $\sigma(t)(a)$ denotes the probability of action a according to the distribution on acts induced by $\sigma(t)$. Given a joint strategy $\vec{\sigma}$, we can take $u_i^{\vec{\sigma}}$ to be the random variable on the type space T defined by taking $u_i^{\vec{\sigma}}(\vec{t}) = \sum_{\vec{a} \in A} (\sigma_1(t_1)(a_1) \times \ldots \times \sigma_m(t_m)(a_m)) u_i(\vec{t}, \vec{a})$. Player i's expected utility if $\vec{\sigma}$ is played, denoted $U_i(\vec{\sigma})$, is then just $\mathbf{E}_{\Pr}[u_i^{\vec{\sigma}}] = \sum_{\vec{t} \in T} \Pr(\vec{t}) u_i^{\vec{\sigma}}(\vec{t})$.

2.2 Bayesian Machine Games

In a Bayesian game, it is implicitly assumed that computing a strategy—that is, computing what move to make given a type—is free. We want to take the cost of computation into account here. To this end, we consider what we call *Bayesian machine games*, where we replace strategies by *machines.* For definiteness, we take the machines to be Turing machines, although the exact choice of computing formalism is not significant for our purposes. Given a type, a strategy in a Bayesian game returns a distribution over actions. Similarly, given as input a type, the machine returns a distribution over actions. As is standard, we model the distribution by assuming that the machine actually gets as input not only the type, but a random string of 0s and 1s (which can be thought of as the sequence of heads and tails), and then (deterministically) outputs an action. Just as we talk about the expected utility of a strategy profile in a Bayesian game, we can talk about the expected utility of a machine profile in a Bayesian machine game. However, we can no longer compute the expected utility by just taking the expectation over the action profiles that result from playing the game. A player's utility depends not only on the type profile and action profile played by the machine, but also on the "complexity" of the machine given an input. The complexity of a machine can represent, for example, the running time or space usage of the machine on that input, the size of the program description, or some combination of these factors. For simplicity, we describe the complexity by a single number, although, since a number of factors may be relevant, it may be more appropriate to represent it by a tuple of numbers in some cases. (We can, of course, always encode the tuple as a single number, but in that case, "higher" complexity is not necessarily worse.) Note that when determining player i's utility, we consider the complexity of all machines in the profile, not just that of i's machine. For example, i might be happy as long as his machine takes fewer steps than j's.

We assume that nature has a type in $\{0,1\}^*$. While there is no need to include a type for nature in standard Bayesian games (we can effectively incorporate nature's type into the type of the players), once we take computation into account, we obtain a more expressive class of games by allowing nature to have a type (since the complexity of computing the utility may depend on nature's type). We assume that machines take as input strings of 0s and 1s and output strings of 0s and 1s. Thus, we assume that both types and actions can be represented as elements of $\{0,1\}^*$. We allow machines to randomize, so given a type as input, we actually get a distribution over strings. To capture this, we assume that the input to a machine is not only a type, but also a string chosen with uniform probability from $\{0,1\}^\infty$ (which we can view as the outcome of an infinite sequence of coin tosses). The machine's output is then a deterministic function of its type and the infinite random string.

We use the convention that the output of a machine that does not terminate is a fixed special symbol ω. We define a *view* to be a pair (t, r) of two bitstrings; we think of t as that part of the type that is read, and r is the string of random bits used. (Our definition is slightly different from the traditional way of defining a view, in that we include only the parts of the type and the random sequence *actually* read. If computation is not taken into account, there is no loss in generality in including the full type and the full random sequence, and this is what has traditionally been done in the literature. However, when computation is costly, this might no longer be the case.) We denote by $t; r$ a string in $\{0,1\}^*; \{0,1\}^*$ representing the view. (Note that here and elsewhere, we use ";" as a special symbol that acts as a separator between strings in $\{0,1\}^*$.) If $v = (t; r)$ and r is a finite string, we take $M(v)$ to be the output of M given input type t and random string $r \cdot 0^\infty$.

We now briefly review our computational game-theoretic framework [Halpern and Pass 2008], which will form the basis of our game-theoretic definition of security. We then introduce some additional notions that will be necessary to capture security. We use a *complexity function* $\mathscr{C} : \mathbf{M} \times \{0,1\}^*; \{0,1\}^* \cup \{0,1\}^\infty \to I\!\!N$, where $\mathbf{M}$ denotes the set of Turing machines to describe the complexity of a machine given a view. If $t \in \{0,1\}^*$ and $r \in \{0,1\}^\infty$, we identify $\mathscr{C}(M, t; r)$ with $\mathscr{C}(M, t; r')$, where r' is the finite prefix of r actually used by M when running on input t with random string r.

For now, we assume that machines run in isolation, so the output and complexity of a machine does not depend on the machine profile. We remove this restriction in the next section, where we allow machines to communicate with mediators (and thus, implicitly, with each other via the mediator).

Definition 2.1 (Bayesian machine game) *A* Bayesian machine game G *is described by a tuple* $([m], \mathcal{M}, T, \Pr, \mathscr{C}_1, \ldots, \mathscr{C}_m, u_1, \ldots, u_m)$, *where*

- $[m] = \{1, \ldots, m\}$ *is the set of players;*
- $\mathcal{M}$ *is the set of possible machines;*
- $T \subseteq (\{0,1\}^*)^{m+1}$ *is the set of type profiles, where the* $(m+1)$*st element in the profile corresponds to nature's type;*
- $\Pr$ *is a distribution on* T*;*
- $\mathscr{C}_i$ *is a complexity function;*
- $u_i : T \times (\{0,1\}^*)^m \times I\!\!N^m \to I\!\!R$ *is player* i*'s utility function. Intuitively,* $u_i(\vec{t}, \vec{a}, \vec{c})$ *is the utility of player* i *if* $\vec{t}$ *is the type profile,* $\vec{a}$ *is the action profile (where we identify* i*'s action with* M_i*'s output), and* $\vec{c}$ *is the profile of machine complexities.*

We can now define the expected utility of a machine profile. Given a Bayesian machine game $G = ([m], \mathcal{M}, \Pr, T, \vec{\mathscr{C}}, \vec{u})$ and $\vec{M} \in \mathcal{M}^m$, define the random variable $u_i^{G,\vec{M}}$ on $T \times (\{0,1\}^\infty)^m$ (i.e., the space of type profiles and sequences of random strings) by taking

$$u_i^{G,\vec{M}}(\vec{t}, \vec{r}) = u_i(\vec{t}, M_1(t_1; r_1), \ldots, M_m(t_m; r_m), \mathscr{C}_1(M_1, t_1; r_1), \ldots, \mathscr{C}_m(M_m, t_m)).$$

Note that there are two sources of uncertainty in computing the expected utility: the type t and realization of the random coin tosses of the players, which is an element of $(\{0,1\}^\infty)^k$. Let $\Pr_U^k$ denote the uniform distribution on $(\{0,1\}^\infty)^k$. Given an arbitrary distribution $\Pr_X$ on a space X, we write $\Pr_X^{+k}$ to denote the distribution $\Pr_X \times \Pr_U^k$ on $X \times (\{0,1\}^\infty)^k$. If k is clear from context or not relevant, we often omit it, writing $\Pr_U$ and $\Pr_X^+$. Thus, given the probability $\Pr$ on T, the expected utility of player i in game G if $\vec{M}$ is used is the expectation of the random variable $u_i^{G,\vec{M}}$ with respect to the distribution $\Pr^+$ (technically, $\Pr^{+m}$): $U_i^G(\vec{M}) = \mathbf{E}_{\Pr^+}[u_i^{G,\vec{M}}]$. Note that this notion of utility allows an agent to prefer a machine that runs faster to one that runs slower, even if they give the same output, or to prefer a machine that has faster running time to one that gives a better output. Because we allow the utility to depend on the whole profile of complexities, we can capture a situation where i can be "happy" as long as his machine runs faster than j's machine. Of course, an important special case is where i's utility depends only on his own complexity. All of our results continue to hold if we make this restriction.

Given the definition of utility above, we can now define (ϵ-) Nash equilibrium in the standard way.

Definition 2.2 (Nash equilibrium in machine games) *Given a Bayesian machine game $G = ([m], \mathcal{M}, T, \Pr, \vec{\mathscr{C}}, \vec{u})$, a machine profile $\vec{M} \in \mathcal{M}^m$, and $\epsilon \geq 0$, M_i is an ϵ-best response to $\vec{M}_{-i}$ if, for every $M_i' \in \mathcal{M}$,*

$$U_i^G[(M_i, \vec{M}_{-i})] \geq U_i^G[(M_i', \vec{M}_{-i})] - \epsilon.$$

*(As usual, $\vec{M}_{-i}$ denotes the tuple consisting of all machines in $\vec{M}$ other than M_i.) $\vec{M}$ is an ϵ-*Nash equilibrium *of G if, for all players i, M_i is an ϵ-best response to $\vec{M}_{-i}$. A* Nash equilibrium *is a 0-Nash equilibrium.*

For further intuition into this notion, we refer the reader to our companion paper [Halpern and Pass 2008] where we provide a more in-depth study of traditional game-theoretic questions (such as existence of Nash equilibria) for computational games, and show how computational considerations can help explain experimentally-observed phenomena in well-studied games in a psychologically appealing way.

One immediate advantage of taking computation into account is that we can formalize the intuition that ϵ-Nash equilibria are reasonable, because players will not bother changing strategies for a gain of ϵ. Intuitively, the complexity function can "charge" ϵ for switching strategies. Specifically, an ϵ-Nash equilibrium $\vec{M}$ can be converted to a Nash equilibrium by modifying player i's complexity function to incorporate the overhead of switching from M_i to some other strategy, and having player i's utility function decrease by $\epsilon' > \epsilon$ if the switching cost is nonzero; we omit the formal details here. Thus, the framework lets us incorporate explicitly the reasons that players might be willing to play an ϵ-Nash equilibrium. This justification of ϵ-Nash equilibrium seems particularly appealing when designing mechanisms (e.g., cryptographic protocols) where the equilibrium strategy is made "freely" available to the players (e.g., it is accessible on a web-page), but any other strategy requires some implementation cost.

In order to define our game-theoretic notion of protocol security, we need to introduce some extensions to the basic framework of [Halpern and Pass 2008]. Specifically, we will be interested only in equilibria that are robust in a certain sense, and we want equilibria that deal with deviations by coalitions, since the security literature allows malicious players that deviate in a coordinated way. Furthemore, we need to formally define mediated games.

2.3 Computationally Robust Nash Equilibrium

Computers get faster, cheaper, and more powerful every year. Since utility in a Bayesian machine game takes computational complexity into account, this suggests that an agent's utility function will change when he replaces one computer by a newer computer. We are thus interested in *robust* equilibria, intuitively, ones that continue to be equilibria (or, more precisely, ϵ-equilibria for some appropriate ϵ) even if agents' utilities change as a result of upgrading computers.

Definition 2.3 (Computationally robust Nash equilibrium) *Let $p : \mathbb{N} \rightarrow \mathbb{N}$. The complexity function $\mathscr{C}'$ is* at most a p-speedup *of the complexity function $\mathscr{C}$ if, for all machines M and views v,*

$$\mathscr{C}'(M, v) \leq \mathscr{C}(M, v) \leq p(\mathscr{C}'(M, v)).$$

Game $G' = ([m'], \mathcal{M}', \Pr', \vec{\mathscr{C}}', \vec{u}')$ *is* at most a p-speedup *of game* $G = ([m], \mathcal{M}, \Pr, \vec{\mathscr{C}}, \vec{u})$ *if* $m' = m$, $\Pr = \Pr'$ *and* $\vec{u} = \vec{u}'$ *(i.e.,* G' *and* G' *differ only in their complexity and machine profiles), and* $\mathscr{C}_i'$ *is at most a* p*-speedup of* $\mathscr{C}_i$*, for* $i = 1, \ldots, m$. M_i *is a* p-robust ϵ-best response *to* $\vec{M}_{-i}$ *in* G*, if for every game* $\tilde{G}$ *that is at most a* p*-speedup of* G*,* M_i *is an* ϵ*-best response to* $\vec{M}_{-i}$*.* $\vec{M}$ *is a* p-robust ϵ-equilibrium *for* G *if, for all* i*,* M_i *is an* p*-robust* ϵ*-best response to* $\vec{M}_{-i}$*.*

Intuitively, if we think of complexity as denoting running time and $\mathscr{C}$ describes the running time of machines (i.e., programs) on an older computer, then $\mathscr{C}'$ describes the running time of machines on an upgraded computer. For instance, if the upgraded computer runs at most twice as fast as the older one (but never slower), then $\mathscr{C}'$ is a $\bar{2}$-speedup of $\mathscr{C}$, where $\bar{k}$ denotes the constant function k. Clearly, if $\vec{M}$ is a Nash equilibrium of G, then it is a $\bar{1}$-robust equilibrium. We can think of p-robust equilibrium as a refinement of Nash equilibrium for machine games, just like *sequential equilibrium* [Kreps and Wilson 1982] or *perfect equilibrium* [Selten 1975]; it provides a principled way of ignoring "bad" Nash equilibria. Note that in games where computation is free, every Nash equilibrium is also computationally robust.

2.4 Coalition Machine Games

We strengthen the notion of Nash equilibrium to allow for deviating coalitions. Towards this goal, we consider a generalization of Bayesian machine games called *coalition machine games*, where, in the spirit of *coalitional games* [Neumann and Morgenstern 1947], each *subset* of players has a complexity function and utility function associated with it. In analogy with the traditional notion of Nash equilibrium, which considers only "single-player" deviations, we consider only "single-coalition" deviations.

More precisely, given a subset Z of $[m]$, we let $-Z$ denote the set $[m]/Z$. We say that a machine M_Z' *controls* the players in Z if M_Z' controls the input and output tapes of the players in set Z (and thus can coordinate their outputs). In addition, the adversary that controls Z has its own input and output tape. A *coalition machine game* G is described by a tuple $([m], \mathcal{M}, \Pr, \vec{\mathscr{C}}, \vec{u})$, where $\vec{\mathscr{C}}$ and $\vec{u}$ are sequences of complexity functions $\mathscr{C}_Z$ and utility functions u_Z, respectively, one for each subset Z of $[m]$; m, $\mathcal{M}$, and Pr are defined as in Definition 2.1. In contrast, the utility function u_Z for the set Z is a function $T \times (\{0,1\}^*)^m \times (I\!N \times I\!N^{m-|Z|+1}) \rightarrow I\!R$, where $u_Z(\vec{t}, \vec{a}, (c_Z, \vec{c}_{-Z}))$ is the utility of the coalition Z if $\vec{t}$ is the (length $m+1$) type profile, $\vec{a}$ is the (length m) action profile (where we identify i's action as player i output), c_Z is the complexity of the coalition Z, and $\vec{c}_{-Z}$ is the (length $m - |Z|$) profile of machine complexities for the players in $-Z$. The complexity c_Z is a measure of the complexity according to whoever controls coalition Z of running the coalition. Note that even if the coalition is controlled by a machine M_Z' that lets each of the players in Z perform independent computations, the complexity of M_Z' is not necessarily some function of the complexities c_i of the players $i \in Z$ (such as the sum or the max). Moreover, while cooperative game theory tends to focus on *superadditive* utility functions, where the utility of a coalition is at least the sum of the utilities of any partition of the coalition into sub-coalitions or individual players, we make no such restrictions; indeed when taking complexity into account, it might very well be the case that larger coalitions are more expensive than smaller ones. Also note that, in our calculations, we assume that, other than the coalition Z, all the other players play individually (so that we use c_i for $i \notin Z$); there is at most one coalition in the picture. Having defined u_Z, we can define the expected utility of the group Z in the obvious way.

The *benign machine for coalition* Z, denoted M_Z^b, is the one where that gives each player $i \in Z$ its true input, and each player $i \in Z$ outputs the output of M_i; M_Z^b write nothing on its output tape. Essentially, the benign machine does exactly what all the players in the coalition would have done anyway. We now extend the notion of Nash equilibrium to deal with coalitions; it requires that in an equilibrium $\vec{M}$, no coalition does (much) better than it would using the benign machine, according to the utility function for that coalition.

Definition 2.4 (Nash equilibrium in coalition machine games) *Given an m-player coalition machine game G, a machine profile $\vec{M}$, a subset Z of $[m]$ and $\epsilon \geq 0$, M_Z^b is an ϵ-best response to $\vec{M}_{-Z}$ if, for every coalition machine $M'_Z \in \mathcal{M}$,*

$$U_Z^G[(M_Z^b, \vec{M}_{-Z})] \geq U_Z^G[(M'_Z, \vec{M}_{-Z})] - \epsilon.$$

Given a set $\mathcal{Z}$ of subsets of $[m]$, $\vec{M}$ is a $\mathcal{Z}$-safe ϵ-Nash equilibrium *for G if, for all $Z \in \mathcal{Z}$, M_Z^b is an ϵ-best response to $\vec{M}_{-Z}$.*

Our notion of coalition games is quite general. In particular, if we disregard the costs of computation, it allows us to capture some standard notions of coalition resistance in the literature, by choosing u_Z appropriately. For example, Aumann's [1959] notion of *strong equilibrium* requires that, for all coalitions, it is not the case that there is a deviation that makes everyone in the coalition strictly better off. To capture this, fix a profile $\vec{M}$, and define $u_Z^{\vec{M}}(M'_Z, \vec{M}'_{-Z}) = \min_{i \in Z} u_i(M'_Z, \vec{M}'_{-Z}) - u_i(\vec{M})$.[6] We can capture the notion of *k-resilient equilibrium* [Abraham, Dolev, Gonen, and Halpern 2006; Abraham, Dolev, and Halpern 2008], where the only deviations allowed are by coalitions of size at most k, by restricting $\mathcal{Z}$ to consist of sets of cardinality at most k (so a 1-resilient equilibrium is just a Nash equilibrium). Abraham et al. [2006, 2008] also consider a notion of *strong* k-resilient equilibrium, where there is no deviation by the coalition that makes even one coalition member strictly better off. We can capture this by replacing the min in the definition of $u_Z^{\vec{M}}$ by max.

2.5 Machine Games with Mediators

Up to now we have assumed that the only input a machine receives is the initial type. This is appropriate in a normal-form game, but does not allow us to consider game where players can communicate with each other and (possibly) with a trusted mediator. We now extend Bayesian machine games to allow for communication. For ease of exposition, we assume that all communication passes between the players and a trusted mediator. Communication between the players is modeled by having a trusted mediator who passes along messages received from the players. Thus, we think of the players as having reliable communication channels to and from a mediator; no other communication channels are assumed to exist.

The formal definition of a Bayesian machine game with a mediator is similar in spirit to that of a Bayesian machine game, but now we assume that the machines are *interactive* Turing machines, that can also send and receive messages. We omit the formal definition of an interactive Turing machine (see, for example, [Goldreich 2001]); roughly speaking, the machines use a special tape where the message to be sent is placed and another tape where a message to be received is written. The mediator is modeled by an interactive Turing machine that we denote $\mathcal{F}$. A *Bayesian machine game with a mediator* (or a mediated Bayesian machine game) is thus a pair $(G, \mathcal{F})$, where $G = ([m], \mathcal{M}, \Pr, \mathscr{C}_1, \ldots, \mathscr{C}_n, u_1, \ldots, u_n)$ is a Bayesian machine game (except that $\mathcal{M}$ here denotes a set of *interactive* machines) and $\mathcal{F}$ is an interactive Turing machine.

Like machines in Bayesian machine games, interactive machines in a game with a mediator take as argument a view and produce an outcome. Since what an interactive machine does can depend on the history of messages sent by the mediator, the message history (or, more precisely, that part

[6] Note that if we do not disregard the cost of computation, it is not clear how to define the individual complexity of a player that is controlled by M'_Z.

of the message history actually read by the machine) is also part of the view. Thus, we now define a view to be a string $t;h;r$ in $\{0,1\}^*;\{0,1\}^*;\{0,1\}^*$, where, as before, t is that part of the type actually read and r is a finite bitstring representing the string of random bits actually used, and h is a finite sequence of messages received and read. Again, if $v = t;h;r$, we take $M(v)$ to be the output of M given the view.

We assume that the system proceeds in synchronous stages; a message sent by one machine to another in stage k is received by the start of stage $k+1$. More formally, following [Abraham, Dolev, Gonen, and Halpern 2006], we assume that a *stage* consists of three phases. In the first phase of a stage, each player i sends a message to the mediator, or, more precisely, player i's machine M_i computes a message to send to the mediator; machine M_i can also send an empty message, denoted λ. In the second phase, the mediator receives the message and mediator's machine sends each player i a message in response (again, the mediator can send an empty message). In the third phase, each player i performs an action other than that of sending a message (again, it may do nothing). The messages sent and the actions taken can depend on the machine's message history (as well as its initial type).

We can now define the expected utility of a profile of interactive machines in a Bayesian machine game with a mediator. The definition is similar in spirit to the definition in Bayesian machine games, except that we must take into account the dependence of a player's actions on the message sent by the mediator. Let $\mathsf{view}_i(\vec{M},\mathcal{F},\vec{t},\vec{r})$ denote the string $(t_i;h_i;r_i)$ where h_i denotes the messages received by player i if the machine profile is $\vec{M}$, the mediator uses machine $\mathcal{F}$, the type profile is $\vec{t}$, and $\vec{r}$ is the profile of random strings used by the players and the mediator. Given a mediated Bayesian machine game $G' = (G,\mathcal{F})$, we can define the random variable $u_i^{G',\vec{M}}(\vec{t},\vec{r})$ as before, except that now $\vec{r}$ must include a random string for the mediator, and to compute the outcome and the complexity function, M_j gets as an argument $\mathsf{view}_j(\vec{M},\mathcal{F},\vec{t},\vec{r})$, since this is the view that machine M_j gets in this setting. Finally, we define $U_i^{G'}(\vec{M}) = \mathbf{E}_{\Pr^+}[u_i^{G',\vec{M}}]$ as before, except that now $\Pr^+$ is a distribution on $T \times (\{0,1\}^\infty)^{n+1}$ rather than $T \times (\{0,1\}^\infty)^n$, since we must include a random string for the mediator as well as the players' machines. We can define Nash equilibrium and computationally robust Nash equilibrium in games with mediators as in Bayesian machine games; we leave the details to the reader.

Up to now, we have considered only players communicating with a mediator. We certainly want to allow for the possibility of players communicating with each other. We model this using a particular mediator that we call the *communication mediator*, denoted comm, which corresponds to what cryptographers call *secure channels* and economists call *cheap talk*. With this mediator, if i wants to send a message to j, it simply sends the message and its intended recipient to the mediator comm. The mediator's strategy is simply to forward the messages, and the identities of the senders, to the intended recipients. (Technically, we assume that a message m from i to the mediator with intended recipient j has the form $m;j$. Messages not of this form are ignored by the mediator.)

3 A Game-Theoretic Notion of Protocol Security

In this section we extend the traditional notion of game-theoretic implementation of mediators to consider computational games. Our aim is to obtain a notion of implementation that can be used to capture the cryptographic notion of secure computation. For simplicity, we focus on implementations of mediators that receive a single message from each player and return a single message to each player (i.e., the mediated games consist only of a single stage).

We provide a definition that captures the intuition that the machine profile $\vec{M}$ implements a mediator $\mathcal{F}$ if, whenever a set of players want to to truthfully provide their "input" to the mediator

$\mathcal{F}$, they also want to run $\vec{M}$ using the same inputs. To formalize "whenever", we consider what we call *canonical coalition games*, where each player i has a type t_i of the form $x_i; z_i$, where x_i is player i's intended "input" and z_i consists of some additional information that player i has about the state of the world. We assume that the input x_i has some fixed length n. Such games are called *canonical games of input length n*.[7]

Let $\Lambda^{\mathcal{F}}$ denote the machine that, given type $t = x; z$ sends x to the mediator $\mathcal{F}$ and outputs as its action whatever string it receives back from $\mathcal{F}$, and then halts. (Technically, we model the fact that $\Lambda^{\mathcal{F}}$ is expecting to communicate with $\mathcal{F}$ by assuming that the mediator $\mathcal{F}$ appends a signature to its messages, and any messages not signed by $\mathcal{F}$ are ignored by $\Lambda^{\mathcal{F}}$.) Thus, the machine $\Lambda^{\mathcal{F}}$ ignores the extra information z. Let $\vec{\Lambda}^{\mathcal{F}}$ denote the machine profile where each player uses the machine $\Lambda^{\mathcal{F}}$. Roughly speaking, to capture the fact that whenever the players want to use $\mathcal{F}$, they also want to run $\vec{M}$, we require that if $\vec{\Lambda}^{F}$ is an equilibrium in the game $(G, \mathcal{F})$ (i.e., if it is an equilibrium to simply provide the intended input to $\mathcal{F}$ and finally output whatever $\mathcal{F}$ replies), running $\vec{M}$ using the intended input is an equilibrium as well.

We actually consider a more general notion of implementation: we are interested in understanding how well equilibrium in a set of games with mediator $\mathcal{F}$ can be implemented using a machine profile $\vec{M}$ and a possibly different mediator $\mathcal{F}'$. Roughly speaking, we want that, for every game G in some set $\mathcal{G}$ of games, if $\vec{\Lambda}^{\mathcal{F}}$ is an equilibrium in $(G, \mathcal{F})$, then $\vec{M}$ is an equilibrium in $(G, \mathcal{F}')$. In particular, we want to understand what degree of robustness p in the game $(G, \mathcal{F})$ is required to achieve an ϵ-equilibrium in the game $(G, \mathcal{F}')$. We also require that the equilibrium with mediator $\mathcal{F}'$ be as "coalition-safe" as the equilibrium with mediator $\mathcal{F}$.

Definition 3.1 (Universal implementation) *Suppose that $\mathcal{G}$ is a set of m-player canonical games, $\mathcal{Z}$ is a set of subsets of $[m]$, $\mathcal{F}$ and $\mathcal{F}'$ are mediators, $M_1, \ldots, M_m$ are interactive machines, $p : I\!N \times I\!N \rightarrow I\!N$, and $\epsilon : I\!N \rightarrow I\!R$. $(\vec{M}, \mathcal{F}')$ is a $(\mathcal{G}, \mathcal{Z}, p)$-*universal implementation of $\mathcal{F}$ with error ϵ *if, for all $n \in I\!N$, all games $G \in \mathcal{G}$ with input length n, and all $\mathcal{Z}' \subseteq \mathcal{Z}$, if $\vec{\Lambda}^{\mathcal{F}}$ is a $p(n, \cdot)$-robust $\mathcal{Z}'$-safe Nash equilibrium in the mediated machine game $(G, \mathcal{F})$ then*

1. (Preserving Equilibrium) *$\vec{M}$ is a $\mathcal{Z}'$-safe $\epsilon(n)$-Nash equilibrium in the mediated machine game $(G, \mathcal{F}')$.*
2. (Preserving Action Distributions) *For each type profile $\vec{t}$, the action profile induced by $\vec{\Lambda}^{\mathcal{F}}$ in $(G, \mathcal{F})$ is identically distributed to the action profile induced by $\vec{M}$ in $(G, \mathcal{F}')$.*

As we have observed, although our notion of universal implementation does not explicitly consider the privacy of players' inputs, it can nevertheless capture privacy requirements. It suffices to consider a game where a player gains significant utility by knowing some information about a player's input.

Note that, depending on the class $\mathcal{G}$, our notion of universal implementation imposes severe restrictions on the complexity of the machine profile $\vec{M}$. For instance, if $\mathcal{G}$ consists of all games, it requires that the complexity of $\vec{M}$ is the same as the complexity of $\vec{\Lambda}^{\mathcal{F}}$. (If the complexity of $\vec{M}$ is higher than that of $\vec{\Lambda}^{\mathcal{F}}$, then we can easily construct a game G by choosing the utilities appropriately such that it is an equilibrium to run $\vec{\Lambda}^{\mathcal{F}}$ in $(G, \mathcal{F})$, but running $\vec{M}$ is too costly.) Also note that if $\mathcal{G}$ consists of games where players strictly prefer smaller complexity, then universal implementation requires that $\vec{M}$ be the optimal algorithm (i.e., the algorithm with the lowest complexity) that implements the functionality of $\vec{M}$, since otherwise a player would prefer to switch to the optimal implementation. Since few algorithms algorithms have been shown to be provably optimal with

[7] Note that by simple padding, canonical games represent a setting where all parties' input lengths are upper-bounded by some value n that is common knowledge. Thus, we can represent any game where there are only finitely many possible types as a canonical game for some input length n.

respect to, for example, the number of computational steps of a Turing machines, this, at first sight, seems to severely limit the use of our definition. However, if we consider games with "coarse" complexity functions where, say, the first T steps are "free" (e.g., machines that execute less than T steps are assigned complexity 1), or n^2 computational steps count as one unit of complexity, the restrictions above are not so severe. Indeed, it seems quite natural to assume that a player is indifferent between "small" differences in computation in such a sense.

Our notion of universal implementation is related to a number of earlier notions of implementation. We now provide a brief comparison with the most relevant ones.

- Our definition of universal implementation captures intuitions similar in spirit to Forges' [1990] notion of a *universal mechanism*. It differs in one obvious way: our definition considers *computational* games, where the utility functions depend on complexity considerations. Dodis, Halevi and Rabin [2000] (and more recent work, such as [Abraham, Dolev, Gonen, and Halpern 2006; Lepinski, Micali, Peikert, and Shelat 2004; Halpern and Teadgue 2004; Abraham, Dolev, Gonen, and Halpern 2006; Gordon and Katz 2006; Kol and Naor 2008]) consider notions of implementation where the players are modeled as polynomially-bounded Turing machines, but do not consider computational games. As such, the notions considered in these works do not provide any *a priori* guarantees about the incentives of players with regard to computation.

- Our definition is more general than earlier notions of implementation in that we consider universality with respect to (sub-)classes $\mathcal{G}$ of games, and allow deviations by coalitions.

- Our notion of coalition-safety also differs somewhat from earlier related notions. Note that if $\mathcal{Z}$ contains all subsets of players with k or less players, then universal implementation implies that all k-resilient Nash equilibria and all strong k-resilient Nash equilibria are preserved. However, unlike the notion of k-resilience considered by Abraham et al. [2006, 2008], our notion provides a "best-possible" guarantee for games that do not have a k-resilient Nash equilibrium. We guarantee that if a certain subset Z of players have no incentive to deviate in the mediated game, then that subset will not have incentive to deviate in the cheap-talk game; this is similar in spirit to the definitions of [Izmalkov, Lepinski, and Micali 2008; Lepinski, Micali, Peikert, and Shelat 2004]. Note that, in contrast to [Izmalkov, Lepinski, and Micali 2008; Lepinski, Micali, and Shelat 2005], rather than just allowing colluding players to communicate only through their moves in the game, we allow coalitions of players that are controlled by a single entity; this is equivalent to considering collusions where the colluding players are allowed to freely communicate with each other. In other words, whereas the definitions of [Izmalkov, Lepinski, and Micali 2008; Lepinski, Micali, and Shelat 2005] require protocols to be "signalling-free", our definition does not impose such restrictions. We believe that this model is better suited to capturing the security of cryptographic protocols in most traditional settings (where signalling is not an issue).

- We require only that a Nash equilibrium is preserved when moving from the game with mediator $\mathcal{F}$ to the communication game. Stronger notions of implementation require that the equilibrium in the communication game be a *sequential equilibrium* [Kreps and Wilson 1982]; see, for example, [Gerardi 2004; Ben-Porath 2003]. Since every Nash equilibrium in the game with the mediator $\mathcal{F}$ is also a sequential equilibrium, these stronger notions of implementation actually show that sequential equilibrium is preserved when passing from the game with the mediator to the communication game.

  While these notions of implementation guarantee that an equilibrium with the mediator is preserved in the communication game, they do not guarantee that new equilibria are not introduced in the latter. An even stronger guarantee is provided by Izmalkov, Lepinski, and Micali's [2008] notion of *perfect implementation*; this notion requires a one-to-one correspondence f between *strategies* in the corresponding games such that each player's utility with strategy profile $\vec{\sigma}$ in the game with the mediator is the same as his utility with strategy profile $(f(\sigma_1), \ldots, f(\sigma_n))$ in the communication game without the mediator. Such a correspondence, called *strategic equivalence* by Izmalkov, Lepinski, and Micali [2008], guarantees (among other things) that *all* types of equilibria are preserved when passing from one game to the other, and that no new equilibria are introduced in the communication game. However, strategic equivalence can be achieved only with the use of strong primitives, which cannot be implemented under standard computational and systems assumptions [Lepinski, Micali, and Shelat 2005]. We focus on the simpler notion of implementation, which requires only that Nash equilibria are preserved, and leave open an exploration of more refined notions.

Strong Universal Implementation Intuitively, $(\vec{M}, \mathcal{F}')$ universally implements $\mathcal{F}$ if, whenever a set of parties want to use $\mathcal{F}$ (i.e., it is an equilibrium to use $\vec{\Lambda}^{\mathcal{F}}$ when playing with $\mathcal{F}$), then the parties also want to run $\vec{M}$ (using $\mathcal{F}'$) (i.e., using $\vec{M}$ with $\mathcal{F}'$ is also an equilibrium). We now strengthen this notion to also require that whenever a subset of the players do *not* want to use $\mathcal{F}$ (specifically, if they prefer to do "nothing"), then they also do not want to run $\vec{M}$, even if all other players do so. Recall that $\perp$ denotes the (canonical) machine that does nothing. We use $\perp$ to denote the special machine that sends no messages and writes nothing on the output tape.

Definition 3.2 (Strong Universal Implementation) *Let* $(\vec{M}, \mathcal{F}')$ *be a* $(\mathcal{G}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ. $(\vec{M}, \mathcal{F}')$ *is a* strong $(\mathcal{G}, \mathcal{Z}, p)$-implementation of $\mathcal{F}$ *if, for all* $n \in I\!N$,

all games $G \in \mathcal{G}$ *with input length* n*, and all* $Z \in \mathcal{Z}$*, if* $\vec{\perp}_Z$ *is a* $p(n, \cdot)$*-robust best response to* $\Lambda^{\mathcal{F}}_{-Z}$ *in* $(G, \mathcal{F})$*, then* $\vec{\perp}_Z$ *is an* ϵ*-best response to* $\vec{M}_{-Z}$ *in* $(G, \mathcal{F}')$.

4 Relating Cryptographic and Game-Theoretic Implementation

We briefly recall the notion of *precise secure computation* [Micali and Pass 2006; Micali and Pass 2007], which is a strengthening of the traditional notion of secure computation [Goldreich, Micali, and Wigderson 1987]; more details are given in Appendix A.

An m*-ary functionality* f is specified by a random process that maps vectors of inputs to vectors of outputs (one input and one output for each player). That is, $f : ((\{0,1\}^*)^m \times \{0,1\}^\infty) \rightarrow (\{0,1\}^*)^m$, where we view f_i as the ith component of the output vector; that is, $f = (f_1, \ldots, f_m)$. We often abuse notation and suppress the random bitstring r, writing $f(\vec{x})$ or $f_i(\vec{x})$. (We can think of $f(\vec{x})$ and $f_i(\vec{x})$ as random variables.) A mediator $\mathcal{F}$ (resp., a machine profile $\vec{M}$) *computes* f if, for all $n \in N$, all inputs $\vec{x} \in (\{0,1\}^n)^m$, if the players tell the mediator their inputs and output what the mediator $\mathcal{F}$ tells them (resp., the output vector of the players after an execution of $\vec{M}$ where M_i gets input x_i) is identically distributed to $f^n(\vec{x})$. Roughly speaking, a protocol $\vec{M}$ for computing a function f is *secure* if, for every adversary A participating in the *real execution* of $\vec{M}$, there exists a "simulator" $\tilde{A}$ participating in an *ideal execution* where all players directly talk to a trusted third party (i.e., a mediator) computing f; the job of $\tilde{A}$ is to provide appropriate inputs to the trusted party, and to reconstruct the view of A in the real execution such that no *distinguisher* D can distinguish the outputs of the parties and the view of the adversary A in the real and the ideal execution. (Note that in the real execution the view of the adversary is simply the actual view of A in the execution, whereas in the ideal execution it is the view output by the simulator $\tilde{A}$). The traditional notion of secure computation [Goldreich, Micali, and Wigderson 1987] requires only that the *worst-case* complexity (size and running-time) of $\tilde{A}$ is polynomially related to that of A. Precise secure computation [Micali and Pass 2006; Micali and Pass 2007] additionally requires that the running time of the simulator $\tilde{A}$ "respects" the running time of the adversary A in an "execution-by-execution" fashion: a secure computation is said to have precision $p(n,t)$ if the running-time of the simulator $\tilde{A}$ (on input security parameter n) is bounded by $p(n,t)$ whenever $\tilde{A}$ outputs a view in which the running-time of A is t.

We introduce here a weakening of the notion of precise secure computation. The formal definition is given in Appendix A.1. We here highlight the key differences:

- The standard definition requires the existence of a simulator for every A, such that the real and the ideal execution cannot be distinguished given any set of inputs and any distinguisher. In analogy with the work of Dwork, Naor, Reingold, and Stockmeyer [2003], we change the order of the quantifiers. We simply require that given any adversary, any input distribution and any distinguisher, there exists a simulator that tricks that particular distinguisher, except with probability $\epsilon(n)$; ϵ is called the error of the secure computation.
- The notion of precise simulation requires that the simulator *never* exceeds its precision bounds. We here relax this assumption and let the simulator exceed its bound with probability $\epsilon(n)$.

We also generalize the notion by allowing arbitrary complexity measures $\vec{\mathscr{C}}$ (instead of just running-time) and *general adversary structures* [Hirt and Maurer 2000] (where the specification of a secure computation includes a set $\mathcal{Z}$ of subsets of players such that the adversary is allowed to corrupt only the players in one of the subsets in $\mathcal{Z}$; in contrast, in [Goldreich, Micali, and Wigderson 1987; Micali and Pass 2006] only *threshold adversaries* are considered, where $\mathcal{Z}$ consists of all subsets up to a pre-specified size k). The formal definition of *weak $\vec{\mathscr{C}}$-precise secure computation* is given in Appendix A.1. Note that the we can always regain the "non-precise" notion of secure computation by instantiating $\mathscr{C}_Z(M,v)$ with the sum of the *worst-case* running-time of M (on inputs of the same length as the input length in v) and size of M. Thus, by the results of [Ben-Or, Goldwasser, and Wigderson 1988; Goldwasser, Micali, and Rackoff 1989; Goldreich, Micali, and Wigderson 1987], it follows that there exists weak $\vec{\mathscr{C}}$-precise secure computation protocols with precision $p(n,t) = poly(n,t)$ when $\mathscr{C}_Z(M,v)$ is the sum of the worst-case running-time of M and size of M. The results of [Micali and Pass 2006; Micali and Pass 2007] extend to show the existence of weak $\mathscr{C}$-precise secure computation protocols with precision $p(n,t) = O(t)$ when $\mathscr{C}_Z(M,v)$ is the sum of the running time (as opposed to just worst-case running-time) of $M(v)$ and size of M. The results above continue to hold if we consider "coarse" measures of running-time and size; for instance, if, say, n^2 computational steps correspond to one unit of complexity (in canonical machine games with input length n). See Appendix 4.2 for more details.

4.1 Equivalences: The Information-theoretic Case

As a warm-up, we show that "error-free" secure computation, also known as *perfectly-secure computation* [Ben-Or, Goldwasser, and Wigderson 1988], already implies the traditional game-theoretic notion of implementation [Forges 1990] (which does not consider computation). To do this, we first formalize the traditional game-theoretic notion using our notation: Let $\vec{M}$ be an m-player profile of machines. We say that $(\vec{M}, \mathcal{F}')$ is a *traditional game-theoretic implementation* of $\mathcal{F}$ if $(\vec{M}, \mathcal{F}')$ is a $(\mathcal{G}^{\mathsf{nocomp}}, \{\{1\}, \dots \{m\}\}, 0)$-universal implementation of $\mathcal{F}$ with 0-error, where $\mathcal{G}^{\mathsf{nocomp}}$ denotes the class of all m-player canonical machine games where the utility functions do not depend on the complexity profile. (Recall that the traditional notion does not consider computational games or coalition games.)

Proposition 4.1 *If f is an m-ary functionality, $\mathcal{F}$ is a mediator that computes f, and $\vec{M}$ is a perfectly-secure computation of $\mathcal{F}$, then $(\vec{M}, \mathsf{comm})$ is a game-theoretic implementation of $\mathcal{F}$.*

Proof: We start by showing that running $\vec{M}$ is a Nash equilibrium if running $\vec{\Lambda}^{\mathcal{F}}$ with mediator $\mathcal{F}$ is one. Recall that the cryptographic notion of error-free secure computation requires that for every player i and every "adversarial" machine M'_i controlling player i, there exists a "simulator" machine $\tilde{M}_i$, such that the outputs of all players in the execution of $(M'_i, \vec{M}_{-i})$ are *identically distributed* to the outputs of the players in the execution of $(\tilde{M}_i, \vec{\Lambda}^{\mathcal{F}}_{-i})$ with mediator $\mathcal{F}$.[8] In game-theoretic terms, this means that every "deviating" strategy M'_i in the communication game can be mapped into a deviating strategy $\tilde{M}_i$ in the mediated game with the same output distribution for each type, and, hence, the same utility, since the utility depends only on the type and the output distribution; this follows since we require universality only with respect to games in $\mathcal{G}^{\mathsf{nocomp}}$. Since no deviations in the mediated game can give higher utility than the Nash equilibrium strategy of using $\Lambda^{\mathcal{F}}_i$, running $\vec{M}$ must also be a Nash equilibrium.

It only remains to show that $\vec{M}$ and $\vec{\Lambda}^{\mathcal{F}}$ induce the same action distribution; this follows directly from the definition of secure computation by considering an adversary that does not corrupt any parties.

∎

We note that the converse implication does not hold. Since the traditional game-theoretic notion of implementation does not consider computational cost, it does not take into account computational advantages possibly gained by using $\vec{M}$, issues that are critical in the cryptographic notion of zero-knowledge simulation.

We now show that weak precise secure computation is equivalent to strong $\mathcal{G}$-universal implementation for certain natural classes $\mathcal{G}$ of games. For this result, we assume that the only machines that can have a complexity of 0 are those that "do nothing": we require that, for all complexity functions $\mathscr{C}$, $\mathscr{C}(M, v) = 0$ for some view v iff $M = \perp$ iff $\mathscr{C}(M, v) = 0$ for all views v. (Recall that $\perp$ is a canonical representation of the TM that does nothing: it does not read its input, has no state changes, and writes nothing.) If $G = ([m], \mathcal{M}, \Pr, \vec{\mathscr{C}}, \vec{u})$ is a canonical game with input length n, then

1. G is *machine universal* if the machine set $\mathcal{M}$ is the set of terminating Turing machines;

2. G is *normalized* if the range of u_Z is $[0, 1]$ for all subsets Z of $[m]$;

3. G is *monotone* (i.e., "players never prefer to compute more") if, for all subset Z of $[m]$, all type profiles $\vec{t}$, action profiles $\vec{a}$, and all complexity profiles $(c_Z, \vec{c}_{-Z})$, $(c'_Z, \vec{c}_{-Z})$, if $c'_Z > c_Z$, then $u_Z(\vec{t}, \vec{a}, (c'_Z, \vec{c}_{-Z})) \leq u_i(\vec{t}, \vec{a}, (c_Z, \vec{c}_{-Z}))$;

4. G is a $\vec{\mathscr{C}}'$-game if $\mathscr{C}_Z = \mathscr{C}'_Z$ for all subsets Z of $[m]$.

Let $\mathcal{G}^{\vec{\mathscr{C}}}$ denote the class of machine-universal, normalized, monotone, canonical $\vec{\mathscr{C}}$-games. For our theorem we need some minimal constraints on the complexity function. For the forward direction of our equivalence results (showing that precise secure computation implies universal implementation), we require that honestly running the protocol should have constant complexity, and that it be the same with and without a mediator. More precisely, we assume that the complexity profile $\vec{\mathscr{C}}$ is $\vec{M}$*-acceptable*, that is, for every subset Z, the machines $(\Lambda^{\mathcal{F}})^b_Z$ and M^b_Z have the same complexity c_0 for all inputs; that is, $\mathscr{C}_Z((\Lambda^{\mathcal{F}})^b_Z, \cdot) = c_0$ and $\mathscr{C}_Z(M^b_Z, \cdot) = c_0$.[9] Note that an assumption of this nature is necessary in order to show that $(\vec{M}, \mathsf{comm})$ is a $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$. If the complexity of $\vec{M}$ is higher than that of $\vec{\Lambda}^{\mathcal{F}}$, then we can construct a game G such that it is an equilibrium to run $\vec{\Lambda}^{\mathcal{F}}$ in $(G, \mathcal{F})$, but running $\vec{M}$ is too costly. The assumption that $\vec{M}$

[8] The follows from the fact that perfectly-secure computation is error-free.

[9] Our results continue to hold if c_0 is a function of the input length n, but otherwise does not depend on the view.

and $\vec{\Lambda}^{\mathcal{F}}$ have the same complexity is easily satisfied when considering coarse complexity function (where say the first T steps of computation are free). Another way of satisfying this assumption is to consider a complexity function that simply charges c_0 for the use of the mediator, where c_0 is the complexity of running the protocol. Given this view, universal implementation requires only that players want to run $\vec{M}$ as long as they are willing to pay c_0 in complexity for talking to the mediator. For the backward direction of our equivalence (showing that universal implementation implies precise secure computation), we require that certain operations, like moving output from one tape to another, do not incur any additional complexity. Such complexity functions are called *output-invariant*; we provide a formal definition at the beginning of Appendix B.

We can now state the connection between secure computation and game-theoretic implementation. In the forward direction, we restrict attention to protocols $\vec{M}$ computing some m-ary functionality f that satisfy the following natural property: if a subset of the players "aborts" (not sending any messages, and outputting nothing), their input is intepreted as λ.[10] More precisely, $\vec{M}$ is an *abort-preserving* computation of f if for all $n \in N$, every subset Z of $[m]$, all inputs $\vec{x} \in (\{0,1\}^n)^m$, the output vector of the players after an execution of $(\vec{\bot}_Z, \vec{M}_{-Z})$ on input $\vec{x}$ is identically distributed to $f(\lambda_Z, \vec{x}_{-Z})$.

Theorem 4.2 (Equivalence: Information-theoretic case) *Suppose that f is an m-ary functionality, $\mathcal{F}$ is a mediator that computes f, $\vec{M}$ is a machine profile that computes f, $\mathcal{Z}$ is a set of subsets of $[m]$, $\vec{\mathscr{C}}$ is a complexity function, and p a precision function.*

- *If $\vec{\mathscr{C}}$ is $\vec{M}$-acceptable and $\vec{M}$ is an abort-preserving weak $\mathcal{Z}$-secure computation of f with $\vec{\mathscr{C}}$-precision p and ϵ-statistical error, then $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ.*
- *If $\vec{\mathscr{C}}$ is $\vec{M}$-acceptable and output-invariant, and $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ', then for every $\epsilon < \epsilon'$, $\vec{M}$ is a weak $\mathcal{Z}$-secure computation of f with $\vec{\mathscr{C}}$-precision p and ϵ-statistical error.*

As a corollary of Theorem 4.2, we get that known (precise) secure computation protocols directly yield appropriate universal implementations, provided that we consider complexity functions that are $\vec{M}$-acceptable. For instance, by the results of [Ben-Or, Goldwasser, and Wigderson 1988; Micali and Pass 2007], every efficient m-ary functionality f has a weak $\mathcal{Z}$-secure computation protocol $\vec{M}$ with $\mathscr{C}$-precision $p(n,t) = t$ if $\mathscr{C}_Z(M, v)$ is the sum of the running time of $M(v)$ and size of M, and $\mathcal{Z}$ consists of all subsets of $[m]$ of size smaller than $|m|/3$. This result still holds if we consider "coarse" measures of running-time and size where, say, $O(n^c)$ computational steps (and size) correspond to one unit of complexity (in canonical machine games with input length n). Furthermore, protocol $\vec{M}$ is abort-preserving, has a constant description, and has running time smaller than some fixed polynomial $O(n^c)$ (on inputs of length n). So, if we consider an appropriately coarse notion of running time and description size, $\mathscr{C}_Z$ is $\vec{M}$-acceptable. By Theorem 4.2, it then immediately follows that every efficient m-ary functionality f has a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, O(1))$-universal implementation with error 0.

Theorem 4.2 also shows that a universal implementation of a mediator $\mathcal{F}$ computing a function f with respect to *general* classes of games is "essentially" as hard to achieve as a secure computations of f. In particular, as long as the complexity function is output-invariant, such a universal implementation is a weak precise secure computation. Although the output-invariant condition might seem somewhat artificial, Theorem 4.2 illustrates that overcoming the "secure-computation

[10] All natural secure computation protocols that we are aware of (e.g., [Goldreich, Micali, and Wigderson 1987; Ben-Or, Goldwasser, and Wigderson 1988]) satisfy this property.

barrier" with respect to general classes of games requires making strong (and arguably unnatural[11]) assumptions about the complexity function. We have not pursued this path. In Section 6, we instead consider universal implementation with respect to restricted class of games. As we shall see, this provides an avenue for circumventing traditional impossibility results with respect to secure computation.

In Section 4.2 we also provide a "computational" analogue of the equivalence theorem above, as well as a characterization of the "standard" (i.e., "non-precise") notion of secure computation.

Proof overview We now provide a high-level overview of the proof of Theorem 4.2. Needless to say, this oversimplified sketch leaves out many crucial details that complicate the proof.

Weak precise secure computation implies strong universal implementation. At first glance, it might seem like the traditional notion of secure computation of [Goldreich, Micali, and Wigderson 1987] easily implies the notion of universal implementation: if there exists some (deviating) strategy A in the communication game implementing mediator $\mathcal{F}$ that results in a different distribution over actions than in equilibrium, then the simulator $\tilde{A}$ for A could be used to obtain the same distribution; moreover, the running time of the simulator is within a polynomial of that of A. Thus, it would seem like secure computation implies that any "poly"-robust equilibrium can be implemented. However, the utility function in the game considers the complexity of each execution of the computation. So, even if the worst-case running time of $\tilde{A}$ is polynomially related to that of A, the utility of corresponding executions might be quite different. This difference may have a significant effect on the equilibrium. To make the argument go through we need a simulation that preserves complexity in an execution-by-execution manner. This is exactly what precise zero knowledge [Micali and Pass 2006] does. Thus, intuitively, the degradation in computational robustness by a universal implementation corresponds to the precision of a secure computation.

More precisely, to show that a machine profile $\vec{M}$ is a universal implementation, we need to show that whenever Λ is a p-robust equilibrium in a game G with mediator $\mathcal{F}$, then $\vec{M}$ is an ϵ-equilibrium (with the communication mediator comm). Our proof proceeds by contradiction: we show that a deviating strategy M'_Z (for a coalition Z) for the ϵ-equilibrium $\vec{M}$ can be turned into a deviating strategy $\tilde{M}_Z$ for the p-robust equilibrium $\vec{\Lambda}$. We here use the fact that $\vec{M}$ is a weak precise secure computation to find the machine $\tilde{M}_Z$; intuitively $\tilde{M}_Z$ will be the simulator for M'_Z. The key step in the proof is a method for embedding any coalition machine game G into a distinguisher D that "emulates" the role of the utility function in G. If done appropriately, this ensures that the utility of the (simulator) strategy $\tilde{M}_Z$ is close to the utility of the strategy M'_Z, which contradicts the assumption that $\vec{\Lambda}$ is an ϵ-Nash equilibrium.

The main obstacle in embedding the utility function of G into a distinguisher D is that the utility of a machine $\tilde{M}_Z$ in G depends not only on the types and actions of the players, but also on the complexity of running $\tilde{M}_Z$. In contrast, the distinguisher D does not get the complexity of $\tilde{M}$ as input (although it gets its output v). On a high level (and oversimplifying), to get around this problem, we let D compute the utility *assuming* (incorrectly) that $\tilde{M}_Z$ has complexity $c = \mathscr{C}(M', v)$ (i.e., the complexity of M'_Z in the view v output by $\tilde{M}_Z$). Suppose, for simplicity, that $\tilde{M}_Z$ is *always* "precise" (i.e., it always respects the complexity bounds).[12] Then it follows that (since the complexity c is always close to the actual complexity of $\tilde{M}_Z$ in every execution) the utility computed by D corresponds to the utility of some game $\tilde{G}$ that is at most a p-speed up of G. (To ensure that $\tilde{G}$ is indeed a speedup and not a "slow-down", we need to take special care with

[11] With a coarse complexity measure, it seems natural to assume that moving content from one output tape to another incurrs no change in complexity.

[12] This is an unjustified assumption; in the actual proof we actually need to consider a more complicated construction.

simulators that potentially run faster than the adversary they are simulating. The monotonicity of G helps us to circumvent this problem.) Thus, although we are not able to embed G into the distinguisher D, we can embed a related game $\tilde{G}$ into D. This suffices to show that $\vec{\Lambda}$ is not a Nash equilibrium in $\tilde{G}$, contradicting the assumption that $\vec{\Lambda}$ is a p-robust Nash equilibrium. A similar argument can be used to show that $\perp$ is also an ϵ-best response to $\vec{M}_{-Z}$ if $\perp$ is a p-robust best response to $\vec{\Lambda}_{-Z}$, demonstrating that $\vec{M}$ in fact is a strong universal implementation. We here rely on the fact $\vec{M}$ is abort-preserving to ensure that aborting in $(G, \mathcal{F})$ has the same effect as in (G, comm).

Strong universal implementation implies weak precise secure computation. To show that strong universal implementation implies weak precise secure computation, we again proceed by contradiction. We show how the existence of a distinguisher D and an adversary M'_Z that cannot be simulated by *any* machine $\tilde{M}_Z$ can be used to construct a game G for which $\vec{M}$ is not a strong implementation. The idea is to have a utility function that assigns high utility to some "simple" strategy M^*_Z. In the mediated game with $\mathcal{F}$, no strategy can get better utility than M^*_Z. On the other hand, in the cheap-talk game, the strategy M'_Z does get higher utility than M^*_Z. As D indeed is a function that "distinguishes" a mediated execution from a cheap-talk game, our approach will be to try to embed the distinguisher D into the game G. The choice of G depends on whether $M'_Z = \perp$. We now briefly describe these games.

If $M'_Z = \perp$, then there is no simulator for the machine $\perp$ that simply halts. In this case, we construct a game G where using $\perp$ results in a utility that is determined by running the distinguisher. (Note that $\perp$ can be easily identified, since it is the only strategy that has complexity 0.) All other strategies instead get some canonical utility d, which is higher than the utility of $\perp$ in the mediated game. However, since $\perp$ cannot be "simulated", playing $\perp$ in the cheap-talk game leads to an even higher utility, contradicting the assumption that $\vec{M}$ is a universal implementation.

If $M'_Z \neq \perp$, we construct a game G' in which each strategy other than $\perp$ gets a utility that is determined by running the distinguisher. Intuitively, efficient strategies (i.e., strategies that have relatively low complexity compared to M'_Z) that output views on which the distinguisher outputs 1 with high probability get high utility. On the other hand, $\perp$ gets a utility d that is at least as good as what the other strategies can get in the mediated game with $\mathcal{F}$. This makes $\perp$ a best response in the mediated game; in fact, we can define the game G' so that it is actually a p-robust best response. However, it is not even an ϵ-best-response in the cheap-talk game: M'_Z gets higher utility, as it receives a view that cannot be simulated. (The output-invariant condition on the complexity function $\mathscr{C}$ is used to argue that M'_Z can output its view at no cost.) ∎

4.2 Equivalences: The Computational Case

To prove a "computational" analogue of our equivalence theorem (relating computational precise secure computation and universal implementation), we need to introduce some further restrictions on the complexity functions, and the classes of games considered.

- A (vector of) complexity functions $\vec{\mathscr{C}}$ is *efficient* if each function is computable by a (randomized) polynomial-sized circuit.
- A secure computation game $G = ([m], \mathcal{M}^{T(n)}, \Pr, \vec{\mathscr{C}}, \vec{u})$ with input length n is said to be $T(\cdot)$-*machine universal* if
  - the machine set $\mathcal{M}^{T(n)}$ is the set of Turing machines implementable by $T(n)$-sized randomized circuits, and
  - $\vec{u}$ is computable by a $T(n)$-sized circuit.

Let $\mathcal{G}^{\vec{\mathscr{C}},T}$ denote the class of $T(\cdot)$-machine universal, normalized, monotone, canonical $\vec{\mathscr{C}}$-games. Let $\mathcal{G}^{\vec{\mathscr{C}},\mathrm{poly}}$ denote the union of $\mathcal{G}^{\vec{\mathscr{C}},T}$ for all polynomial functions T.

Theorem 4.3 (Equivalence: Computational Case) *Suppose that f is an m-ary functionality, $\mathcal{F}$ is a mediator that computes f, $\vec{M}$ is a machine profile that computes f, $\mathcal{Z}$ is a set of subsets of $[m]$, $\vec{\mathscr{C}}$ is an efficient complexity function, and p a precision function.*

- *If $\vec{\mathscr{C}}$ is $\vec{M}$-acceptable and $\vec{M}$ is an abort-preserving weak $\mathcal{Z}$-secure computation of f with computational $\vec{\mathscr{C}}$-precision p, then for every polynomial T, there exists some negligible function ϵ such that $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}},T}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ.*
- *If $\vec{\mathscr{C}}$ is $\vec{M}$-acceptable and output-invariant, and for every polynomial T, there exists some negligible function ϵ, such that $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ, then $\vec{M}$ is a weak $\mathcal{Z}$-secure computation of f with computational $\vec{\mathscr{C}}$-precision p.*

Theorem 4.3 is proved in Appendix C

Relating Universal Implementation and "Standard" Secure Computation Note that Theorem 4.3 also provides a game-theoretic characterization of the "standard" (i.e., "non-precise") notion of secure computation. We simply consider a "coarse" version of the complexity function $\mathsf{wc}(v)$ that is the sum of the size of M and the worst-case running-time of M on inputs of the same length as in the view v. (We need a coarse complexity function to ensure that $\mathscr{C}$ is $\vec{M}$-acceptable and output-invariant.) With this complexity function, the definition of weak precise secure computation reduces to the traditional notion of weak secure computation without precision (or, more precisely, with "worst-case" precision just as in the traditional definition). Given this complexity function, the precision of a secure computation protocol becomes the traditional "overhead" of the simulator (this is also called *knowledge tightness* [Goldreich, Micali, and Wigderson 1991]). Roughly speaking, "weak secure computation" with overhead p is thus equivalent to strong $(\mathcal{G}^{\vec{\mathsf{wc}},\mathrm{poly}}, p)$-universal implementation with negligible error.

5 Universal Implementation for Specific Classes of Games

Our equivalence result for secure computation might seem like a negative result. It demonstrates that considering only rational players (as opposed to adversarial players) does not facilitate protocol design. Note, however, that for the equivalence to hold, we must consider implementations universal with respect to essentially *all* games. In many settings, it might be reasonable to consider implementations universal with respect to only certain subclasses of games; in such scenarios, universal implementations may be significantly simpler or more efficient, and may also circumvent traditional lower bounds. We list some natural restrictions on classes of games below, and discuss how such restrictions can be leveraged in protocol design. These examples illustrate some of the benefits of a fully game-theoretic notion of security that does not rely on the standard cryptographic simulation paradigm, and shows how our framework can capture in a natural way a number of natural notions of security.

To relate our notions to the standard definition of secure computation, we here focus on classes of games $\mathcal{G}$ that are subsets of $G^{\vec{\mathsf{wc}},poly}$ (as defined in Section 4.2). Furthermore, we consider only 2-player games and restrict attention to games G where the utility function is *separable* in the following sense: there is a standard game G' (where computational costs are not taken into account) and a function u_i^C on complexity profiles for each player i, such that, for each player i,

$u_i(\vec{t},\vec{a},\vec{c}) = u_i^{G'}(\vec{t},\vec{a}) + u_i^C(\vec{c})$. We refer to G' as the standard game *embedded* in G. Intuitively, this says that the utilities in G are just the sum of the utility in a game G' where computation is not taken into account and a term that depends only on computation costs.

Games with punishment: Many natural situations can be described as games where players can choose actions that "punish" an individual player i. For instance, this punishment can represent the cost of being excluded from future interactions. Intuitively, games with punishment model situations where players do not want to be caught cheating. Punishment strategies (such as the *grim-trigger* strategy in repeated prisoner's dilemma, where a player defects forever once his opponent defects once [Axelrod 1984]) are extensively used in the game-theory literature. We give two examples where cryptographic protocol design is facilitated when requiring only implementations that are universal with respect to games with punishment.

Covert adversaries: As observed by Malkhi et al. [2004], and more recently formalized by Aumann and Lindell [2006] [AL from now on], in situations were players do not want to be caught cheating, it is easier to construct efficient protocols. Using our framework, we can formalize this intuition in a straightforward way. To explain the intuitions, we consider a particularly simple setting. Let $\mathcal{G}^{\mathsf{punish}}$ consist of normalized 2-player games G (with a standard game G' embedded in G), where (1) honestly reporting your input and outputting whatever the mediator replies (i.e., playing the strategy implemented by $\vec{\Lambda}$) is a Nash equilibrium in $(G', \mathcal{F})$ where both players are guaranteed to get utility 1/2 (not just in expectation, but even in the worst-case), and (2) there is a special string punish such that player $1-i$ receives payoff 0 in G if player i outputs the string punish. In this setting, we claim that any secure computation of a function f with respect to covert adversaries with deterrent 1/2 in the sense of [Aumann and Lindell 2006, Definition 3.3] is a $(\mathcal{G}^{\mathsf{punish}}, poly, \mathcal{Z})$- universal implementation of $\mathcal{F}$ with negligible error (where $\mathcal{Z} = \{\{1\},\{2\}\}$, and $\mathcal{F}$ is a mediator computing f). Roughly speaking, the AL definition follows the traditional definition of secure computation, but changes how the ideal-model trusted party operates. More precisely, the ideal-model adversary, $\tilde{A}$, is given two new special messages it can send to the trusted party: cheat and $\mathsf{corrupted}$. If it sends $\mathsf{corrupted}$, the trusted party simply outputs punish to both players—this amounts to the adversary admitting that it is a "cheater"; if it sends cheat, then with probability 1/2 the adversary gets to see the input of the honest player and to select the honest players' output (this models successful undetected cheating), and with probability 1/2 the trusted party outputs punish to both players (this models the event that cheating was detected). We rely on the proofs of Theorem B.2 and C.1 to show that this notion of security suffices to get a universal implementation. For any strategy M' in the cheap-talk game, we want to construct a strategy $\tilde{M}$ in the mediated game $(\tilde{G}, \mathcal{F})$ (where $\mathcal{F}$ computes f, and $\tilde{G}$ is a poly-speedup of G) with roughly the same utility (formally, with utility negligibly close); if we can do this, the rest of the proof follows as in Proposition B.2.

We first show how to do this for a different mediated game $(\hat{G}, \hat{\mathcal{F}})$, where $\hat{G}$ is a poly-speedup of G and $\hat{\mathcal{F}}$ is a variant of $\mathcal{F}$ that considers the extra special operations considered in the AL trusted-party definition. In this game, we can simply use the simulator $\hat{M}$ for M', letting $\hat{G}$ be a speedup of G that takes care of the overhead in complexity of $\hat{M}$ with respect to M'; this can be done just as in the proof of Theorem B.2, but is much simpler as we here only consider worst-case complexity. (Recall that a game $\hat{G}$ is a speedup of G if $\hat{G}$ and G are identical except for the complexity profiles and the machine set, and where the complexity profile in $\hat{G}$ is a speedup of that in G.) Next, we convert $\hat{M}$ into a machine $\tilde{M}$ that *never* outputs any of the special messages $\mathsf{corrupt}$ and cheat. $\tilde{M}$ simply runs $\hat{M}$; if at any point $\hat{M}$ attempts to output a special message, $\tilde{M}$ instead (honestly) outputs the input of the corrupted player (and finally outputs whatever the mediator replies). We first claim that $\tilde{M}$ gets at least as high utility as $\hat{M}$ in the standard game G'. This follows since honestly outputting the true input gives a utility of $1/2$ (by definition); on the other hand, outputting $\mathsf{corrupt}$ gives a payoff of 0, and outputting cheat gives a payoff of at most $1/2 \times 1 + 1/2 \times 0$. Furthemore, the overhead in complexity of $\tilde{M}$ with respect to $\hat{M}$ is at most polynomial; thus we can construct a game $\tilde{G}$ that is at most a poly-speedup of $\hat{G}$ (and thus also of G) such that the utility of $\tilde{M}$ in $(\tilde{G}, \mathcal{F})$ is at least that of $\hat{M}$ in $(\hat{G}, \hat{F})$. This concludes the proof.

Fairness: It is well-known that, for many functions, secure 2-player computation where both players receive output is impossible if we require *fairness* (i.e., that either both or neither of the players receives an output) [Goldreich 2004]. Such impossibility results can be easily circumvented by considering universal implementation with respect to games with punishment. This follows from the fact that although it is impossible to get secure computation with fairness, the weaker notion of *secure computation with abort* [Goldwasser and Lindell 2002] is achievable. Intuitively, this notion guarantees that the only attack possible is one where one of the players prevents the other player from getting its output; this is called an *abort*. This is formalized by adapting the trusted-party in the ideal model to allow the adversary to send a special abort message to the trusted party after seeing its own output, which blocks it from delivering an output to the honest party. To get a universal implementation with respect to games with punishment, it is sufficient to use any secure computation protocol with abort (see [Goldwasser and Lindell 2002; Micali and Pass 2007]) modified so that players output punish if the other player aborts. It immediately follows that a player can never get a higher utility by aborting (as this will be detected by the other player, and consequently the aborting player will be punished). Again, this is formalized by showing that for any ideal-model adversary that sends an abort message to the trusted party, there exists some other adversary (with essentially the same complexity) that simply does not send the abort message; this can only improve its utility (since aborting guarantees a utility of 0). This result can be viewed as a generalization of the approach of [Dodis, Halevi, and Rabin 2000].[13]

Strictly monotone games: In our equivalence results we considered monotone games, where players never prefer to compute more. It is sometimes reasonable to assume that players *strictly* prefer to compute less. We outline a few possible advantages of considering universal implementations with respect to strictly monotone games.

Gradual-release protocols: One vein of research on secure computation considers protocols for achieving fair exchanges using *gradual-release* protocols (see e.g., [Boneh and Naor 2000]). In a gradual-release protocol, the players are guaranteed that if at any point one player aborts, then the other player(s) can compute the output within a comparable amount of time (e.g., we can require that if a player aborts and can compute the answer in t time units, then all the other players should be able to compute it within $2t$ time units). We believe that by making appropriate assumptions about the utility of computation, we can ensure that players never have incentives to deviate. Consider, for instance, a two-player computation of a function f where in the last phase the players invoke a gradual exchange protocol such that if any player

[13] For this application, it is not necessary to use our game-theoretic definition of security. An alternative way to capture fairness in this setting would be to require security with respect to the standard (simulation-based) definition with abort, and additionally fairness (but not security) with respect to rational agents, according to the definition of [Dodis, Halevi, and Rabin 2000; Halpern and Teadgue 2004]; this approach is similar to the one used by Kol and Naor [2008]. Our formalization is arguably more natural, and also considers rational agents that "care" about computation.

aborts during the gradual exchange protocol, the other players attempts to recover the secret using a brute-force search. Intuitively, if for each player the cost of computing t extra steps is positive, even if the other player computes, say, $2t$ extra steps, it will never be worth it for a player to abort: by the security of the gradual-release protocol, an aborting player can only get its output twice as fast as the other player. Note that merely making assumptions about the cost of computing does not suffice to make this approach work; we also need to ensure that players prefer getting the output to not getting it, even if they can trick other players into computing for a long time. Otherwise, a player might prefer to abort and not compute anything, while the other player attempts to compute the output. We leave a full exploration of this approach for future work.

Error-free implementations: Unlike perfectly-secure protocols, computationally-secure protocols protocols inherently have a nonzero error probability. For instance, secure 2-player computation can be achieved only with computational security (with nonzero error probability). By our equivalence result, it follows that strong universal implementations with respect to the most general classes of 2-player games also require nonzero error probability. Considering universality with respect to only strictly monotone games gives an approach for achieving error-free implementations. This seems particularly promising if we consider an idealized model where cryptographic functionalities (such as one-way functions) are modeled as black boxes (see, e.g., the random oracle model of Bellare and Rogaway [1993]), and the complexity function considers the number of calls to the cryptographic function. Intuitively, if the computational cost of trying to break the cryptographic function is higher than the expected gain, it is not worth deviating from the protocol. We leave open an exploration of this topic. (A recent paper by Micali and Shelat [2009] relies on this idea, in combination with *physical* devices, to acheive error-free implementations in the context of secret sharing.)

Using computation as payment. Shoham and Tennenholtz [Shoham and Tennenholtz 2005] have investigated what functions f of two players' inputs x_1, x_2 can be computed by the players if they have access to a trusted party. The players are assumed to want to get the output $y = f(x_1, x_2)$, but each player i does not want to reveal more about his input x_i than what can be deduced from y. Furthermore, each player i prefers that other players do not get the output (although this is not as important as i getting the output and not revealing its input x_i). Interestingly, as Shoham and Tennenholtz point out, the simple binary function AND cannot be truthfully computed by two players, even if they have access to a trusted party. A player that has input 0 always knows the output y and thus does not gain anything from providing its true input to the trusted party: in fact, it always prefers to provide the input 1 in order to trick the other player.

We believe that for strictly monotone games this problem can be overcome by the use of cryptographic protocols. The idea is to construct a cryptographic protocol for computing AND where the players are required to solve a computational puzzle if they want to use 1 as input; if they use input 0 they are not required to solve the puzzle. The puzzle should have the property that it requires a reasonable amount of computational effort to solve. If this computational effort is more costly than the potential gain of tricking the other player to get the wrong output, then it is not worth it for a player to provide input 1 unless its input actually is 1. To make this work, we need to make sure the puzzle is "easy" enough to solve, so that a player with input 1 will actually want to solving the puzzle in order to get the correct output. We leave a full exploration of this idea for future work.

More generally, we believe that combining computational assumptions with assumptions about utility will be a fruitful line of research for secure computation. For instance, it is conceivable that difficulties associated with concurrent executability of protocols could be alleviated by making assumptions regarding the cost of message scheduling; the direction of Cohen, Kilian, and Petrank [2001] (where players who delay messages are themselves punished with delays) seems relevant in this regard.

6 Directions for Future Research

We have provided a game-theoretic definitin of protocol security, and shown a close connection between computationally robust Nash equilibria and precise secure computation. This opens the door to a number of exciting research directions in both secure computation and game theory. We describe a few here:

- Our notion of universal implementation uses Nash equilibrium as solution concept. It is well known that in (traditional) extensive form games (i.e., games defined by a game tree), a Nash equilibrium might prescribe non-optimal moves at game histories that do no occur on the equilibrium path. This can lead to "empty threats": "punishment" strategies that are non-optimal and thus not credible. Many recent works on implementation (see e.g., [Gerardi 2004; Izmalkov, Lepinski, and Micali 2008]) therefore focus on stronger solution concepts such as *sequential equilibrium* [Kreps and Wilson 1982]. We note that when taking computation into account, the distinction between credible and non-credible threats becomes more subtle: the threat of using a non-optimal strategy in a given history might be credible if, for instance, the overall complexity of the strategy is smaller than any strategy that is optimal at every history. Thus, a simple strategy that is non-optimal off the equilibrium path might be preferred to a more complicated (and thus more costly) strategy that performs better off the equilibrium path (indeed, people often use non-optimal but simple "rules-of-thumbs" when making decisions); see [Halpern and Pass 2008] for more details. Finding a good definition of empty threats in games with computation costs seems challenging.

- As we have seen, universal implementation is equivalent to a variant of precise secure computation with the order of quantification reversed. It would be interesting to find a notion of implementation that corresponds more closely to the standard definition, without a change in the order of quantifier; in particular, whereas the traditional definition of zero-knowledge guarantees *deniability* (i.e., the property that the interaction does not leave any "trace"), the new one does not. Finding a game-theoretic definition that also captures deniability seems like an interesting question.

- A natural next step would be to introduce notions of computation in the epistemic logic. There has already been some work in this direction (see, for example, [Halpern, Moses, and Tuttle 1988; Halpern, Moses, and Vardi 1994; Moses 1988]). We believe that combining the ideas of this paper with those of the earlier papers will allow us to get, for example, a cleaner knowledge-theoretic account of zero knowledge than that given by Halpern, Moses, and Tuttle [1988]. A first step in this direction is taken in [Halpern, Pass, and Raman 2009].

7 Acknowledgments

The second author wishes to thank Silvio Micali and abhi shelat for many exciting discussions about cryptography and game theory, and Ehud and Adam Kalai for enlightening discussions.

References

Abraham, I., D. Dolev, R. Gonen, and J. Y. Halpern (2006). Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proc. 25th ACM Symposium on Principles of Distributed Computing*, pp. 53–62.

Abraham, I., D. Dolev, and J. Y. Halpern (2008). Lower bounds on implementing robust and resilient mediators. In *Fifth Theory of Cryptography Conference*, pp. 302–319.

Aumann, R. J. (1959). Acceptable points in general cooperative n-person games. In A. W. Tucker and R. D. Luce (Eds.), *Contributions to the Theory of Games IV, Annals of Mathematical Studies 40*, pp. 287–324. Princeton, N. J.: Princeton University Press.

Aumann, Y. and Y. Lindell (2006). Security against covert adversaries: Efficient protocols for realistic adversaries. In *Proc. of Theory of Cryptography Conference*.

Axelrod, R. (1984). *The Evolution of Cooperation*. New York: Basic Books.

Bellare, M. and P. Rogaway (1993). Random oracles are practical: a paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pp. 62–73.

Ben-Or, M., S. Goldwasser, and A. Wigderson (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symposium on Theory of Computing*, pp. 1–10.

Ben-Porath, E. (2003). Cheap talk in games with incomplete information. *Journal of Economic Theory 108*(1), 45–71.

Ben-Sasson, E., A. Kalai, and E. Kalai (2007). An approach to bounded rationality. In *Advances in Neural Information Processing Systems 19 (Proc. of NIPS 2006)*, pp. 145–152.

Boneh, D. and M. Naor (2000). Timed commitments. In *Proc. CRYPTO 2000*, Lecture Notes in Computer Science, Volume 1880, pp. 236–254. Springer-Verlag.

Canetti, R. (2001). Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, pp. 136–145.

Cleve, R. (1986). Limits on the security of coin flips when half the processors are faulty. In *Proc. 18th ACM Symposium on Theory of Computing*, pp. 364–369.

Cohen, T., J. Kilian, and E. Petrank (2001). Responsive round complexity and concurrent zero-knowledge. In *Advances in Cryptology: ASIACRYPT 2001*, pp. 422–441.

Dodis, Y., S. Halevi, and T. Rabin (2000). A cryptographic solution to a game theoretic problem. In *CRYPTO 2000: 20th International Cryptology Conference*, pp. 112–130. Springer-Verlag.

Dodis, Y. and T. Rabin (2007). Cryptography and game theory. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*, Cambridge, U.K. Cambridge University Press.

Dwork, C., M. Naor, O. Reingold, and L. J. Stockmeyer (2003). Magic functions. *Journal of the ACM 50*(6), 852–921.

Forges, F. (1986). An approach to communication equilibria. *Econometrica 54*(6), 1375–85.

Forges, F. (1990). Universal mechanisms. *Econometrica 58*(6), 1341–64.

Gerardi, D. (2004). Unmediated communication in games with complete and incomplete information. *Journal of Economic Theory 114*, 104–131.

Goldreich, O. (2001). *Foundations of Cryptography, Vol. 1.* Cambridge University Press.

Goldreich, O. (2004). *Foundations of Cryptography, Vol. 2.* Cambridge University Press.

Goldreich, O., S. Micali, and A. Wigderson (1986). Proofs that yield nothing but their validity and a methodology of cryptographic design. In *Proc. 27th IEEE Symposium on Foundations of Computer Science.*

Goldreich, O., S. Micali, and A. Wigderson (1987). How to play any mental game. In *Proc. 19th ACM Symposium on Theory of Computing*, pp. 218–229.

Goldreich, O., S. Micali, and A. Wigderson (1991). Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proof systems. *Journal of the ACM 38*(1), 691–729.

Goldwasser, S. and Y. Lindell (2002). Secure computation without agreement. In *DISC '02: Proc. 16th International Conference on Distributed Computing*, pp. 17–32. Springer-Verlag.

Goldwasser, S., S. Micali, and C. Rackoff (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*(1), 186–208.

Gordon, D. and J. Katz (2006). Rational secret sharing, revisited. In *SCN (Security in Communication Networks) 2006*, pp. 229–241.

Halpern, J. Y., Y. Moses, and M. R. Tuttle (1988). A knowledge-based analysis of zero knowledge. In *Proc. 20th ACM Symposium on Theory of Computing*, pp. 132–147.

Halpern, J. Y., Y. Moses, and M. Y. Vardi (1994). Algorithmic knowledge. In *Theoretical Aspects of Reasoning about Knowledge: Proc. Fifth Conference*, pp. 255–266.

Halpern, J. Y. and R. Pass (2008). Algorithmic rationality: Game theory with costly computation. Unpublished manuscript.

Halpern, J. Y. and R. Pass (2009). A computational game-theoretic framework for cryptography. unpublished manuscript.

Halpern, J. Y., R. Pass, and V. Raman (2009). An epistemic characterization of zero-knowledge. In *Theoretical Aspects of Rationality and Knowledge: Proc. Twelth Conference (TARK 2009)*, pp. 156–165.

Halpern, J. Y. and V. Teadgue (2004). Rational secret sharing and multiparty computation: extended abstract. In *Proc. 36th ACM Symposium on Theory of Computing*, pp. 623–632.

Hirt, M. and U. Maurer (2000). Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology 13*(1), 31–60.

Izmalkov, S., M. Lepinski, and S. Micali (2008). Perfect implementation of normal-form mechanisms. Available at http://people.csail.mit.edu/silvio/. This is an revised version of "Rational secure computation and ideal mechanism design", *Proc. 46th IEEE Symposium on Foundations of Computer Science*, 2005, pp. 585–595.

Kol, G. and M. Naor (2008). Cryptography and game theory: Designing protocols for exchanging information. In *Theory of Cryptography Conference*, pp. 320–339.

Kreps, D. M. and R. B. Wilson (1982). Sequential equilibria. *Econometrica 50*, 863–894.

Lepinski, M., S. Micali, C. Peikert, and A. Shelat (2004). Completely fair SFE and coalition-safe cheap talk. In *Proc. 23rd ACM Symposium on Principles of Distributed Computing*, pp. 1–10.

Lepinski, M., S. Micali, and A. Shelat (2005). Collusion-free protocols. In *Proc. 37th ACM Symposium on Theory of Computing*, pp. 543–552.

Malkhi, D., N. Nisan, B. Pinkas, and Y. Sella (2004). Fairplay—secure two-party computation system. In *Proc. 13th USENIX Security Symposium*, pp. 287–302.

Micali, S. and R. Pass (2006). Local zero knowledge. In *Proc. 38th ACM Symposium on Theory of Computing*, pp. 306–315.

Micali, S. and R. Pass (2007). Precise cryptography. Available at http://www.cs.cornell.edu/~rafael.

Micali, S., R. Pass, and A. Rosen (2006). Input-indistinguishable computation. In *Proc. 47th IEEE Symposium on Foundations of Computer Science*, pp. 367–378.

Micali, S. and A. Shelat (2009). Purely rational secret sharing. In *Proc. TCC 2009*.

Miltersen, P., J. Nielsen, and N. Triandopoulos (2009). Privacy-enhancing first-price auctions using rational cryptography. E-print Archive.

Moses, Y. (1988). Resource-bounded knowledge. In *Proc. Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 261–276.

Neumann, J. v. and O. Morgenstern (1947). *Theory of Games and Economic Behavior* (2nd ed.). Princeton, N.J.: Princeton University Press.

Pass, R. (2006). *A Precise Computational Approach to Knowledge*. Ph. D. thesis, MIT. Available at http://www.cs.cornell.edu/~rafael.

Rubinstein, A. (1986). Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory 39*, 83–96.

Selten, R. (1975). Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory 4*, 25–55.

Shelat, A. and A. Rosen (2000). Rational defense against concurrent attacks. Manuscript.

Shoham, Y. and M. Tennenholtz (2005). Non-cooperative computing: Boolean functions with correctness and exclusivity. *Theoretical Computer Science 343*(1–2), 97–113.

Appendix

A Precise Secure Computation

In this section, we review the notion of precise secure computation [Micali and Pass 2006; Micali and Pass 2007], which is a strengthening of the traditional notion of secure computation [Goldreich, Micali, and Wigderson 1987]. We consider a system where players are connected through secure (i.e., authenticated and private) point-to-point channels. We consider a malicious adversary that is allowed to corrupt a subset of the m players before the interaction begins; these players may then deviate arbitrarily from the protocol. Thus, the adversary is *static*; it cannot corrupt players based on history.

An *m-ary functionality* is specified by a random process that maps vectors of inputs to vectors of outputs (one input and one output for each player). That is, formally, $f : ((\{0,1\}^*)^m \times \{0,1\}^\infty) \rightarrow (\{0,1\}^*)^m$ and $f = (f_1, \ldots, f_m)$. We often abuse notation and suppress the random bitstring r, writing $f(\vec{x})$ or $f_i(\vec{x})$. (We can think of $f(\vec{x})$ and $f_i(\vec{x})$ as random variables.) A machine profile $\vec{M}$ *computes* f if for all $n \in N$, all inputs $\vec{x} \in (\{0,1\}^n)^m$ the output vector of the players after an execution of $\vec{M}$ on input $\vec{x}$ (where M_i gets input x_i) is identically distributed to $f^n(\vec{x})$.[14] As usual, the security of protocol $\vec{M}$ for computing a function f is defined by comparing the *real execution* of $\vec{M}$ with an *ideal execution* where all players directly talk to a trusted third party (i.e., a mediator) computing f. In particular, we require that the outputs of the players in both of these executions cannot be distinguished, and additionally that the view of the adversary in the real execution can be reconstructed by the ideal-execution adversary (called the *simulator*). Additionally, *precision* requires that the running-time of the simulator in each run of the ideal execution is closely related to the running time of the real-execution adversary in the (real-execution) view output by the simulator.

The ideal execution Let f be an m-ary functionality. Let $\tilde{A}$ be a probabilistic polynomial-time machine (representing the ideal-model adversary) and suppose that $\tilde{A}$ controls the players in $Z \subseteq [m]$. We characterize the *ideal execution of f* given adversary $\tilde{A}$ using a function denoted $\text{IDEAL}_{f,\tilde{A}}$ that maps an input vector $\vec{x}$, an auxiliary input z, and a tuple $(r_{\tilde{A}}, r_f) \in (\{0,1\}^\infty)^2$ (a random string for the adversary $\tilde{A}$ and a random string for the trusted third party) to a triple $(\vec{x}, \vec{y}, v)$, where $\vec{y}$ is the output vector of the players $1, \ldots, m$, and v is the output of the adversary $\tilde{A}$ on its tape given input $(z, \vec{x}, r_{\tilde{A}})$, computed according to the following three-stage process.

In the first stage, each player i receives its input x_i. Each player $i \notin Z$ next sends x_i to the trusted party. (Recall that in the ideal execution, there is a trusted third party.) The adversary $\tilde{A}$ determines the value $x_i' \in \{0,1\}^*$ a player $i \in Z$ sends to the trusted party. We assume that the system is synchronous, so the trusted party can tell if some player does not send a message; if player i does not send a message i is taken to have sent λ. Let $\vec{x}'$ be the vector of values received by the trusted party. In the second stage, the trusted party computes $y_i = f_i(\vec{x}', r_f)$ and sends y_i to P_i for every $i \in [m]$. Finally, in the third stage, each player $i \notin Z$ outputs the value y_i received from the trusted party. The adversary $\tilde{A}$ determines the output of the players $i \in Z$. $\tilde{A}$ finally also outputs an arbitrary value v (which is supposed to be the "reconstructed" view of the real-execution adversary A). Let $\mathsf{view}_{f,\tilde{A}}(\vec{x}, z, \vec{r})$ denote the the view of $\tilde{A}$ in this execution. We occasionally abuse notation and suppress the random strings, writing $\text{IDEAL}_{f,\tilde{A}}(\vec{x}, z)$ and $\mathsf{view}_{f,\tilde{A}}(\vec{x}, z)$; we can think of $\text{IDEAL}_{f,\tilde{A}}(\vec{x}, z)$ and $\mathsf{view}_{f,\tilde{A}}(\vec{x}, z)$ as random variables.

[14] A common relaxation requires only that the output vectors are statistically close. All our results can be modified to apply also to protocols that are satisfy only such a "statistical" notion of computation.

The real execution Let f be an m-ary functionality, let Π be a protocol for computing f, and let A be a machine that controls the same set Z of players as $\tilde{A}$. We characterize the *real execution of* Π given adversary A using a function denoted $\textsc{real}_{\Pi,A}$ that maps an input vector $\vec{x}$, an auxiliary input z, and a tuple $\vec{r} \in (\{0,1\}^\infty)^{m+1-|Z|}$ ($m - |Z|$ random strings for the players not in Z and a random string for the adversary A), to a triple $(\vec{x}, \vec{y}, v)$, where $\vec{y}$ is the output of players $1, \ldots, m$, and v is the view of A that results from executing protocol Π on inputs $\vec{x}$, when players $i \in Z$ are controlled by the adversary A, who is given auxiliary input z. As before, we often suppress the vector of random bitstrings $\vec{r}$ and write $\textsc{real}_{\Pi,A}(\vec{x}, z)$.

We now formalize the notion of precise secure computation. For convenience, we slightly generalize the definition of [Micali and Pass 2006] to consider *general adversary structures* [Hirt and Maurer 2000]. More precisely, we assume that the specification of a secure computation protocol includes a set $\mathcal{Z}$ of subsets of players, where the adversary is allowed to corrupt only the players in one of the subsets in $\mathcal{Z}$; the definition of [Micali and Pass 2006; Goldreich, Micali, and Wigderson 1987] considers only *threshold adversaries* where $\mathcal{Z}$ consists of all subsets up to a pre-specified size k. We first provide a definition of precise computation in terms of running time, as in [Micali and Pass 2006], although other complexity functions could be used; we later consider general complexity functions.

Let $\textsc{steps}$ be the complexity function that, on input a machine M and a view v, roughly speaking, gives the number of "computational steps" taken by M in the view v. In counting computational steps, we assume a representation of machines such that a machine M, given as input an encoding of another machine A and an input x, can emulate the computation of A on input x with only linear overhead. (Note that this is clearly the case for "natural" memory-based models of computation. An equivalent representation is a universal Turing machine that receives the code it is supposed to run on one input tape.)

In the following definition, we say that a function is *negligible* if it is asymptotically smaller than the inverse of any fixed polynomial. More precisely, a function $\nu : \mathbb{N} \to \mathbb{R}$ is negligible if, for all $c > 0$, there exists some n_c such that $\nu(n) < n^{-c}$ for all $n > n_c$.

Roughly speaking, a computation is secure if the ideal execution cannot be distinguished from the real execution. To make this precise, a *distinguisher* is used. Formally, a distinguisher gets as input a bitstring z, a triple $(\vec{x}, \vec{y}, v)$ (intuitively, the output of either $\textsc{ideal}_{f,\tilde{A}}$ or $\textsc{real}_{\Pi,A}$ on $(\vec{x}, z)$ and some appropriate-length tuple of random strings) and a random string r, and outputs either 0 or 1. As usual, we typically suppress the random bitstring and write, for example, $D(z, \textsc{ideal}_{f,\tilde{A}}(\vec{x}, z))$ or $D(z, \textsc{real}_{\Pi,A}(\vec{x}, z))$.

Definition A.1 (Precise Secure Computation) *Let f be an m-ary function, Π a protocol computing f, $\mathcal{Z}$ a set of subsets of $[m]$, $p : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and $\epsilon : \mathbb{N} \to \mathbb{R}$. Protocol Π is a $\mathcal{Z}$-*secure computation of f with precision p and ϵ-statistical error *if, for all $Z \in \mathcal{Z}$ and every real-model adversary A that controls the players in Z, there exists an ideal-model adversary $\tilde{A}$, called the* simulator*, that controls the players in Z such that, for all $n \in N$, all $\vec{x} = (x_1, \ldots, x_m) \in (\{0,1\}^n)^m$, and all $z \in \{0,1\}^*$, the following conditions hold:*

1. *For every distinguisher D,*

$$\left|\Pr_U[D(z, \textsc{real}_{\Pi,A}(\vec{x}, z)) = 1] - \Pr_U[D(z, \textsc{ideal}_{f,\tilde{A}}(\vec{x}, z))] = 1\right| \leq \epsilon(n).$$

2. $\Pr_U[\textsc{steps}(\tilde{A}, \mathsf{view}_{f,\tilde{A}}(\vec{x}, z)) \leq p(n, \textsc{steps}(A, \tilde{A}(\mathsf{view}_{f,\tilde{A}}(\vec{x}, z)))]) = 1.$[15]

[15] Note that the three occurrences of $\Pr_U$ in the first two clauses represent slightly different probability measures,

Π *is a* $\mathcal{Z}$-secure computation of f with precision p and (T,ϵ)-computational error *if it satisfies the two conditions above with the adversary A and the distinguisher D restricted to being computable by a TM with running time bounded by $T(\cdot)$.*

Protocol Π *is a* $\mathcal{Z}$-secure computation of f with statistical precision p *if there exists some negligible function ϵ such that Π is a $\mathcal{Z}$-secure computation of f with precision p and ϵ-statistical error. Finally, protocol* Π *is a* $\mathcal{Z}$-secure computation of f with computational precision p *if for every polynomial T, there exists some negligible function ϵ such that Π is a $\mathcal{Z}$-secure computation of f with precision p and (T,ϵ)-computational error.*

The traditional notion of secure computation is obtained by replacing condition 2 with the requirement that the worst-case running-time of $\tilde{A}$ is polynomially related to the worst-case running time of A.

The following theorems were provided by Micali and Pass [2007, 2006], using the results of Ben-Or, Goldwasser and Wigderson [1988] and Goldreich, Micali and Wigderson [1987]. Let $\mathcal{Z}_t^m$ denote all the subsets of $[m]$ containing t or less elements. An m-ary functionality f is said to be *well-formed* if it essentially ignores arguments that are not in $(\{0,1\}^n)^m$ for some n. More precisely, if there exist j, j' such that $|x_j| \neq |x_{j'}|$, then $f_i(\vec{x}) = \lambda$ for all $i \in [m]$. (See [Goldreich 2004, p. 617] for motivation and more details.)

Theorem A.2 *For every well-formed m-ary functionality f, there exists a precision function p such that $p(n,t) = O(t)$ and a protocol Π that $\mathcal{Z}^m_{\lceil m/3\rceil-1}$-securely computes f with precision p and* 0*-statistical error.*

This result can also be extended to more general adversary structures by relying on the results of [Hirt and Maurer 2000]. We can also consider secure computation of specific 2-party functionalities.

Theorem A.3 *Suppose that there exists an enhanced trapdoor permutation.*[16] *For every well-formed* 2*-ary functionality f where only one party gets an output (i.e., $f_1(\cdot) = 0$), there exists a a precision function p such that $p(n,t) = O(t)$ and protocol Π that $\mathcal{Z}_1^2$-securely computes f with computational-precision p.*

Micali and Pass [2006] also obtain unconditional results (using statistical security) for the special case of zero-knowledge proofs. We refer the reader to [Micali and Pass 2006; Pass 2006] for more details.

A.1 Weak Precise Secure Computation

Universal implementation is not equivalent to precise secure computation, but to a (quite natural) weakening of it. *Weak precise secure computation*, which we are about to define, differs from precise secure computation in the following respects:

- Just as in the traditional definition of zero knowledge [Goldwasser, Micali, and Rackoff 1989], precise zero knowledge requires that for every adversary, there exists a simulator that, on all inputs, produces an interaction that no distinguisher can distinguish from the real interaction. This simulator must work for all inputs and all distinguishers. In analogy with the notion of "weak zero knowledge" [Dwork, Naor, Reingold, and Stockmeyer 2003], we here switch

although this is hidden by the fact that we have omitted the superscripts. The first occurrence of $\Pr_U$ should be $\Pr_U^{m-|Z|+3}$, since we are taking the probability over the $m+2-|Z|$ random inputs to $\text{REAL}_{f,A}$ and the additional random input to D; similarly, the second and third occurrences of $\Pr_U$ should be $\Pr_U^3$.

[16] See [Goldreich 2004] for a definition of enhanced trapdoor permutations; the existence of such permutations is implied by the "standard" hardness of factoring assumptions.

the order of the quantifiers and require instead that for every input distribution Pr over $\vec{x} \in (\{0,1\}^n)^m$ and $z \in \{0,1\}^*$, and every distinguisher D, there exists a (precise) simulator that "tricks" D; in essence, we allow there to be a different simulator for each distinguisher. As argued by Dwork et al. [2003], this order of quantification is arguably reasonable when dealing with concrete security. To show that a computation is secure in every concrete setting, it suffices to show that, in every concrete setting (where a "concrete setting" is characterized by an input distribution and the distinguisher used by the adversary), there is a simulator.

- We further weaken this condition by requiring only that the probability of the distinguisher outputting 1 on a real view be (essentially) no higher than the probability of outputting 1 on a simulated view. In contrast, the traditional definition requires these probabilities to be (essentially) equal. If we think of the distinguisher outputting 1 as meaning that the adversary has learned some important feature, then we are saying that the likelihood of the adversary learning an important feature in the real execution is essentially no higher than that of the adversary learning an important feature in the "ideal" computation. This condition on the distinguisher is in keeping with the standard intuition of the role of the distinguisher.

- We allow the adversary and the simulator to depend not only on the probability distribution, but also on the particular security parameter n (in contrast, the definition of [Dwork, Naor, Reingold, and Stockmeyer 2003] is *uniform*). That is why, when considering weak precise secure computation with (T,ϵ)-computational error, we require that the adversary A and the simulator D be computable by *circuits* of size at most $T(n)$ (with a possibly different circuit for each n), rather than a Turing machine with running time $T(n)$. Again, this is arguably reasonable in a concrete setting, where the security parameter is known.

- We also allow the computation not to meet the precision bounds with a small probability. The obvious way to do this is to change the requirement in the definition of precise secure computation by replacing 1 by $1-\epsilon$, to get

  $$\Pr_U[\textsc{steps}(\tilde{A}, \mathsf{view}_{f,\tilde{A}}(\vec{x},z)) \leq p(n, \textsc{steps}(A, \tilde{A}(\mathsf{view}_{f,\tilde{A}}(\vec{x},z))] \geq 1-\epsilon(n),$$

  where n is the input length. We change this requirement in two ways. First, rather than just requiring that this precision inequality hold for all $\vec{x}$ and z, we require that the probability of the inequality holding be at least $1-\epsilon$ for all distributions Pr over $\vec{x} \in (\{0,1\}^n)^m$ and $z \in \{0,1\}^*$.

  The second difference is to add an extra argument to the distinguisher, which tells the distinguisher whether the precision requirement is met. In the real computation, we assume that the precision requirement is always met, thus, whenever it is not met, the distinguisher can distinguish the real and ideal computations. We still want the probability that the distinguisher can distinguish the real and ideal computations to be at most $\epsilon(n)$. For example, our definition disallows a scenario where the complexity bound is not met with probability $\epsilon(n)/2$ and the distinguisher can distinguish the computations with (without taking the complexity bound into account) with probability $\epsilon(n)/2$.

- In keeping with the more abstract approach used in the definition of robust implementation, the definition of weak precise secure computation is parametrized by the abstract complexity measure $\mathscr{C}$, rather than using $\textsc{steps}$. This just gives us a more general definition; we can always instantiate $\mathscr{C}$ to measure running time.

Definition A.4 (Weak Precise Secure Computation) *Let f, Π, $\mathcal{Z}$, p, and ϵ be as in the definition of precise secure computation, and let $\vec{\mathscr{C}}$ be a complexity function. Protocol* Π *is a* weak

$\mathcal{Z}$-secure computation of f with $\vec{\mathscr{C}}$-precision p and ϵ-statistical error *if, for all $n \in N$, all $Z \in \mathcal{Z}$, all real-execution adversaries A that control the players in Z, all distinguishers D, and all probability distributions* Pr *over* $(\{0,1\}^n)^m \times \{0,1\}^*$*, there exists an ideal-execution adversary $\tilde{A}$ that controls the players in Z such that*

$$\begin{aligned}&\Pr{}^+(\{(\vec{x},z) : D(z, \text{REAL}_{\Pi,A}(\vec{x},z), 1) = 1\})\\ &\quad - \Pr{}^+(\{(\vec{x},z) : D(z, \text{IDEAL}_{f,\tilde{A}}(\vec{x},z), \mathsf{precise}_{Z,A,\tilde{A}}(n, \mathsf{view}_{f,\tilde{A}}(\vec{x},z))) = 1\}) \le \epsilon(n),\end{aligned}$$

where $\mathsf{precise}_{Z,A,\tilde{A}}(n,v) = 1$ *if and only if* $\mathscr{C}_Z(\tilde{A}, v) \le p(n, \mathscr{C}_Z(A, \tilde{A}(v)))$.[17] Π *is a* weak $\mathcal{Z}$-secure computation of f with $\vec{\mathscr{C}}$-precision p and (T,ϵ)-computational error *if it satisfies the condition above with the adversary A and the distinguisher D restricted to being computable by a randomized circuit of size $T(n)$. Protocol* Π *is a* $\mathcal{Z}$-weak secure computation of f with statistical $\vec{\mathscr{C}}$-precision p *if there exists some negligible function ϵ such that Π is a $\mathcal{Z}$-weak secure computation of f with precision p and statistical ϵ-error. Finally, Protocol* Π *is a* $\mathcal{Z}$-weak secure computation of f with computational $\vec{\mathscr{C}}$-precision p *if for every polynomial $T(\cdot)$, there exists some negligible function ϵ such that Π is a $\mathcal{Z}$-weak secure computation of f with precision p and (T,ϵ)-computational error.*

Our terminology suggests that weak precise secure computation is weaker than precise secure computation. This is almost immediate from the definitions if $\mathscr{C}_Z(M,v) = \text{STEPS}(M,v)$ for all $Z \in \mathcal{Z}$. A more interesting setting considers a complexity measure that can depend on $\text{STEPS}(M,v)$ and the size of the description of M. It directly follows by inspection that Theorems A.2 and A.3 also hold if, for example, $\mathscr{C}_Z(M,v) = \text{STEPS}(M,v) + O(|M|)$ for all $Z \in \mathcal{Z}$, since the simulators in those results incur only a constant additive overhead in size. (This is not a coincidence. As argued in [Micali and Pass 2006; Pass 2006], the definition of precise simulation guarantees the existence of a "universal" simulator S, with "essentially" the same precision, that works for *every* adversary A, provided that S also gets the code of A; namely given a real-execution adversary A, the ideal-execution adversary $\tilde{A} = S(A)$.[18] Since $|S| = O(1)$, it follows that $|\tilde{A}| = |S| + |A| = O(|A|)$.) That is, we have the following variants of Theorems A.2 and A.3:

Theorem A.5 *For every well-formed m-ary functionality f, $\mathscr{C}_Z(M,v) = \text{STEPS}(M,v) + O(|M|)$ for all sets Z, there exists a precision function p such that $p(n,t) = O(t)$ and a protocol Π that weak $\mathcal{Z}^m_{\lceil m/3\rceil-1}$-securely computes f with $\vec{\mathscr{C}}$-precision p and 0-statistical error.*

Theorem A.6 *Suppose that there exists an enhanced trapdoor permutation, and $\mathscr{C}_Z(M,v) = \text{STEPS}(M,v) + O(|M|)$ for all sets Z. For every well-formed 2-ary functionality f where only one party gets an output (i.e., $f_1(\cdot) = \lambda$), there exists a precision function p such that $p(n,t) = O(t)$ and a protocol Π that weak $\mathcal{Z}^2_1$-securely computes f with computational $\vec{\mathscr{C}}$-precision p.*

It is easy to see that the theorems above continue to hold when considering "coarse" versions of the above complexity functions, where, say, n^2 computational steps (or size) correspond to one unit of complexity (in canonical machine game with input length n).

B Proof of Theorem 4.2

In this section, we prove Theorem 4.2. Recall that for one direction of our main theorem we require that certain operations, like moving output from one tape to another, do not incur any additional complexity. We now make this precise.

[17] Recall that $\Pr^+$ denotes the product of Pr and $\Pr_U$ (here, the first $\Pr^+$ is actually $\Pr^{+(m+3-|Z|)}$, while the second is $\Pr^{+3}$).

[18] This follows by considering the simulator S for the universal TM (which receives the code to be executed as auxiliary input).

Recall that in the definition of a secure computation, the ideal-execution adversary, M_Z, is an algorithm that controls the players in Z and finally provides an output for each of the players it controls and additionally produces an output of its own (which is supposed to be the reconstructed view of the real-execution adversary). Roughly speaking, a complexity function is output-invariant if M_Z can "shuffle" content between its tapes at no cost.

Definition B.1 *A complexity function $\mathscr{C}$ is* output-invariant *if, for every set Z of players, there exists some canonical player $i_Z \in Z$ such that the following three conditions hold:*

1. **(Outputting view)** *For every machine M_Z controlling the players in Z, there exists some machine M'_Z with the same complexity as M_Z such that the output of $M'_Z(v)$ is identical to $M_Z(v)$ except that player i_Z outputs $y; v$, where y is the output of i_Z in the execution of $M_Z(v)$ (i.e., M'_Z is identical to M_Z with the only exception being that player i_Z also outputs the view of M'_Z).*

2. **(Moving content to a different tape)** *For every machine M'_Z controlling players Z, there exists some machine M_Z with the same complexity as M_Z such that the output of $M_Z(v)$ is identical to $M'_Z(v)$ except if player i_Z outputs $y; v'$ for some $v' \in \{0,1\}^*$ in the execution of $M'_Z(v)$. In that case, the only difference is that player i_Z outputs only y and $M_Z(v)$ outputs v'.*

3. **(Duplicating content to another output tape)** *For every machine M'_Z controlling players Z, there exists some machine M_Z with the same complexity as M_Z such that the output of $M_Z(v)$ is identical to $M'_Z(v)$ except if player i_Z outputs $y; v'$ for some $v' \in \{0,1\}^*$ in the execution of $M'_Z(v)$. In that case, the only difference is that $M_Z(v)$ outputs v'.*

Note that the only difference between condition 2 and 3 is that in condition 2, player i_z only outputs y, whereas in condition 3 it still outputs its original output $y; v'$.

We stress that we need to consider output-invariant complexity functions only to show that universal implementation implies precise secure computation.

We now prove each direction of Theorem 4.2 separately, to make clear what assumptions we need for each part. We start with the "only if" direction.

Theorem B.2 *Let $\vec{M}, f, \mathcal{F}, \mathcal{Z}$ be as in the statement of Theorem 4.2, and let $\vec{\mathscr{C}}$ be an $\vec{M}$-acceptable complexity function. If $\vec{M}$ is an abort-preserving weak $\mathcal{Z}$-secure computation of f with $\mathscr{C}$-precision p and error ϵ, then $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ.*

Proof: Suppose that $\vec{M}$ is a weak $\mathcal{Z}$-secure computation of f with $\vec{\mathscr{C}}$-precision p and ϵ-statistical error. Since $\vec{M}$ computes f, for every game $G \in \mathcal{G}^{\vec{\mathscr{C}}}$, the action profile induced by $\vec{M}$ in (G, comm) is identically distributed to the action profile induced by $\vec{\Lambda}^{\mathcal{F}}$ in $(G, \mathcal{F})$. We now show that $(\vec{M}, \mathsf{comm})$ is a $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ.

Claim B.3 $(\vec{M}, \mathsf{comm})$ *is a $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ.*

Proof: Let $G \in \mathcal{G}^{\vec{\mathscr{C}}}$ be a game with input length n such that $\vec{\Lambda}^{\mathcal{F}}$ is a $p(n, \cdot)$-robust $\mathcal{Z}$-safe equilibrium in $(G, \mathcal{F})$. We show that $\vec{M}$ is a $\mathcal{Z}$-safe $\epsilon(n)$-equilibrium in (G, comm). Recall that this is equivalent to showing that no coalition of players $Z \in \mathcal{Z}$ can increase their utility by more than $\epsilon(n)$ by deviating from their prescribed strategies. In other words, for all $Z \in \mathcal{Z}$ and machines M'_Z, we need to show that

$$U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) \leq U_Z^{(G,\mathsf{comm})}(M^b_Z, \vec{M}_{-Z}) + \epsilon(n).$$

Suppose, by way of contradiction, that there exists some machine M'_Z such that

$$U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) > U_Z^{(G,\mathsf{comm})}(M^b_Z, \vec{M}_{-Z}) + \epsilon(n). \tag{1}$$

We now obtain a contradiction by showing that there exists some other game $\tilde{G}$ that is at most a $p(n,\cdot)$-speedup of G and a machine $\tilde{M}_Z$ such that

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(\tilde{G},\mathcal{F})}((\Lambda^{\mathcal{F}})^b_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}). \tag{2}$$

This contradicts the assumption that $\Lambda^{\mathcal{F}}$ is a p-robust equilibrium.

Note that the machine M'_Z can be viewed as a real-execution adversary controlling the players in Z. The machine $\tilde{M}_Z$ will be defined as the simulator for M'_Z for some appropriately defined input distribution Pr on $T \times \{0,1\}^*$ and distinguisher D. Intuitively, Pr will be the type distribution in the game G (where z is nature's type), and the distinguisher D will capture the utility function u_Z. There is an obvious problem with using the distinguisher to capture the utility function: the distinguisher outputs a single bit, whereas the utility function outputs a real. To get around this problem, we define a probabilistic distinguisher that outputs 1 with a probability that is determined by the expected utility; this is possible since the game is normalized, so the expected utility is guaranteed to be in $[0,1]$. We also cannot quite use the same distribution for the machine M as for the game G. The problem is that, if $G \in \mathcal{G}$ is a canonical game with input length n, the types in G have the form $x;z$, where $x \in \{0,1\}^n$. The protocol $\Lambda^{\mathcal{F}}_i$ in (G,comm) ignores the z, and sends the mediator the x. On the other hand, in a secure computation, the honest players provide their input (i.e., their type) to the mediator. Thus, we must convert a type $x_i;z_i$ of a player i in the game G to a type x for $\Lambda^{\mathcal{F}}_i$.

More formally, we proceed as follows. Suppose that G is a canonical game with input length n, and the type space of G is T. Given $\vec{t} = (x_1;z_1,\ldots,x_n;z_n,t_N) \in T$, define $\vec{t}^D$ by taking $t^D_i = x_1;z_i$ if $i \in Z$, $t^D_i = x_i$ if $i \notin Z$, and $t^D_N = t_N;z1;\ldots;z_m$. Say that $(\vec{x},z)$ is *acceptable* if there is some (necessarily unique) $\vec{t} \in T$, $z = t^D_1;\ldots;t^D_n;t^D_N$, and $\vec{x} = (t^D_1,\ldots,t^D_m)$. If $(\vec{x},z)$ is acceptable, let $\vec{t}_{\vec{x},z}$ be the element of T determined this way. If $\Pr_G$ is the probability distribution over types, $\Pr(\vec{x},z) = \Pr_G(\vec{t}_{\vec{x},z})$ if $(\vec{x},z)$ is acceptable, and $\Pr(\vec{x},z) = 0$ otherwise.

Define the probabilistic distinguisher D as follows: if $\mathit{precise} = 0$ or $(\vec{x},z)$ is not acceptable, then $D(z,(\vec{x},\vec{y},\mathit{view}),\mathit{precise}) = 0$; otherwise $D(z,(\vec{x},\vec{y},\mathit{view}),\mathit{precise}) = 1$ with probability $u_Z(\vec{t}_{\vec{x},z},\vec{y},\mathscr{C}_Z(M'_Z,\mathit{view}),\vec{c_0}_{-Z})$.

Since we can view M'_Z as a real-execution adversary controlling the players in Z, the definition of weak precise secure computation guarantees that, for the distinguisher D and the distribution Pr described above, there exists a simulator $\tilde{M}_Z$ such that

$$\begin{aligned}&\Pr{}^+(\{(\vec{x},z) : D(z,\mathrm{REAL}_{\vec{M},M'_Z}(\vec{x},z),1) = 1\})\\ &\quad - \Pr{}^+(\{(\vec{x},z) : D(z,\mathrm{IDEAL}_{f,\tilde{M}_Z}(\vec{x},z),\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\mathsf{view}_{f,\tilde{M}_Z}(\vec{x},z)) = 1\}) \le \epsilon(n).\end{aligned} \tag{3}$$

We can assume without loss of generality that if $\tilde{M}_Z$ sends no messages and outputs nothing, then $\tilde{M}_Z = \perp$. (This can only make its complexity smaller and thus make D output 1 more often.)

We next define a new complexity function $\vec{\tilde{\mathscr{C}}}$ that, by construction, will be at most a $p(n,\cdot)$-speedup of $\vec{\mathscr{C}}$. Intuitively, this complexity function will consider the speedup required to make up for the "overhead" of the simulator $\tilde{M}_Z$ when simulating M'_Z. To ensure that the speedup is not too large, we incur it only on views where the simulation by $\tilde{M}_Z$ is "precise". Specifically, let the complexity function $\vec{\tilde{\mathscr{C}}}$ be identical to $\vec{\mathscr{C}}$, except that if $\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\tilde{v}) = 1$ and $\mathscr{C}(\tilde{M}_Z,\tilde{v}) \ge \mathscr{C}(M'_Z,v)$, where v is the view output by $\tilde{M}_Z(\tilde{v})$, then $\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\tilde{v}) = \mathscr{C}_Z(M'_Z,v)$. (Note that $\tilde{v}$ is a view for the ideal execution. M'_Z runs in the real execution, so we need to give it as input the view output by $\tilde{M}_Z$ given view $\tilde{v}$, namely v. Recall that the simulator $\tilde{M}_Z$ is trying to reconstruct the view of M'_Z. Also, note that we did not define $\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\tilde{v}) = \mathscr{C}_Z(M'_Z,v)$ if $\mathscr{C}(\tilde{M}_Z,\tilde{v}) < \mathscr{C}(M'_Z,v)$, for then $\tilde{\mathscr{C}}_Z$ would not be a speedup of $\mathscr{C}_Z$. Finally, to ensure that $\tilde{\mathscr{C}}_Z$ assigns 0 complexity only to $\bot$, as is required for a complexity function. Note that if $M'_Z = \bot$ then, without loss of generality, $\tilde{M}_Z = \bot$ as well; this directly follows from the fact that $\vec{M}$ is abort-preserving.) By construction, $\tilde{\mathscr{C}}_Z$ is at most a $p(n,\cdot)$-speedup of $\mathscr{C}_Z$. Let $\tilde{G}$ be identical to G except that the complexity function is $\vec{\tilde{\mathscr{C}}}$. It is immediate that $\tilde{G}$ is at most a $p(n,\cdot)$-speedup of G.

We claim that it follows from the definition of D that

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z,\vec{\Lambda}^{\mathcal{F}}_{-Z}) \geq \Pr{}^+(\{(\vec{x},z) : D(z,\text{IDEAL}_{f,\tilde{M}_Z}(\vec{x},z),\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\mathsf{view}_{f,\tilde{M}}(\vec{x},z))) = 1\}). \tag{4}$$

To see this, let $a_Z(\vec{t},\vec{r})$ (resp., $a_i(\vec{t},\vec{r})$) denote the output that $\tilde{M}_Z$ places on the output tapes of the players in Z (resp., the output of player $i \notin Z$) when the strategy profile $(\tilde{M}_Z,\Lambda^{\mathcal{F}}_{-Z})$ is used with mediator $\mathcal{F}$, the type profile is $\vec{t}$, and the random strings are $\vec{r}$. (Note that these outputs are completely determined by $\vec{t}$ and $\vec{r}$.) Similarly, let $\mathit{view}_{\tilde{M}_Z}(\vec{t},\vec{r})$ and $\mathit{view}_{\Lambda_i^{\mathcal{F}}}(\vec{t},\vec{r})$ denote the views of the adversary and player $i \notin Z$ in this case.

Since each type profile $\vec{t} \in T$ is $t_{\vec{x},z}$ for some $(\vec{x},z)$, and $\Pr_G(t_{\vec{x},z}) = \Pr(\vec{x},z)$, we have

$$\begin{array}{rl} & U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z,\vec{\Lambda}^{\mathcal{F}}_{-Z}) \\ = & \sum_{\vec{t},\vec{r}} \Pr^+_G(\vec{t},\vec{r}) u_Z(\vec{t},(a_Z(\vec{t},\vec{r}),\vec{a}_{-Z}(\vec{t},\vec{r})),(\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t},\vec{r})),\vec{\tilde{\mathscr{C}}}_{-Z}(\Lambda^{\mathcal{F}}_{-Z},\mathit{view}_{-Z}(\vec{t},\vec{r})))) \\ = & \sum_{\vec{x},z,\vec{r}} \Pr^+(\vec{x},z,\vec{r}) u_Z(\vec{t}_{\vec{x},z},(a_Z(\vec{t}_{\vec{x},z},\vec{r}),\vec{a}_{-Z}(\vec{t}_{\vec{x},z},\vec{r})),(\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r})),\vec{c}_{0-Z})). \end{array}$$

In the third line, we use the fact that $\tilde{\mathscr{C}}_i(\Lambda_i^{\mathcal{F}},\mathit{view}_i(\vec{t},\vec{r})) = \mathscr{C}_i(\Lambda_i^{\mathcal{F}},\mathit{view}_i(\vec{t},\vec{r})) = c_0$ for all $i \notin Z$, since $\mathscr{C}$ is $\vec{M}$-acceptable. Thus, it suffices to show that, for all $\vec{x}$, z, and $\vec{r}$,

$$\begin{array}{rl} & u_Z(\vec{t}_{\vec{x},z},(a_Z(\vec{t}_{\vec{x},z},\vec{r}),\vec{a}_{-Z}(\vec{t}_{\vec{x},z},\vec{r})),\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r})),\vec{c}_{0-Z}) \\ \geq & \Pr_U(D(z,\text{IDEAL}_{f,\tilde{M}_Z}(\vec{x},z,\vec{r}),\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\mathsf{view}_{f,\tilde{M}}(\vec{x},z,\vec{r}))) = 1). \end{array} \tag{5}$$

This inequality clearly holds if $\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\mathsf{view}_{f,\tilde{M}}(\vec{x},z),\vec{r}) = 0$, since in that case the right-hand side is 0.[19] Next consider the case when $\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\mathsf{view}_{f,\tilde{M}_Z}(\vec{x},z,\vec{r})) = 1$. In this case, by the definition of D, the right-hand side equals

$$u_Z(\vec{t}_{\vec{x},z},(a_Z(\vec{t}_{\vec{x},z},\vec{r}),\vec{a}_{-Z}(\vec{t}_{\vec{x},z},\vec{r})),(\mathscr{C}_Z(M'_Z,v_Z(\vec{t}_{\vec{x},z},\vec{r})),\vec{c}_{0-Z})),$$

where $v_Z(\vec{t}_{\vec{x},z},\vec{r}) = \tilde{M}_Z(\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r}))$ (i.e., the view output by $\tilde{M}_Z$). By the definition of $\tilde{\mathscr{C}}$, it follows that when $\mathscr{C}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r})) \geq \mathscr{C}_Z(M'_Z,v_Z(\vec{t}_{\vec{x},z},\vec{r}))$ and $\mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n,\mathsf{view}_{f,\tilde{M}_Z}(\vec{x},z,\vec{r})) = 1$, then $\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r})) = \mathscr{C}_Z(M'_Z,v_Z(\vec{t}_{\vec{x},z}))$, and (5) holds with $\geq$ replaced by $=$. On the other hand, when $\mathscr{C}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r})) < \mathscr{C}_Z(M'_Z,v_Z(\vec{t}_{\vec{x},z},\vec{r}))$, then $\tilde{\mathscr{C}}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r})) = \mathscr{C}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r}))$, and thus $\mathscr{C}_Z(M'_Z,v_Z(\vec{t}_{\vec{x},z},\vec{r})) > \tilde{\mathscr{C}}_Z(\tilde{M}_Z,\mathit{view}_{\tilde{M}_Z}(\vec{t}_{\vec{x},z},\vec{r}))$; (5) then holds by the monotonicity of u_Z.

Similarly, we have that

$$\Pr{}^+(\{(\vec{x},z) : D(z,\text{REAL}_{\vec{M},M'_Z}(\vec{x},z),1) = 1\}) = U_Z^{(G,\mathsf{comm})}(M'_Z,\vec{M}_{-Z}). \tag{6}$$

[19] Note that we here rely on the fact that G is $\mathscr{C}$-natural and hence normalized, so that the range of u_Z is $[0,1]$.

In more detail, a similar argument to that for (4) shows that it suffices to show that, for all $\vec{x}$, z, and $\vec{r}$,

$$\begin{array}{rl} & u_Z(\vec{t}_{\vec{x},z}, a_Z(\vec{t}_{\vec{x},z},\vec{r}), \vec{a}_{-Z}(\vec{t}_{\vec{x},z},\vec{r}), \mathscr{C}_Z(M'_Z, \mathit{view}_{M'_Z}(\vec{t}_{\vec{x},z},\vec{r})), \mathscr{C}_{-Z}(M_{-Z}, \mathit{view}_{M_{-Z}}(\vec{t}_{\vec{x},z},\vec{r}))) \\ = & \Pr_U(D(z, \text{REAL}_{\vec{M},M'_Z}(\vec{x},z,\vec{r}),1)=1), \end{array}$$

where $a_Z(\vec{t},\vec{r})$, $a_i(\vec{t},\vec{r})$, $\mathit{view}_{M'_Z}(\vec{t},\vec{r})$, and $\mathit{view}_{M_i}(\vec{t},\vec{r})$ are appropriately defined outputs and views in an execution of $(M'_Z, \vec{M'}_Z)$. Since $\vec{C}$ is $\vec{M}$-acceptable, $\mathscr{C}_{-Z}(M_{-Z}, \mathit{view}_{M_{-Z}}(\vec{t}_{\vec{x},z},\vec{r})) = \vec{c_0}_{-Z}$, and Equation 6 follows.

It now follows immediately from (3), (4), and (6) that

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) \geq U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) - \epsilon(n). \tag{7}$$

Combined with (1), this yields

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(G,\mathsf{comm})}(M^b_Z, \vec{M}_{-Z}). \tag{8}$$

Since $\vec{M}$ and $\mathcal{F}$ both compute f (and thus must have the same distribution over outcomes), it follows that

$$U_Z^{(G,\mathsf{comm})}(M^b_Z, \vec{M}_{-Z}) = U_Z^{(G,\mathcal{F})}((\Lambda^{\mathcal{F}})^b_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) = U_Z^{(\tilde{G},\mathcal{F})}((\Lambda^{\mathcal{F}})^b_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}). \tag{9}$$

For the last equality, recall that $\tilde{G}$ is identical to G except for the complexity of $\tilde{M}_Z$ (and hence the utility of strategy profiles involving $\tilde{M}_Z$). Thus, the last equality follows once we show that $(\Lambda^{\mathcal{F}})^b_Z \neq \tilde{M}_Z$. This follows from the various technical assumptions we have made. If $(\Lambda^{\mathcal{F}})^b_Z = \tilde{M}_Z$, then $\tilde{M}_Z$ sends no messages (all the messages sent by $\Lambda^{\mathcal{F}}$ to the communication mediator are ignored, since they have the wrong form), and does not output anything (since messages from the communication mediator are not signed by $\mathcal{F}$). Thus, $\tilde{M}_Z$ acts like $\perp$. By assumption, this means that $\tilde{M}_Z = \perp$, so $\tilde{M}_Z \neq (\Lambda^{\mathcal{F}})^b_Z$.

From (8) and (9), we conclude that

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(\tilde{G},\mathcal{F})}((\Lambda^{\mathcal{F}})^b_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}),$$

which gives the desired contradiction. ∎

It remains to show that $(\vec{M}, \mathsf{comm})$ is also a *strong* $(\mathcal{G}^{\mathscr{C}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ. That is, if $\perp$ is a $p(n,\cdot)$-best response to $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ in $(G, \mathcal{F})$ then $\perp$ an ϵ-best response to $\vec{M}_{-Z}$ in (G, comm). Suppose, by way of contradiction, that there exists some M'_Z such that

$$U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) > U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}) + \epsilon(n). \tag{10}$$

It follows using the same proof as in Claim B.3 (see Equation 7) that there exists a game $\tilde{G}$ that is at most a $p(n,\cdot)$ speedup of G and a machine $\tilde{M}_Z$ such that

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}) - \epsilon(n).$$

Combined with (10), this yields

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}). \tag{11}$$

Since $\vec{M}$ is an abort-preserving computation of f and $\mathcal{F}$ computes f (and thus must have the same distribution over outcomes), it follows that

$$U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}) = U_Z^{(G,\mathcal{F})}(\perp, \vec{\Lambda}^{\mathcal{F}}_{-Z}) = U_Z^{(\tilde{G},\mathcal{F})}(\perp, \vec{\Lambda}^{\mathcal{F}}_{-Z}). \tag{12}$$

The last equality follows as in the proof of Claim B.3, since $\tilde{G}$ is identical to G except for the complexity of $\tilde{M}_Z$. Combining 11 and 12 we have

$$U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(\tilde{G},\mathcal{F})}(\perp, \vec{\Lambda}^{\mathcal{F}}_{-Z}).$$

But this contradicts the assumption that $\perp$ is a $p(n,\cdot)$-robust best response to $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ in $(G,\mathcal{F})$. ∎

We now prove the "if" direction of Theorem 4.2. For this direction, we need the assumption that $\mathscr{\vec{C}}$ is output-invariant. Moreover, we get a slightly weaker implication: we show only that for every ϵ, ϵ' such that $\epsilon' < \epsilon$ it holds that strong universal implementation with error ϵ' implies weak secure computation with error ϵ. (After proving this result, we introduce some additional restrictions on $\mathscr{C}$ that suffice to prove the implication for the case when $\epsilon' = \epsilon$.) However, we no longer need the assumption that $\vec{M}$ is abort-preserving.

Theorem B.4 *Suppose that $\vec{M}, f, \mathcal{F}, \mathcal{Z}$ are as above, $\epsilon' < \epsilon$, and $\vec{\mathscr{C}}$ is an $\vec{M}$-acceptable output-invariant complexity function. If $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ', then $\vec{M}$ is a weak $\mathcal{Z}$-secure computation of f with $\mathscr{C}$-precision p and error ϵ.*

Proof: Let $(\vec{M}, \mathsf{comm})$ be a $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error $\epsilon(\cdot)$. We show that $\vec{M}$ $\mathcal{Z}$-securely computes f with $\mathscr{C}$-precision $p(\cdot,\cdot)$ and error $\epsilon(\cdot)$. Suppose, by way of contradiction, that there exists some $n \in I\!N$, a distribution Pr on $(\{0,1\}^n)^m \times \{0,1\}^*$, a subset $Z \in \mathcal{Z}$, a distinguisher D, and a machine $M'_Z \in \mathcal{M}$ that controls the players in Z such that for all machines $\tilde{M}_Z$,

$$\begin{aligned}&\Pr{}^+(\{(\vec{x},z) : D(z, \textsc{real}_{\vec{M},M'_Z}(\vec{x},z), 1) = 1\})\\ &\quad - \Pr{}^+(\{(\vec{x},z) : D(z, \textsc{ideal}_{f,\tilde{M}_Z}(\vec{x},z), \mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n, \mathsf{view}_{f,\tilde{M}_Z}(\vec{x},z) = 1))) > \epsilon(n).\end{aligned} \tag{13}$$

To obtain a contradiction we consider two cases: $M'_Z = \perp$ or $M'_Z \neq \perp$.

Case 1: $M'_Z = \perp$. We define a game $G \in \mathcal{G}^{\vec{\mathscr{C}}}$ such that $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ is a p-robust $\mathcal{Z}$-safe equilibrium in the game $(G,\mathcal{F})$, and show that

$$U_Z^{(G,\mathsf{comm})}(M'_Z, M_{-Z}) > U_Z^{(G,\mathsf{comm})}(M^b_Z, \vec{M}_{-Z}) + \epsilon(n),$$

which contradicts the assumption that $\vec{M}$ is a $(\mathcal{G}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$.

Intuitively, G is such that the strategy $\perp$ (which is the only one that has complexity 0) gets a utility that is determined by the probability with which the distinguisher D outputs 1 (on input the type and action profile). On the other hand, all other strategies (i.e., all strategies with positive complexity) get the same utility d. If d is selected so that the probability of D outputting 1 in $(G,\mathcal{F})$ is at most d, it follows that $\vec{\Lambda}^{\mathcal{F}}$ is a p-robust Nash equilibrium. However, $\perp$ will be a profitable deviation in (G, comm).

In more detail, we proceed as follows. Let

$$d = \Pr{}^+(\{(\vec{x},z) : D(z, \textsc{ideal}_{f,\perp}(\vec{x},z), 1) = 1\}). \tag{14}$$

Consider the game $G = ([m], \mathcal{M}, \Pr, \vec{\mathscr{C}}, \vec{u})$, where $u_{Z'}(\vec{t}, \vec{a}, \vec{c}) = 0$ for all $Z' \neq Z$ and

$$u_Z((t_1, \ldots, t_m, t_N), \vec{a}, (c_Z, \vec{c}_{-Z})) = \begin{cases} \Pr_U(D(t_N, ((t_1, \ldots, t_m), \vec{a}, \lambda), 1) = 1) & \text{if } c_Z = 0 \\ d & \text{otherwise,} \end{cases}$$

where, as before, $\Pr_U$ is the uniform distribution on $\{0,1\}^\infty$ (D's random string). It follows from the definition of D (and the fact that only $\perp$ can have complexity 0) that, for all games $\tilde{G}$ that are speedups of G and all machines $\tilde{M}_Z$, $U_Z^{(\tilde{G},\mathcal{F})}(\tilde{M}_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) \leq d$.

Since $M_Z^b \neq \perp$ (since $\mathscr{C}$ is $\vec{M}$-acceptable and $\vec{M}$ thus has complexity $c_0 > 0$), we conclude that $\vec{\Lambda}^{\mathcal{F}}$ is a p-robust Z-safe Nash equilibrium. In contrast (again since $M_Z^b \neq \perp$), $U_Z^{(G,\mathsf{comm})}(M_Z^b, \vec{M}_{-Z}) = d$. But, since $M'_Z = \perp$ by assumption, we have

$$\begin{array}{rll} & U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}) & \\ = & U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) & \\ = & \Pr^+(\{(\vec{x}, z) : D(z, \text{REAL}_{\vec{M},M'_Z}(\vec{x}, z), 1) = 1\}) & \\ > & d + \epsilon(n) & \text{[by (13) and (14)],} \end{array}$$

which is a contradiction.

Case 2: $M'_Z \neq \perp$. To obtain a contradiction, we first show that, without loss of generality, M'_Z lets one of the players in $\mathcal{Z}$ output the view of M'_Z. Next, we define a game $G \in \mathcal{G}^{\vec{\mathscr{C}}}$ such that $\perp$ is a $p(n, \cdot)$-best response to $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ in $(G, \mathcal{F})$. We then show that

$$U_Z^{(G,\mathsf{comm})}(M'_Z, M_{-Z}) \geq U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}) + \epsilon(n),$$

which contradicts the assumption that $\vec{M}$ is a strong $(\mathcal{G}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error $\epsilon' < \epsilon$.

To prove the first step, note that by the first condition in the definition of output-invariant, there exists some canonical player $i_Z \in Z$ and a machine M''_Z controlling the players in Z with the same complexity as M'_Z such that the output of $M''_Z(v)$ is identical to $M'_Z(v)$, except that player i_Z outputs $y; v$, where y is the output of i_Z in the execution of $M'_Z(v)$. We can obtain a counterexample with M''_Z just as well as with M'_Z by considering the distinguisher D' which is defined identically to D, except that if $y_{i_Z} = y; v'$ for some $v' \in \{0,1\}^*$, then $D'(z, (\vec{x}, \vec{y}, v), \mathit{precise}) = D(z, (\vec{x}, \vec{y}', v'), \mathit{precise})$, where $\vec{y}'$ is identical to $\vec{y}$ except that $y'_{i_Z} = y$. Consider an adversary $\tilde{M}'_Z$. We claim that

$$\begin{array}{l} \Pr^+(\{(\vec{x}, z) : D'(z, \text{REAL}_{\vec{M},M''_Z}(\vec{x}, z), 1) = 1\}) \\ \quad - \Pr^+(\{(\vec{x}, z) : D'(z, \text{IDEAL}_{f,\tilde{M}'_Z}(\vec{x}, z), \mathsf{precise}_{Z,M'_Z,\tilde{M}'_Z}(n, \mathsf{view}_{f,\tilde{M}'_Z} = (\vec{x}, z))) = 1\}) > \epsilon(n). \end{array} \tag{15}$$

By definition of D' and M''_Z, it follows that

$$\Pr^+(\{(\vec{x}, z) : D(z, \text{REAL}_{\vec{M},M'_Z}(\vec{x}, z), 1) = 1\}) = \Pr^+(\{(\vec{x}, z) : D'(z, \text{REAL}_{\vec{M},M''_Z}(\vec{x}, z), 1) = 1\}). \tag{16}$$

By the second condition of the definition of output-invariant, there exists a machine $\tilde{M}_Z$ with the same complexity as $\tilde{M}'_Z$ such that the output of $\tilde{M}_Z(v)$ is identical to $\tilde{M}'_Z(v)$ except that if player i_Z outputs $y; v'$ for some $v' \in \{0,1\}^*$ in the execution by $\tilde{M}'_Z(v)$, then it outputs only y in the execution by $\tilde{M}_Z(v)$; furthermore, $\tilde{M}_Z(v)$ outputs v' on its own output tape (representing the reconstructed view of M'_Z). It follows that

$$\begin{array}{rl} & \Pr^+(\{(\vec{x}, z) : D'(z, \text{IDEAL}_{f,\tilde{M}'_Z}(\vec{x}, z), \mathsf{precise}_{Z,M''_Z,\tilde{M}'_Z}(n, \mathsf{view}_{f,\tilde{M}'}(\vec{x}, z))) = 1\}) \\ = & \Pr^+(\{(\vec{x}, z) : D(z, \text{IDEAL}_{f,\tilde{M}_Z}(\vec{x}, z), \mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n, \mathsf{view}_{f,\tilde{M}}(\vec{x}, z))) = 1\}). \end{array} \tag{17}$$

Equation (15) is now immediate from (13), (16), and (17).

We now define a game $G \in \mathcal{G}^{\vec{\mathscr{C}}}$ such that $\perp$ is a $p(n,\cdot)$-robust best response to $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ in $(G, \mathcal{F})$, but $\perp$ is not an ϵ-best response to $\vec{M}_{-Z}$ in (G, comm). Intuitively, G is such that simply playing $\perp$ guarantees a high payoff. However, if a coalition controlling Z can provide a view that cannot be "simulated", then it receives an even higher payoff. By definition, it will be hard to find such a view in the mediated game. However, by our assumption that $\vec{M}$ is not a secure computation protocol, it is possible for the machine controlling Z to obtain such a view in the cheap-talk game.

In more detail, we proceed as follows. Let

$$d = \sup_{\tilde{M}_Z \in \mathbf{M}} \Pr{}^+(\{(\vec{x}, z) : D(z, \text{IDEAL}_{f,\tilde{M}_Z}(\vec{x}, z), \mathsf{precise}_{Z,M'_Z\tilde{M}_Z}(n, \mathsf{view}_{f,\tilde{M}_Z}(\vec{x}, z))) = 1\}). \tag{18}$$

Consider the game $G = ([m], \mathcal{M}, \Pr, \vec{\mathscr{C}}, \vec{u})$, where $u_{Z'}(\vec{t}, \vec{a}, \vec{c}) = 0$ for all $Z' \neq Z$ and

$$u_Z(\vec{t}, \vec{a}, \vec{c}) = \begin{cases} d & \text{if } c_Z = 0 \\ \Pr_U(D(t_N, ((t_1, \ldots, t_m), \vec{a}, v), 1) = 1) & \text{if } a_{i_Z} = y; v,\ 0 < p(n, c_Z) \leq p(n, \mathscr{C}_Z(M'_Z, v)) \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $G \in \mathcal{G}^{\vec{\mathscr{C}}}$.

Claim B.5 *$\perp$ is a $p(n,\cdot)$-robust best response to $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ in $(G, \mathcal{F})$.*

Proof: Suppose, by way of contradiction, that there exists some game $\tilde{G}$ with complexity profile $\vec{\tilde{C}}$ that is at most a $p(n,\cdot)$-speedup of G and a machine M^*_Z such that

$$U_Z^{(\tilde{G},\mathcal{F})}(M^*_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > U_Z^{(\tilde{G},\mathcal{F})}(\perp, \vec{\Lambda}^{\mathcal{F}}_{-Z}). \tag{19}$$

It is immediate from (19) that $M^*_Z \neq \perp$. Thus, it follows from the definition of complexity functions that $\tilde{\mathscr{C}}(M^*_Z, v) \neq 0$ for all $v \in \{0,1\}^*$. That means that when calculating $U^{(\tilde{G},\mathcal{F})}(M^*_Z, \Lambda^{\mathcal{F}}_{-Z})$, the second or third conditions in the definition of u_Z must apply. Moreover, the second condition applies on type $(\vec{x}, z)$ only if a_{i_Z} has the form $y; v$ and $0 < p(n, \tilde{\mathscr{C}}_Z(M^*_Z, \mathsf{view}_{f,M^*_Z}(\vec{x}, z)) \leq p(n, \mathscr{C}_Z(M'_Z, v))$. Since $\tilde{\mathscr{C}}$ is at most a p-speedup of $\mathscr{C}$, the latter condition implies that $0 < \mathscr{C}_Z(M^*_Z, \mathsf{view}_{f,M^*_Z}(\vec{x}, z)) \leq p(n, \mathscr{C}_Z(M'_Z, v))$. Hence, $U_Z^{(\tilde{G},\mathcal{F})}(M^*_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z})$ is at most

$$\Pr{}^+(\{(\vec{x}, z) : D(z, \text{IDEAL}'_{f,M^*_Z}(\vec{x}, z), \mathsf{precise}'_{Z,M'_Z,M^*_Z}(n, \mathsf{view}_{f,M^*_Z}(\vec{x}, z))) = 1\}),$$

where IDEAL'_{f,M^*_Z} is defined identically to IDEAL, except that y_{i_Z} (the output of player i_Z) is parsed as $y; v$, and v is taken as the output of M^*_Z (representing the reconstructed view of M'_Z); analogously, $\mathsf{precise}'$ is defined just as $\mathsf{precise}$ except that v is taken as the view of M'_Z reconstructed by M^*_Z (if $y_{i_Z} = y; v$). Since $\mathscr{C}_Z(\perp, v) = 0$ for all v, the definition of u_Z guarantees that $U_Z^{(\tilde{G},\mathcal{F})}(\perp, \vec{\Lambda}^{\mathcal{F}}_{-Z}) = d$. It thus follows from (19) that

$$U_Z^{(\tilde{G},\mathcal{F})}(M^*_Z, \vec{\Lambda}^{\mathcal{F}}_{-Z}) > d.$$

Thus,

$$\Pr{}^+(\{(\vec{x}, z) : D(z, \text{IDEAL}'_{f,M^*_Z}(\vec{x}, z), \mathsf{precise}'_{Z,M'_Z,M^*_Z}(n, \mathsf{view}_{f,M^*_Z}(\vec{x}, z))) = 1\}) > d.$$

The third condition of the definition of output-invariant complexity implies that there must exist some M^{**}_Z such that

$$\Pr{}^+(\{(\vec{x}, z) : D(z, \text{IDEAL}_{f,M^{**}_Z}(\vec{x}, z), \mathsf{precise}_{Z,M'_Z,M^{**}_Z}(n, \mathsf{view}_{f,M^{**}_Z}(\vec{x}, z))) = 1\}) > d,$$

which contradicts (18). Thus, $\vec{\Lambda}^{\mathcal{F}}$ is a p-robust $\mathcal{Z}$-equilibrium of $(G, \mathcal{F})$. ▮

Since $(\vec{M}, \mathsf{comm})$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}}}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ, and $\perp$ is a $p(n, \cdot)$-robust best response to $\vec{\Lambda}^{\mathcal{F}}_{-Z}$ in $(G, \mathcal{F})$, it must be the case that $\perp$ is an ϵ-best response to $\vec{M}_{-Z}$ in (G, comm). However, by definition of u_Z, we have that

$$\begin{array}{rl} & U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) \\ = & \sum_{\vec{t},\vec{r}} \Pr^+(\vec{t}, \vec{r}) u_Z(\vec{t}, M'_Z(\mathit{view}_{M'_Z}(\vec{t}, \vec{r})), \vec{M}_{-Z}(\mathit{view}_{\vec{M}_{-Z}}(\vec{t}, \vec{r})), \mathscr{C}_Z(M'_Z, \mathit{view}_{M'_Z}(\vec{t}, \vec{r}), \vec{\mathscr{C}}_{-Z}(\vec{M}_{-Z}, \mathit{view}_{\vec{M}_{-Z}}(\vec{t}, \vec{r}))) \\ = & \Pr^+(\{(\vec{x}, z) : D(z, \text{REAL}_{\vec{M},M'_Z}(n, \vec{x}, z), 1) = 1\}) \end{array}$$

where $\mathit{view}_{M'_Z}(\vec{t}, \vec{r})$ (resp., $\mathit{view}_{\vec{M}_{-Z}}(\vec{t}, \vec{r})$) denotes the view of M'_Z (resp. $\vec{M}_{-Z}$) when the strategy profile $(M'_Z, \vec{M}_{-Z})$ is used with mediator comm. The second equality follows from the fact that player i_Z outputs the view of M'_Z. Recall that (13) holds (with strict inequality) for *every* machine $\tilde{M}_Z$. It follows that

$$\begin{array}{rl} & U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) \\ = & \Pr^+(\{(\vec{x}, z) : D(z, \text{REAL}_{\vec{M},M'_Z}(\vec{x}, z), 1) = 1\}) \\ \geq & \sup_{\tilde{M} \in \mathbf{M}} \Pr^+(\{(\vec{x}, z) : D(z, \text{IDEAL}_{f,\tilde{M}_Z}(\vec{x}, z), \mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n, \mathsf{view}_{f,\tilde{M}_Z}(\vec{x}, z) = 1))) + \epsilon(n) \\ = & d + \epsilon(n) \end{array} \tag{20}$$

where the last equality follows from (18).

Since $U_Z^{(G,\mathsf{comm})}(\perp, \vec{M}_{-Z}) = d$, this is a contradiction. This completes the proof of the theorem. ▮

Note that if the set

$$S = \{\Pr^+(\{(\vec{x}, z) : D(z, \text{IDEAL}_{f,\tilde{M}_Z}(\vec{x}, z), \mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n, \mathsf{view}_{f,\tilde{M}_Z}(\vec{x}, z))) = 1) : \tilde{M} \in \mathbf{M}\}$$

has a maximal element d, then by (13), equation (20) would hold with strict inequality, and thus theorem B.4 would hold even if $\epsilon' = \epsilon$. We can ensure this by introducing some additional technical (but natural) restrictions on $\mathscr{C}$. For instance, suppose that $\mathscr{C}$ is such that for every complexity bound c, the number of machines that have complexity at most c is finite, i.e., for every $c \in I\!N$, there exists some constant N such that $|\{M \in \mathbf{M} : \exists v \in \{0,1\}^* \ \mathscr{C}(M, v) \leq c\}| = N$. Under this assumption S is finite and thus has a maximal element.

C Proof of Theorem 4.3

We again separate out the two directions of the proof.

Theorem C.1 *Let $\vec{M}, f, \mathcal{F}, \mathcal{Z}$ be as above, and let $\vec{\mathscr{C}}$ be an $\vec{M}$-acceptable efficient complexity function, and p a precision function. If $(\vec{M}, \mathsf{comm})$ is an abort-preserving weak $\mathcal{Z}$-secure computation of f with computational $\mathscr{C}$-precision p, then for every polynomial T, there exists some negligible function ϵ such that $\vec{M}$ is a strong $(\mathcal{G}^{\vec{\mathscr{C}},T}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ.*

Proof Sketch: The proof follows the same lines as that of Theorem B.2. Assume that $\vec{M}$ computes f with computational $\vec{\mathscr{C}}$-precision p. Since $\vec{M}$ computes f, it follows that for every polynomial T and game $G \in \mathcal{G}^{\vec{\mathscr{C}},T}$, the action profile induced by $\vec{M}$ in (G, comm) is identically distributed to the action profile induced by $\vec{\Lambda}^{\mathcal{F}}$ in $(G, \mathcal{F})$. We now show1 that, for every polynomial T, there exists some negligible function ϵ such that $(\vec{M}, \mathsf{comm})$ is a $(\mathcal{G}^{\vec{\mathscr{C}},T}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ. Assume, by way of contradiction, that there exists polynomials T and g and infinitely many $n \in N$ such that the following conditions hold:

- there exists some game $G \in \mathcal{G}^{\vec{\mathscr{C}},T}$ with input length n such that $\vec{\Lambda}^{\mathcal{F}}$ is a $p(n,\cdot)$-robust $\mathcal{Z}$-safe equilibrium in $(G, \mathcal{F})$;
- there exists some machine $M'_Z \in \mathcal{M}^{T(n)}$ such that

$$U_Z^{(G,\mathsf{comm})}(M'_Z, \vec{M}_{-Z}) > U_Z^{(G,\mathsf{comm})}(M^b_Z, \vec{M}_{-Z}) + \frac{1}{g(n)}. \tag{21}$$

It follows using the same proof as in Theorem B.2 that this contradicts the weak secure computation property of $\vec{M}$. To apply this proof, we need to make sure that the distinguisher D constructed can be implemented by a polynomial-sized circuit. However, since by our assumption $\vec{\mathscr{C}}$ is efficient and $\vec{u}$ is $T(\cdot)$-sized computable, it follows that D can be constructed efficiently. We also need to verify that the "simulator" algorithm $\tilde{M}_Z$ is a valid strategy in $\tilde{G}$. At first sight, this seems to be a problem, since the size of $\tilde{M}_Z$ might be bigger than $T(n)$. Note, however, that the definition of robust Nash equilibrium lets us consider *any* game $\tilde{G}$ that is identical to G except for the complexity profile *and the machine set* $\mathcal{M}$. In particular, if we let the machine set $\mathcal{M}$ in $\tilde{G}$ be the full set of machines $\mathbf{M}$ (just as it was in the proof of Theorem B.2), we ensure that $\tilde{M}$ is a valid strategy in $\tilde{G}$ (although it might not be one in G).

Strong universal implementation follows in an analogous way. ∎

Theorem C.2 *Let $\vec{M}, f, \mathcal{F}, \mathcal{Z}$ be as above, let $\vec{\mathscr{C}}$ be a $\vec{M}$-acceptable output-invariant efficient complexity function, and let p be an efficient precision function. If, for every polynomial T, there exists some negligible function ϵ such that $(\vec{M}, \mathsf{comm})$ is a $(\mathcal{G}^{\vec{\mathscr{C}},T}, \mathcal{Z}, p)$-universal implementation of $\mathcal{F}$ with error ϵ, then $\vec{M}$ is a weak $\mathcal{Z}$-secure computation of f with computational $\vec{\mathscr{C}}$-precision p.*

Proof Sketch: Suppose, by way of contradiction, that there exist polynomials T and g such that for infinitely many $n \in N$, there exists a distribution Pr on $(\{0,1\}^n)^m \times \{0,1\}^*$, a subset $Z \in \mathcal{Z}$, a $T(n)$-sized distinguisher D, and a $T(n)$-sized machine $M'_Z \in \mathcal{M}^{T(n)}$ that controls the players in Z such that, for all machines $\tilde{M}_Z$ (not necessarily $T(n)$-bounded),

$$\begin{aligned}&\Pr{}^+(\{(\vec{x},z) : D(z, \text{REAL}_{\vec{M},M'_Z}(\vec{x},z), 1) = 1\})\\ &\quad - \Pr{}^+(\{(\vec{x},z) : D(z, \text{IDEAL}_{f,\tilde{M}_Z}(\vec{x},z), \mathsf{precise}_{Z,M'_Z,\tilde{M}_Z}(n, \mathsf{view}_{f,\tilde{M}_Z}(\vec{x},z) = 1))) > \tfrac{1}{g(n)}.\end{aligned} \tag{22}$$

Consider any such n. As in the proof of Theorem B.4, we consider two cases: $M'_Z = \perp$ or $M' \neq \perp$. In both cases, we construct a game G that contradicts the assumption that $\mathcal{M}$ is a strong universal implementation. The first thing that needs to be changed is that we need to prove that the game G constructed is in $\mathcal{G}^{\vec{\mathscr{C}},T'}$ for some polynomial T'. That is, we need to prove that $\vec{u}$ can be computed by poly-sized circuits (given than D is poly-size computable). We do not know how to show that the actual utility function u_Z constructed in the proof of Proposition B.4 can be made efficient. However, for each polynomial g', we can approximate it to within an additive term of $\frac{1}{g'(n)}$ using polynomial-sized circuits (by using repeated sampling of $D(t_N, ((t_1, \ldots, t_m), \vec{a}, v), 1)$ to determine an estimate of $\Pr_U(D(t_N, ((t_1, \ldots, t_m), \vec{a}, v), 1) = 1)$, and by receiving an estimate of d as non-uniform advice). This is sufficient to show that there exists some $T' \geq T$ such that we can approximate $\vec{u}$ to within an additive term of $\frac{1}{g'(n)}$, while ensuring that $G \in \mathcal{G}^{\vec{\mathscr{C}},T'}$. Additionally, we need to make sure that the algorithm M'_Z is a valid strategy in G; this easily follows since M'_Z is

$T(n)$-bounded, and $T' \geq T$. Taken together, it follows using the same proof as in Proposition B.4 that there exists some polynomial g'' such that, in case 1, $\vec{M}$ is not a $\frac{1}{g''(n)}$-equilibrium in (G, comm), and in case 2, $\perp$ is not a $\frac{1}{g''(n)}$-best response to $\vec{M}_{-Z}$ in (G, comm). We reach a contradiction in both cases. ■

This completes the proof of Theorem 4.3.